第 1 章　概　　述

1.1　面向对象探源

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.

——**摘自** https://en.wikipedia.org/wiki/Object-oriented_programming

开宗明义，概念先行。这段摘自 Wikipedia 的英文原文介绍的 Object-oriented programming 在我国大陆译为“面向对象程序设计”。这个术语一般用其缩写 OOP 简称，由来已久，早已响彻业界。后续章节将以这个定义为主线，围绕相关概念展开学习和讨论。本节介绍与此相关的行业大背景，追根溯源，了解面向对象概念、.NET 平台，以及 C#语言的来龙去脉，为深入学习相关理论和技术做好准备。

1.1.1　关于计算

说到计算，人们并不陌生。形容一个人“精于计算”，一般是指其数学功底深厚。这里的计算(computing)，特指与计算机相关的目标导向活动，包括计算机软硬件系统的设计和建造、信息的采集和处理、通信和娱乐媒体的创建与使用，以及用计算机进行科学研究等。简而言之，这里的计算是计算机设计和使用的研究，包括理论、实验和工程等。计算机的主要功能是实现计算的自动化，涉及计算的对象(data，即数据)和计算的过程(algorithm，即算法)。数据和算法用程序(program)来描述，计算自动化的核心任务就是程序设计(programming)。

随着计算理论的日渐成熟和计算系统的飞速发展，计算科学已划分成许多理论和实践领域。从工程角度看，计算机硬件制造和软件开发各自发展，形成了计算机工程和软件工程两大独立的学科。计算机硬件和软件产品集成起来，可应用于不同的领域，形成各种各样的信息系统(相关技术统称为信息技术)。当然，不管怎么演化和划分，程序设计都是最为基本的活动，是各计算学科都不可或缺的内容。计算机科学的发展如图 1-1 所示。

程序由描述计算对象的数据结构和描述计算过程的算法构成，程序设计还涉及表示程序的语言(即程序设计语言)和运行程序的平台(即计算环境)。就程序运行平台来说，从早期基于单机的主机计算到后来基于 Internet 的网络分布计算，计算环境发生着深刻的变革。只有真正了解计算环境的这种翻天覆地的变化，才有可能理解新一代程序设计语言所提供的机制和特性，并快速掌握这些更为先进的程序设计技术和方法。

计算机科学的发展.mp4

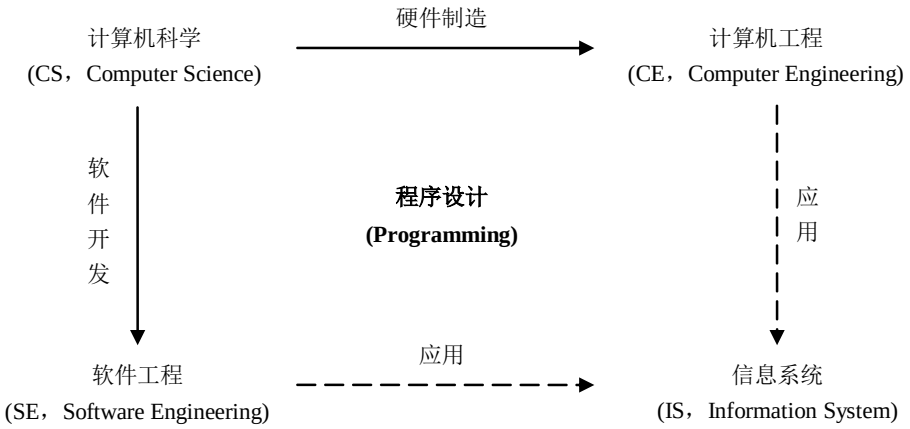

图 1-1　计算机科学的发展

1.1.2　主机计算

现代计算机遵循的是匈牙利数学家约翰·冯·诺依曼(John von Neumann)于 1945 提出的体系结构，如图 1-2 所示。这种体系结构由中央处理器(Central Processing Unit，CPU)、存储器(Memory Unit)、输入设备(Input Device)和输出设备(Input Device)构成。其中，CPU 又由算术/逻辑运算器(Arithmetic/Logic Unit)、处理器寄存器、含有指令寄存器和程序计数器的控制器(Control Unit)构成。由于存储器用于存储数据和指令，这种体系结构的计算机又称为存储程序(stored-program)计算机。

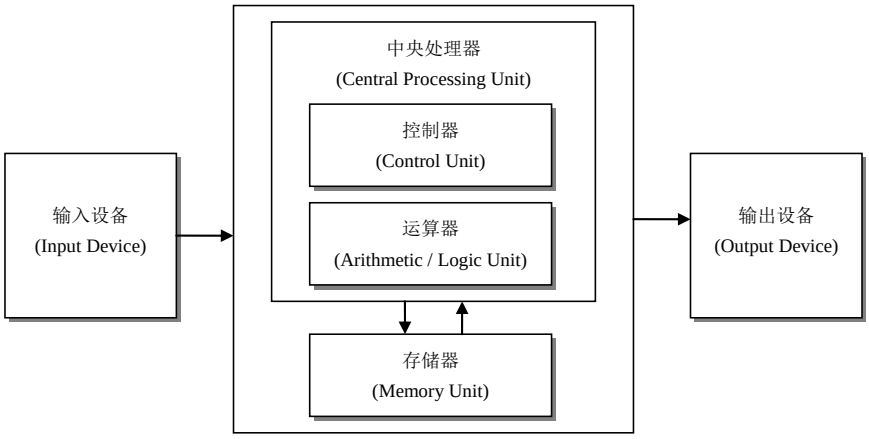

图 1-2　冯·诺依曼体系结构

基于冯·诺依曼体系结构的计算系统(computing system)由硬件(hardware)和软件(software)组成，如图 1-3 所示。硬件是构成计算系统的物理部件的集合，都是有形的物体，包括主机和外部设备两大部分。主机的核心是一块集成电路主板，用于连接计算机的其他部分，包括 CPU、存储器、磁盘驱动器(如 CD、DVD、硬盘等)。主机可以通过主板上的端口或扩展槽连接外部设备，如显示器、键盘、打印机、音箱、麦克风等。软件是能够被硬件存储和执行的指令的集合，一般分为系统软件和应用软件两个部分。系统软件包括设备驱动程序、操作系统和一些用于计算机维护的实用工具软件；应用软件则泛指系统软件之外的所有软件。

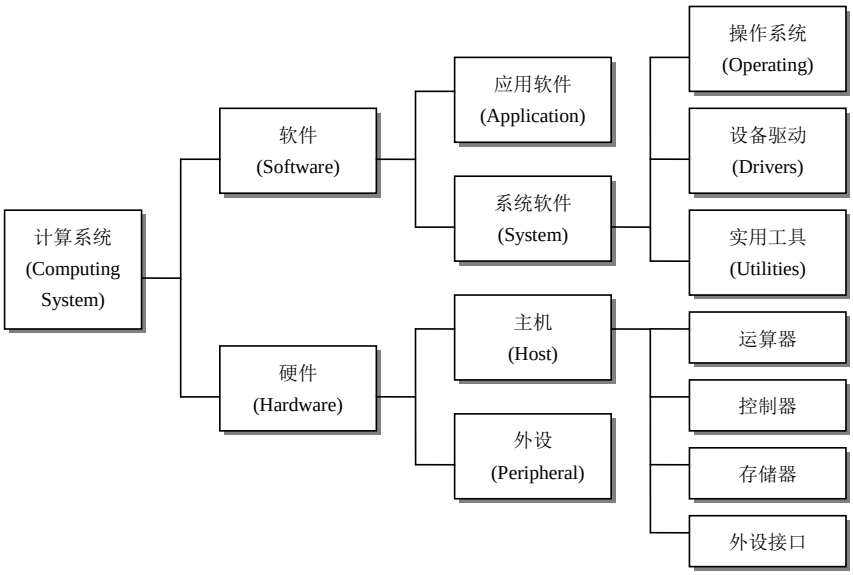

图 1-3　计算系统组成

计算环境是指运行应用程序的平台，包括硬件平台和软件平台，如图 1-4 所示。

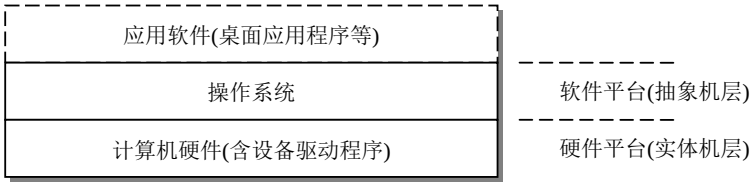

图 1-4　主机计算环境

在图 1-4 中，底层是计算机实体层，可以是各种品牌和类型的计算机。硬件之上是操作系统，用于管理计算机的软硬件资源并提供公共服务，就像一台扩展了硬件功能的虚拟机，一般视为抽象机层。抽象机也可以有多层，如运行 Java 程序的 JVM(Java Virtual Machine)、运行 C#程序的 CLR(Common Language Runtime)等都属于虚拟机。这里的主机计算是指基于单台计算机的程序运行环境，对应的程序设计相对比较简单。

1.1.3　网络分布计算

网络分布计算.mp4

Internet 出现后，随着网络应用需求的飞速增长，网络分布计算逐渐成为新一代计算和应用的主流。这时的计算涉及主机之间的资源共享和协同工作，如图 1-5 所示。

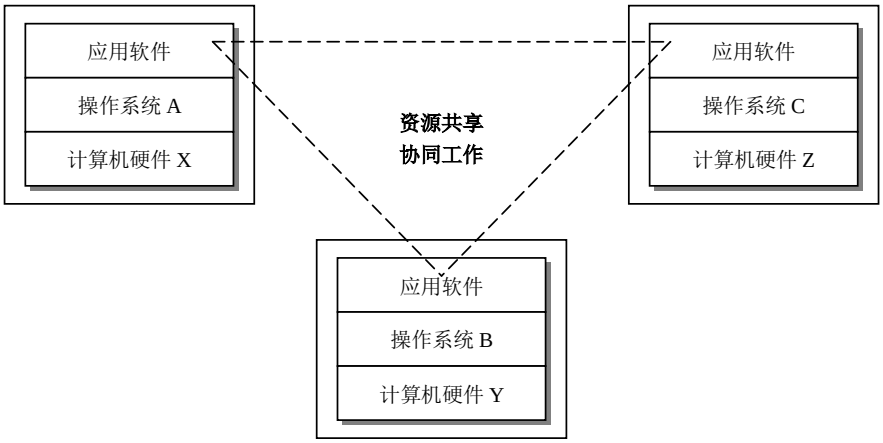

图 1-5　网络分布计算的主要特征

图 1-5 所示的计算环境，真正的挑战来自如何突破软件平台、硬件系统、时间、地点的限制。计算机硬件可能是不同厂商生产的，型号可能不同；操作系统可能是 Unix、Windows 等不同软件；计算机系统可能分布于不同的地点；用户可能在不同的时间使用应用软件。图中看似分布在不同地点不同平台上的应用软件其实是一个整体，如图 1-6 所示。

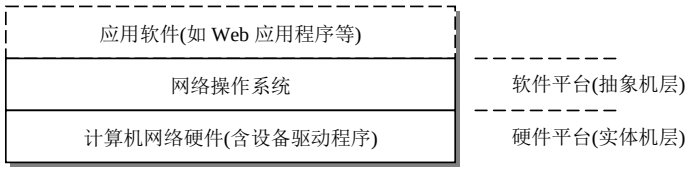

图 1-6　网络分布计算环境

组件技术.mp4

1.1.4　组件技术

在从主机计算向网络分布计算过渡的过程中，软件系统的规模和复杂度呈几何级数增加，程序设计语言和方法都面临着前所未有的挑战。

面对规模越来越大的软件，为了降低复杂度，提高开发效率，人们提出了组件式程序设计方法。组件式方法是“搭积木”思想在程序设计领域的开拓性应用。因为组件(积木)具有可重用性和互操作性，可以通过组件集成来高效地构建复杂的软件系统。

20 世纪 90 年代以来，出现了三种典型的组件技术，即 CORBA、COM、JavaBeans。

1. CORBA

CORBA (Common Object Request Broker Architecture，公共对象请求代理体系结构)是 OMG(Object Management Group，对象管理组织)于 1991 推出的组件技术。OMG 于 1989 年由 3COM、Apple、美国航空、佳能、DG、HP、IBM、Philips、Unisys 和 Sun 等多家公司联合创建，是一个开放型非营利组织，负责制定和维护协同企业应用的计算机工业规范。发展至今，已有八百多家公司、大学和国际组织参与其中。OMG 制定的其他标准还有 UML(Unified Modeling Language，统一建模语言)和 IDL(Interface Definition Language，接口定义语言)等。

2. COM/DCOM/COM+

COM(Component Object Model，组件对象模型)是微软公司于 1993 年提出的一种组件技术，是软件对象组件之间相互通信的一种方式和规范，是一种平台无关、语言中立、位置透明、支持网络的中间件技术。DCOM(Distributed COM，分布式 COM)和 COM+是 COM 的发展，分别于 1996 年和 1999 年推出。

3. JavaBeans

JavaBeans 是 Sun 公司于 1997 年在 Java 的 JDK 1.1 中引入的组件技术，是一个面向对象程序设计接口，可以用它创建可重用的应用程序或能在主流网络操作系统平台配置的程序模块(组件)。

CORBA、COM、JavaBeans 各有优缺点，都面临着不断改进和发展的要求。例如， 继 CORBA 1.0 后，OMG 又分别于 1996 年 8 月推出 2.0、2002 年 7 月推出 3.0，目前的最新标准是 2004 年 3 月 12 日推出的 CORBA 3.0.3。Sun 于 2000 年随 J2EE(Java 2 Platform, Enterprise

Edition，Java 2 平台企业版)引入服务器端的组件技术 EJB(Enterprise JavaBeans，企业级JavaBeans)和网页编程工具 JSP(Java Server Page，Java 服务器网页)，使得 Java 成为一种功能完备的分布式计算环境。COM 源自 OLE(Object Linking and Embedding，对象链接和嵌入)，OLE 源自 DLL(Dynamic Link Libraries，动态链接库)，ActiveX 控件是 COM 的具体应用，ATL(Active Template Library，活动模板库)是开发 COM 的主要工具，也可以用 MFC 直接开发 COM。继 COM+后，微软又于 2002 年推出了.NET 框架，其核心技术就是用来代替 COM 组件功能的 CLR(Common Language Runtime，公共语言运行库)。

这些组件技术是各大公司为使软件开发更符合人类的行为习惯而开发的新技术。利用这些技术，可以开发出各种各样的功能组件，将它们按需组合，就可以构成复杂的应用系统。

这样做不仅能提高软件定制的效率和软件产品的质量，也使得软件系统易于升级和维护。例如可以“现场”替换软件系统中的组件、可以在多个软件系统中重用同一个组件、可以方便地将组件部署到分布式网络环境等。

CORBA、COM、JavaBeans 等组件技术都与面向对象技术密切相关。要掌握组件式程序设计方法，面向对象技术是关键。

1.1.5 面向对象技术

面向对象核心概念.mp4

现在回到本章开篇主题“面向对象程序设计”，中英对照如下：

Object-oriented programming (OOP) is a programming paradigm
面向对象程序设计(OOP)是一种程序设计范式，
based on the concept of “objects”,
它基于“对象”概念，
which may contain
对象可以包含：
data, in the form of fields, often known as attributes;
数据，以字段的形式体现，常称为属性；
and code, in the form of procedures, often known as methods.
代码，以过程的形式体现，常称为方法。

类的世界.mp4

“对象”，也许是学习程序设计技术的路上最易明白的概念了。我们看到的每个从身边走过的人，就是一个个具体的对象。对象有其自身的特性，如年龄、身高等，也有其自身的行为，如走路、微笑等。对应到程序设计，由于派别或翻译等原因，很容易把这些原本简单的概念弄混淆了。这涉及三个“世界”的术语转换问题，如图 1-7 所示。

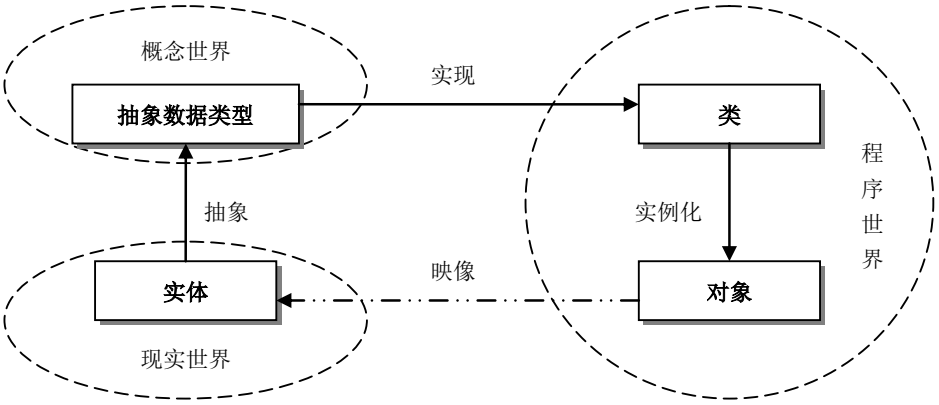

图 1-7　三个世界的变换

首先是从现实世界向概念世界过渡。例如，现实世界中的一个人事部门，有许多实实在在的员工“实体”。要对这些员工进行有效的管理，需要对他们进行了解。人事部门经理分析这些员工，在大脑(概念世界)中对员工信息进行抽象，形成了自己的看法(关注点)，建立了信息模型(即抽象数据类型)，如图 1-8 所示。

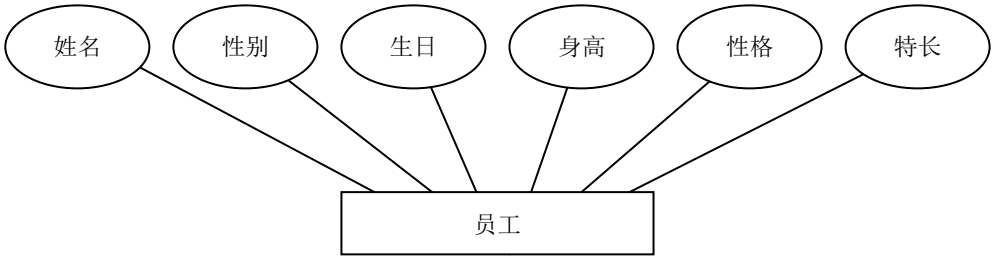

图 1-8　员工信息

当然也要关注他们的行为表现，如签到、写作、说话等。

作为软件工程师，要为这个部门经理开发一套人事管理软件，就得把部门经理的概念模型转换成程序世界的“类”，如图 1-9 所示。

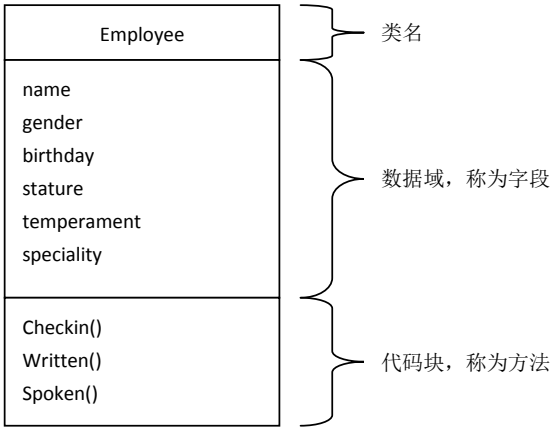

图 1-9　员工类

程序世界的类相当于一个模板，利用这个模板可以创建具体的“对象”。例如：

```
Employee emp = new Employee();   //用 Employee 类实例化一个 emp 对象
emp.name = "张三";               //emp 对象映射了现实世界中张三这个员工
```

本书第 2 章、第 3 章将具体介绍数据字段的表示、代码过程的实现。

1.2 .NET 框架

微软公司于 2000 年 6 月推出了用来代替 COM 的.NET。这是微软面向第三代 Internet 的计算计划，是微软继用 Windows 取代 DOS 之后的又一项战略性举措。.NET 是一个分布式计算环境，提供了一个安全、一致、标准的模型和环境，简化了分布式应用程序开发的难度，能大幅度地提高软件系统的生产率和质量。它面向异构硬件平台、操作系统和网络，为软件提供最大限度的可重用性、互操作性和可扩展性，以实现软件系统之间的智能交互和协同工作，提高整个网络的利用率和效率，特别是企业级的系统集成和资源优化，给开放性企业的生产力水平带来质的飞跃。目前，.NET 已成为 Windows 应用和 Web 应用的主流开发模型。

1.2.1 微软技术的发展

.NET 技术.mp4

微软技术的发展路线如图 1-10 所示。

20 世纪 90 年代末，使用 Microsoft 平台的 Windows 程序设计演化出了许多分支：大多数程序员使用的是 Visual Basic、C 或 C++，使用 C 和 C++ 的程序员中，有的使用 Win32 API(Application Programming Interface，应用程序设计接口)，有的使用 MFC(Microsoft Foundation Classes，微软基础类库)，有的程序员已经转向 COM。

这些技术都有自身的问题。例如，Win32 API 不是面向对象的，使用它比使用 MFC 需要更多的工作量；MFC 是面向对象的，但缺乏一致性；COM 概念简单，但实际编码很复杂且代码也较难阅读。况且，这些程序设计技术主要针对的是桌面应用开发，对 Internet 则显得力不从心。

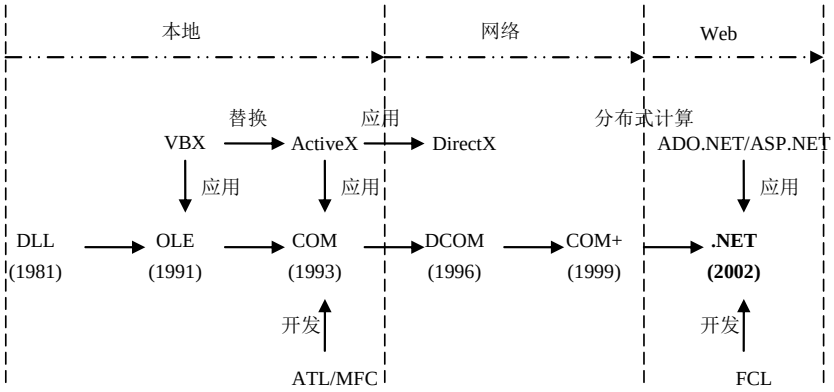

图 1-10 微软组件技术的发展历程

早期的程序代码短小精悍。随着问题规模的越来越大，程序代码也越来越复杂。难以阅读的程序代码必然会给开发和维护带来困难。于是，程序员开始重温那“激情燃烧的岁月”，希望用一种集成的、面向对象的开发框架把一致性和优雅性带回到程序中，回归到代码简洁的时代。为此，他们对下一代计算平台提出了新的要求，希望达到以下目标。

1. 运行环境(Execution Environment)

(1) 安全(Security)；
(2) 多平台(Multiple Platforms)；
(3) 性能(Performance)。

2. 开发环境(Development Environment)

(1) 面向对象的开发环境(Object-Oriented Development Environment)；
(2) 一致的程序设计体验(Consistent Programming Experience)；
(3) 用行业标准进行沟通(Communication Using Industry Standards)；
(4) 简化开发(Simplified Development)；
(5) 语言独立(Language Independence)；
(6) 互操作(Interoperability)。

为了满足这些需求，微软公司开始开发一个能满足这些目标的代码运行环境和应用开发环境，这就是.NET。

.NET 将 Internet 作为构建新一代操作系统的基础，在理念中包含了对操作系统和网络设计思想的延伸。微软计划用.NET 彻底改变软件的开发、发行和使用方式，构建第三代 Internet 平台，解决各种协同合作的问题，实现信息的高效沟通和分享，让整个 Internet 为人们提供全方位的服务。

1.2.2 .NET 规范及其实现

.NET 规范.mp4

微软为.NET 技术制定了一套完整的规范 CLI(Common Language Infrastructure，公共语言基础结构)。CLI 是针对可执行代码格式，以及能执行该代码的运行环境的一种技术规范，包括 CTS(Common Type System，公共类型系统)、CLS(Common Language Specification，公共语言规范)、CIL(Common Intermediate Language，公共中间语言)，以及其他相关的标准化文档、协议和规范等。

CTS 定义了一套类型系统的框架，是被编译器、工具和 CLI 本身所共用的一种统一类型系统。CTS 是一个模型，定义了在声明、使用和管理类型时，CLI 应遵循的规则。CTS 框架使跨语言集成、类型安全和高性能的代码执行成为可能。CLS 是一组语言规则的集合，是语言设计者和框架(类库)设计者之间的一种协定。如果某语言符合 CLS 的所有规则，就是标准的.NET 语言，可与其他.NET 语言跨语言集成；如果某组件使用了 CLS 规定的功能，就是标准的.NET 组件，可与其他.NET 组件交互。CIL 则是一种中性的、与处理器无关的指令语言，任何.NET 程序都可被编译成 CIL 代码，CIL 代码可以被翻译成不同系统平台的机器代码。

微软在 Windows 平台实现的 CLI 就是.NET 框架(Framework)。该框架由程序设计工具、FCL(Framework Class Library，框架类库)和 CLR(Common Language Runtime，公共语言运行机)构成，如图 1-11 所示。

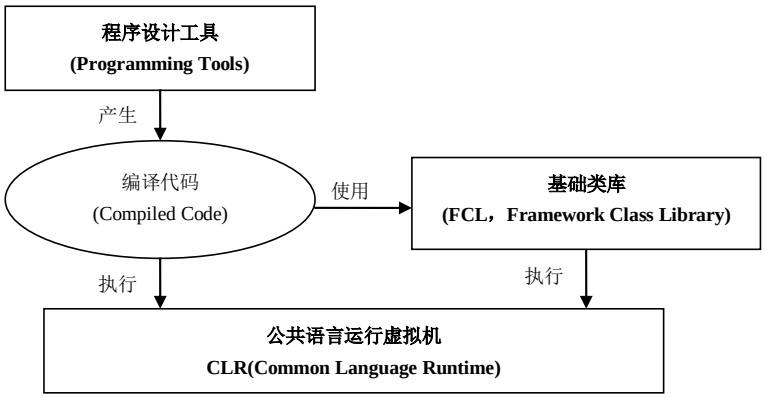

图 1-11 .NET 框架的构成

CLR 是程序的运行环境，它管理程序的运行，包括内存管理、垃圾回收、代码安全验

证、代码执行、线程管理，以及异常处理等。BCL 是一个大型类库，.NET 框架和程序员都可以使用。程序员编码和调试时使用的程序设计工具包括 Visual Studio 集成开发环境(IDE)、.NET 兼容的编译器(例如 C#、Visual Basic .NET、F#、IronRuby、managed ++等)、调试器，以及诸如 ASP.NET、WCF 的 Web 服务器端开发技术。

CLR 改变了传统的主机计算结构，如图 1-12 所示。

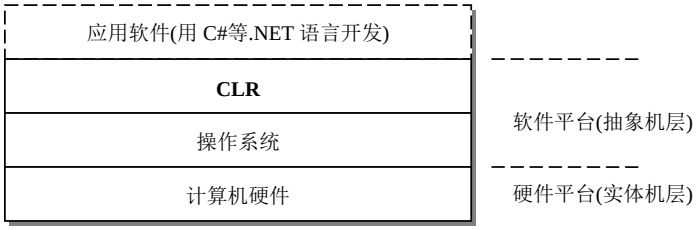

图 1-12　基于 CLR 的主机计算环境

可以看出，由于用.NET 语言开发的应用程序运行在 CLR 上，因此，CLR 相当于操作系统之上的又一层虚拟机。CLR 对程序执行的细节进行了包装，程序员无须关注程序的执行环境，只需专注于程序的业务逻辑和功能流程，从而提高了开发效率。

1.3 C#程序设计语言

20 世纪 80 年代以来，C/C++一直是使用最为广泛的商业化程序设计语言。C/C++具有复杂的底层控制能力，但程序的安全性缺乏保障，且学习周期长，开发效率低。软件业迫切需要一种基于 Web 标准的全新程序设计语言，将底层系统控制和高端应用开发紧密结合起来，在控制力和生产率之间达到良好的平衡，C#语言应时而生。

1.3.1 C#语言的特点

C#语言简介.mp4

C# is a simple, modern, general-purpose, object-oriented programming language.

——摘自 https://www.microsoft.com/net/

C#是一种**简单**、**现代**、**通用**、**面向对象**的程序设计语言。

C#语言的语法与 C/C++、Java 风格类似，支持简单异步模式(simple async patterns)、语言集成查询(language integrated queries，LINQ)、自动内存管理(automatic memory management)等机制，可用于移动(mobile)、Web、云(cloud)、桌面(desktop)、游戏(gaming)、物联网(IoT)等应用软件的开发。随着.NET 技术的普及，C#语言成为.NET 平台最受欢迎的开发语言。

C#从 C/C++发展而来，在继承 C/C++强大功能的同时，汲取了 Java 等多种语言的精华，兼有 Delphi 等 RAD(Rapid Application Development，快速应用开发)语言的高效性，具有语法简洁、面向对象、与 Internet 紧密集成、安全高效、灵活、兼容性好等特点。作为.NET 平台的核心语言，C#能充分享受 CLR 所提供的服务，可方便地与 VB.NET、F#等其他.NET 兼容语言开发的应用程序或组件进行集成和交互。

C#底层控制能力强，高端应用开发快。它就像一把飞刀，不花哨也不滞重，看上去十分简单，反复练习才能运用自如。庸手只能用它削削苹果，高手却能百步穿杨。

C#的学习包括 C#语言本身、FCL 类库两大部分。前者简洁，易学易用；后者庞杂，需反复琢磨，掌握类库的使用规律，持之以恒，就能成为个中高手。

1.3.2 Hello, World

C#程序基本结构.mp4

“Hello, World”是跨入一门新语言的门槛，经典的 C#代码如下：

```
using System;
namespace SayHi
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

这段代码中，只有“Console.WriteLine("Hello, World!");”是程序员自己编写的代码，其他的都可以自动生成，是控制台应用程序项目的代码框架。

Console 表示控制台类，是.NET 框架中现成的类对象。WriteLine 是 Console 类的功能之一，用于在控制台输出信息。"Hello, World!"是字符串常量，放在 WriteLine 的圆括号里即可。运行这个程序，会在屏幕上显示“Hello, World!”。

就是这么简单！这里仅用到了 Console 这个“对象”。

当然，可以稍微花点时间来看看这个控制台应用程序项目的代码框架结构。

(1) using 是 C#的关键字，表示引用其他模块(相当于 C 语言的#include)。System 是名称空间，这是.NET 的 FCL 中的基础类型所在的空间。可以把 System 理解为“程序包”，System 表示包名。要使用这个包里的东西，使用 using 可简化后续程序代码的编写工作，使得代码更加简洁。例如，本例中使用的 Console 类在 System 包中，这里引用后，后面就可以直接使用 Console，否则就得写成“System.Console”。

(2) namespace 是 C#关键字，表示名称空间、名域或包。SayHi 是程序员自己取的名称空间名(一般在创建项目时自动生成)。namespace SayHi{ …} 花括号中的任何内容都属于 SayHi 这个名称空间。如果在别的名称空间中使用 SayHi 中的类，需要用 using 引用，或在类名前加 SayHi.。

名称空间可以是多级结构，理解起来也比较简单。例如，在《红楼梦》第五回中描写到：宝玉见是一个仙姑，喜的忙来作揖，笑问道：“神仙姐姐，不知从那[1]里来，如今要往那[2]里去？我也不知这里是何处，望乞携带携带。”那仙姑道：“吾居离恨天之上灌愁海之中，乃放春山遣香洞太虚幻境警幻仙姑是也。司人间之风情月债，掌尘世之女怨男痴。……”。这位神仙姐姐的名称空间就是：离恨天.灌愁海.放春山.遣香洞.太虚幻境。换句话说，在这个地址可以找到警幻仙姑。

名称空间的级数与实际问题有关。例如，宝玉要找警幻仙姑。如果两人此时都在太虚幻境，直接叫“神仙姐姐”交流即可，不需要限定名称空间(但两人也隐含有名称空间，即太虚幻境)。如果两人不在一起，可以写信交流，信封上必须写清楚警幻仙姑的地址。有了地址，邮递员就能快速地找到收信人。这个地址就相当于 C#的名称空间。

(3) class 是 C#的关键字，表示这是一个类的定义。Program 是类名，可以根据实际需要为类命名。class Program{ …} 花括号中的任何内容都属于 Program 这个类。

(4) static void Main(string[] args){ …}是一个方法(相当于 C 语言的函数)。其中，static 关键字限定这是一个静态方法，表示该方法是唯一的；void 关键字表示这个方法不返回任何数据；Main 是该方法的名称，也是关键字，一个应用程序只能有一个 Main 方法，表示程序运行的入口；圆括号里的“tring[] args”是该方法的参数，args 是参数名，string 表示 args 的数据类型是字符串，[]表示这个参数是一个数组；花括号中的任何内容都属于 Main 这个方法。

1.4 Visual Studio 集成开发环境

微软推出的集成开发环境 Visual Studio .NET 支持对 C#等各种.NET 兼容语言的可视化程序设计，使程序员能够快速构建各类.NET 应用，方便地创建、调试和发布程序。

1.4.1 启动集成开发环境

按如图 1-13 所示顺序找到 Microsoft Visual Studio 2010(或其他版本)。

---

[1] “那”通“哪”。

[2] 同上。

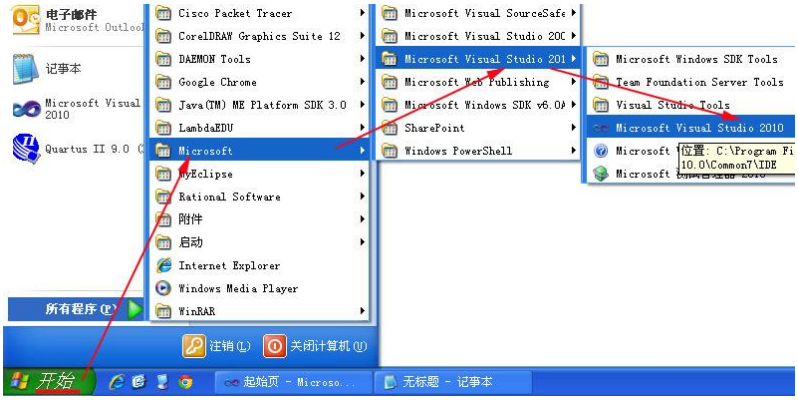

图 1-13　启动 Microsoft Visual Studio 2010

选择 Microsoft Visual Studio 2010 选项，启动集成开发环境，出现如图 1-14 所示的界面，表示启动成功。

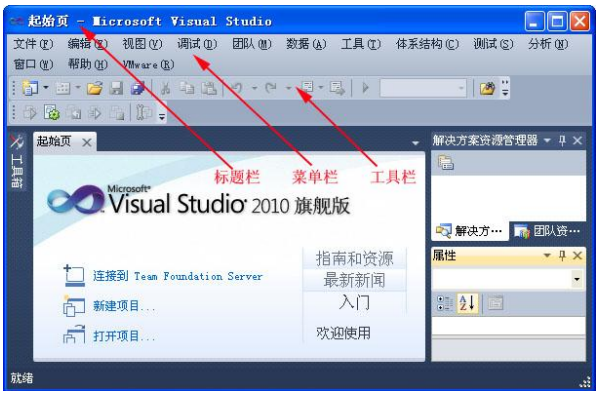

图 1-14　集成开发环境初始界面

图 1-14 中所示的集成开发环境初始界面的格局与常用的 Office 软件大同小异：第一行是标题区，第二行是菜单区，第三行是常用工具按钮区，其余的是工作区。

1.4.2　解决方案与项目类型

程序设计的目的是解决问题。解决问题需要解决方案。问题复杂，解决方案也相应复杂，通常需要分解为若干项目。例如，新建一所学校，建校是一个解决方案，建教学楼、建校舍、建职工住宅等则分属不同的项目。

按如图 1-15 所示的菜单项顺序启动“新建项目”对话框(也可以单击如图 1-14 所示初始界面中的“新建项目”按钮)。

选择“项目”选项后，出现如图 1-16 所示的“新建项目”对话框。

图 1-15　创建项目菜单项

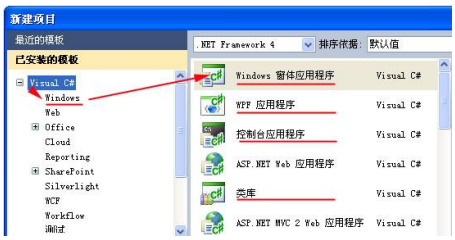

图 1-16　“新建项目”对话框

Visual Studio 集成开发环境为各种类型的项目提供了不同的模板。开发人员可以在这些模板的基础上创建新项目，以节省开发时间。

图 1-16 显示的是 Visual Studio 集成开发环境中预装的模板，左边是模板类别，如 Visual C#、其他语言、测试项目等。模板可以进一步细分，如 Visual C#支持 Windows、Web、Office、Cloud(云)等。当单击左边的模板类型时，右边会显示与所选模板类型对应的项目。例如，图 1-16 右边显示的就是与 Visual C#下的 Windows 对应的控制台应用程序、Windows 窗体应用程序、WPF 应用程序、类库等项目模板。

(1) **控制台应用程序**是指基于 CUI(字符用户界面)进行输入输出的应用程序，适用于初学程序设计语言的基本语法、纯粹的算法研究、一般的科学计算等对用户交互体验要求不高的场合。

(2) **Windows 窗体应用程序**是指基于 GUI(图形用户界面)进行输入输出的应用程序，适用于用菜单、按钮、文本框、列表等控件进行用户交互的场合，也是目前较为常用的应用程序类型，主要应用在管理信息系统、计算机游戏等应用领域。

(3) **WPF** 是 Windows Presentation Foundation 首字母的缩写，即 Windows 表示基础，着重 Windows 表示层的设计。这是基于 Windows 的用户界面框架，提供了统一的程序设计模型、语言和框架，以及新的多媒体交互用户图形界面。它将界面设计师从开发工程师的工作中独立出来，是数据驱动的应用程序的代表，适用于对用户交互体验有较高要求的场合。

(4) **类库**用于生成 DLL 程序，是组件式程序设计的基础。

一般来说，为节省资源和时间，在学习程序设计基础知识、面向对象基本概念时，主要用控制台应用程序模板创建项目并进行实验。而在用户交互体验要求较高的场合，例如桌面应用开发或游戏设计，则使用 Windows 或 WPF 应用程序模板创建项目。当然，对于规模较大的软件项目，应该采用组件技术进行开发，此时就要用类库模板创建项目，以实现各个组件模块。所以，建议学习路线为：控制台应用程序→Windows 应用程序→类库→WPF 应用程序→ASP.NET Web 应用程序。

1.4.3　用控制台应用程序项目实现 HelloWorld

在图 1-16 中，选择 Visual C#→Windows→“控制台应用程序”选项。

在“新建项目”对话框下方的“名称”文本框中输入项目名称，在“位置”下拉列表框中输入项目文件所在目录，如图 1-17 所示。

图 1-17　输入项目名称、位置和解决方案的名称

当勾选图 1-17 中的“为解决方案创建目录”复选框时，“解决方案名称”文本框的内容会随着项目名称的变化而改变。如果不勾选此复选框，可以把新建项目加入已存在的解决方案，此时要在“解决方案名称”文本框中输入已存在的解决方案名。

单击图 1-17 中的“确定”按钮，系统按所选模板创建控制台应用程序项目。创建成功后显示如图 1-18 所示的控制台应用程序设计界面。

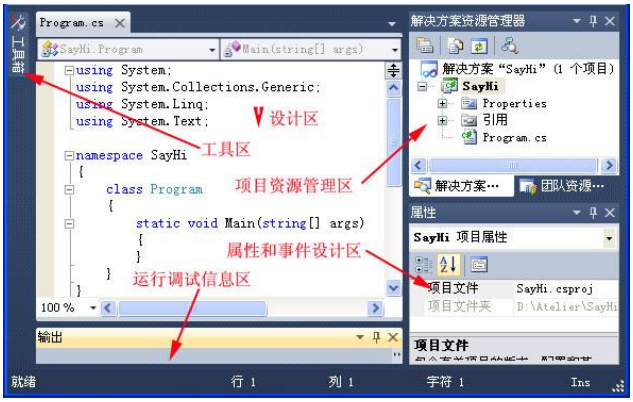

图 1-18　控制台应用程序设计界面

默认的控制台应用程序设计界面由五个部分组成，即设计区、项目资源管理区、运行调试信息区、属性和事件设计区、工具区，用户可以将界面调整成自己喜欢的视图，如设置布局、区域、字体样式等。默认视图是最常用的辅助设计区，有的望文知意，有的则用在特殊场合。例如，放置各种控件和组件的工具区、属性和事件区等主要用于 Windows 应用程序、WPF 应用程序、ASP.NET Web 应用程序等项目的 GUI 界面设计。

项目资源管理区管理的是各种类型的项目文件。该区域类似于 Windows 的资源管理器，呈树型结构。最上层是解决方案的名称，显示该解决方案包含了几个项目。解决方案的下层是项目名称。这个例子中，两者的名字一样，都是 SayHi。项目名称的下层是应用程序的特性(Properties，为避免与其下方区域中的“属性”混淆，以“特性”译之。当然最好是不译，就用 Properties，表示应用程序自身特有的性质)、引用的其他资源、源程序文件名等。这个例子中，默认(自动生成的)程序文件是 Program.cs。cs 是 C#，即 C Sharp 的缩写，表示 Program 是一个 C#源程序文件。

控制台应用程序的设计区就是用于编辑源程序代码的区域。图 1-17 中看到的是集成开发环境按控制台应用模板自动生成的框架代码，在这里可编辑源程序代码。

在图 1-18 中的框架代码的 Main 方法中添加语句，如图 1-19 所示。

在输入 Console 的过程中，集成开发环境会弹出相关列表供选择。熟练掌握这个技巧可以节省编码时间。在图 1-19 中弹出的列表中选择 WriteLine 或 Write，紧跟其后输入一对圆括号，在圆括号内输入“Hello, World!”，在圆括号后面输入分号结束该语句，这次设计工作就完成了，如图 1-20 所示。

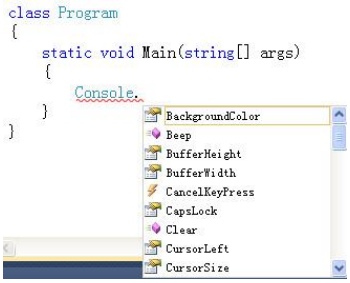

图 1-19　编码

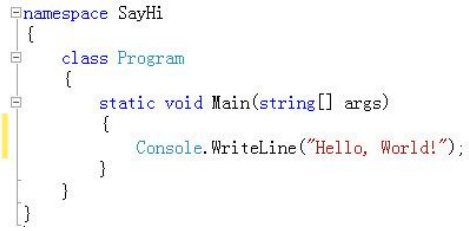

图 1-20　HelloWorld 代码

当然，这只是完成了编写源程序代码的工作。为使得该程序能够运行，还需要对其进行编译和链接。也就是把它编译为二进制代码并与引用的其他资源链接在一起。这个过程可以利用菜单栏的“生成”→“生成解决方案”命令来完成。如果源程序代码没有任何问题，在运行调试信息区会显示“0 个错误”“生成成功”等信息。

为简化操作，选择菜单栏的“调试”选项，弹出如图 1-21 所示用于调试程序的子菜单。其中，“启动调试”和“开始执行(不调试)”包含了程序编译、链接、运行等过程。两者的区别是：对于没有输入的控制台应用程序，选择前者，运行窗口会一闪而过，没法看清结果；选择后者，程序执行完毕后会暂停，按任意键后才关闭运行窗口。

选择“开始执行(不调试)”选项，程序开始运行，结果如图 1-22 所示。

这就是基于 CUI 的显示结果，实现了 Hello World 功能。

图 1-21　“调试”菜单项

图 1-22　控制台应用程序运行结果

1.4.4　用 Windows 窗体应用程序项目实现 HelloWorld

重新创建项目，再次出现图 1-16 所示界面。这次选择 Visual C#、Windows 以及“Windows 窗体应用程序”，输入项目名称，如 HelloWorld，单击“确定”按钮，系统按所选模板创建 Windows 窗体应用程序项目。创建成功后，显示如图 1-23 所示的 Windows 窗体应用程序设计界面。

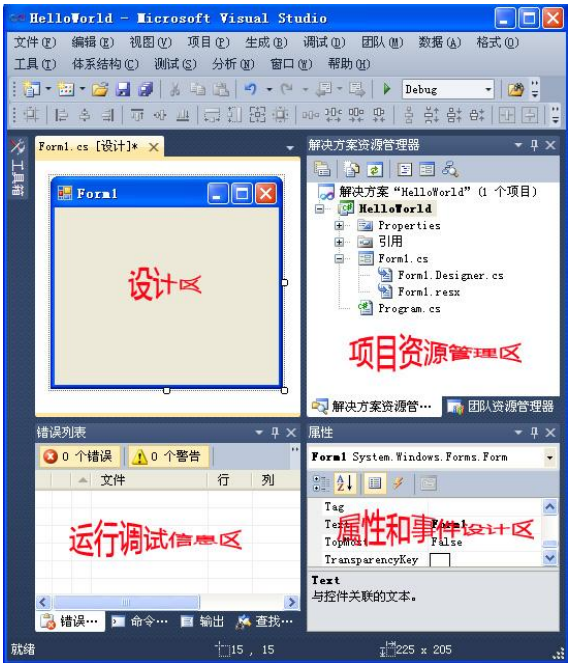

图 1-23　Windows 窗体应用程序设计界面

图 1-23 所示格局与控制台应用程序设计界面相同，但内容更加丰富。

1. 设计区

与控制台应用程序的设计区一样，这里的设计区也是进行 Windows 窗体应用程序设计时最常用的区域之一。默认显示的是一个待设计的窗口界面，称为窗体，其名称 Form1 是自动生成的。右击窗体空白区域，弹出如图 1-24 所示的菜单列表。

选择“查看代码”选项，进入如图 1-25 所示的窗体源程序代码编辑界面。

图 1-25 所示界面与控制台应用程序设计区一样，可以在其中编辑源程序代码。现在看到的是集成开发环境按 Windows 窗体应用程序模板自动生成的框架代码。

2. 项目资源管理区

除了 Program.cs 文件外，多了一个 Form1.cs 文件。该文件下面列有 Form1.Designer.cs 和 Form1.resx 两个文件，前者是集成开发框架的设计器自动生成的窗体界面代码文件，后者是相关的资源文件。这两个文件一般不用手工修改。这个区域的项目资源的名称可以改变，包括自动生成的窗体文件名。右击资源的名字，在弹出的菜单中选择“重命名”选项即可编辑新的资源名。当然，如果源程序代码中直接使用了资源名，就不要随意变动资源的名称。

图 1-24　右击窗体空白区域弹出菜单

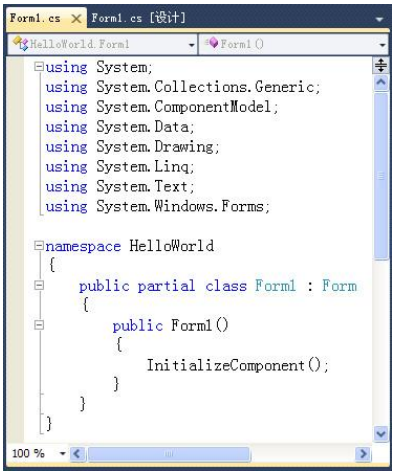

图 1-25　窗体源程序代码编辑界面

3. 工具箱

这里放置的是.NET 框架预定义的控件和组件，用于提高软件开发的效率和质量。在界面设计时，把光标移到工具箱区域，会弹出如图 1-26 所示的界面。

窗体相当于一个容器，可以从工具箱拖动所需控件到窗体进行界面设计。

为实现 HelloWord 功能，可以用工具箱里的 Label 控件来实现。这是.NET 框架自定义的控件，称为标签控件，一般用于在窗体上显示提示信息或不需要修改的数据。

从图 1-26 中拖动 Label 到窗体，如图 1-27 所示。

图 1-26　工具箱中的控件和组件

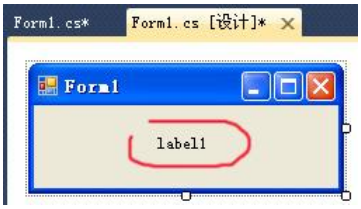

图 1-27　拖动到窗体的 Label 控件

4. 属性和事件设计区

在这个区域可以设置、编辑窗体、控件、组件等对象的属性和事件。单击窗体中的 Label 控件，其属性和事件信息如图 1-28 所示。

其中，属性指的是对象的数据值，如名称、背景色等；事件指的是发生在对象身上的事情，如在对象上单击或双击鼠标等。在图 1-28 中，单击属性按钮和事件按钮可以切换到属性列表和事件列表。列表的左边是属性或事件的名称，右边是该名称对应的取值。

属性值可以修改。例如，图 1-28 显示的是拖到窗体中 Label 控件的属性，其 Name 属性的默认值是 label1，把它改为“hi”。该 Label 的 Text 默认值也是 label1，删除其默认值，把它改为空，可以看到窗体显示的“lable1”消失了。

单击窗体的空白区域，就会在属性和事件设计区显示该窗体的信息。单击图 1-28 中的事件按钮，显示窗体的事件信息，如图 1-29 所示。

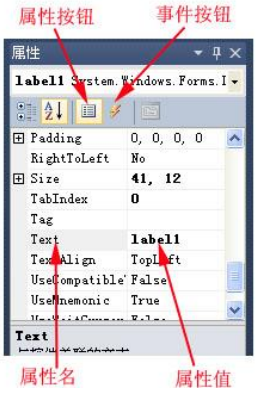

图 1-28　Label 的属性信息

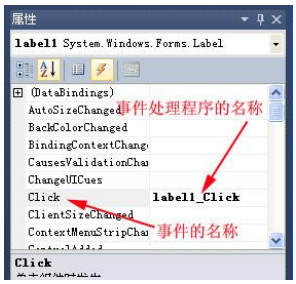

图 1-29　窗体的事件信息

图 1-29 中与事件名对应的是该事件的处理程序，即一旦事件发生，会执行这段程序代码来处理这个事件。例如，Load 是窗体的载入事件。窗体创建后并不会立即显示在屏幕上，其中有一个载入的过程。换句话说，在窗体显示出来之前，可以在其载入过程中编写载入事件处理程序，做一些控件的初始化工作。双击事件名右边的事件处理程序名输入框，可以自动生成该事件的处理程序代码框架。例如，双击 Load 输入框，会自动生成 Load 事件的处理程序代码框架并显示在代码编辑区，如图 1-30 所示。

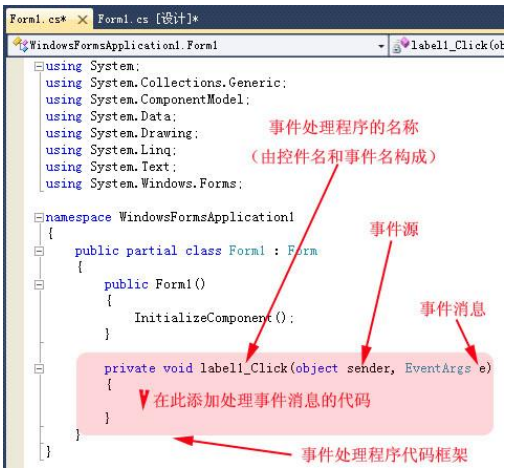

图 1-30　窗体载入事件处理程序框架代码

自动生成的事件处理程序代码由处理声明和处理体两部分组成。其中，处理声明由 4 个部分组成：private 是关键字，表示私有，即这段程序只能在 Form1 这个类中使用；接着是事件处理后返回信息的类型，void 表示不返回任何信息；Form1_Load 是程序名称，由处理事件的对象的名字和事件本身的名字自动组合而成；sender 表示事件发生的源头；e 相当于信息包裹，携带有发生事件的消息。

处理体目前只是一对花括号，表示不处理任何事情。开发人员可以在其中添加处理事件的程序代码。例如，在花括号中添加语句：

```
hi.Text = "Hello, World";
```

语句中，hi 是窗体中的 Label 对象的名字(刚才给它取的名字)，Text 是其属性。给 hi 对象的 Text 属性赋值，运行时，在窗体中显示的就是“Hello, World”。

至此，用 Windows 应用程序显示“Hello, World”的设计工作就完成了，如图 1-31 所示。

依次选择“调试”→“启动调试”选项，程序开始运行，运行结果如图 1-32 所示。

```
namespace HelloWorld
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            hi.Text = "Hello, World";
        }
    }
}
```

图 1-31　窗体载入事件处理代码

图 1-32　Windows 应用程序运行结果

这就是基于 GUI 的显示结果，实现了显示“Hello, World”功能。

实现 Hello World 的过程说起来较长，但真正实现起来很快。熟悉了这个过程，再次实现这个功能，只需很短的时间即可完成任务。对于 C#初学者来说，这个过程涉及了许多不曾见过的术语，在后续章节中会一一展现。

习　题　1

1. 浏览网页 https://www.microsoft.com/net/，对 C#有个总体的了解。
2. 查找相关资料，安装 Microsoft Visual Studio 集成开发环境到自己的计算机上。
3. 按 1.4 节流程进行上机实践，熟悉 Visual Studio 集成开发环境。