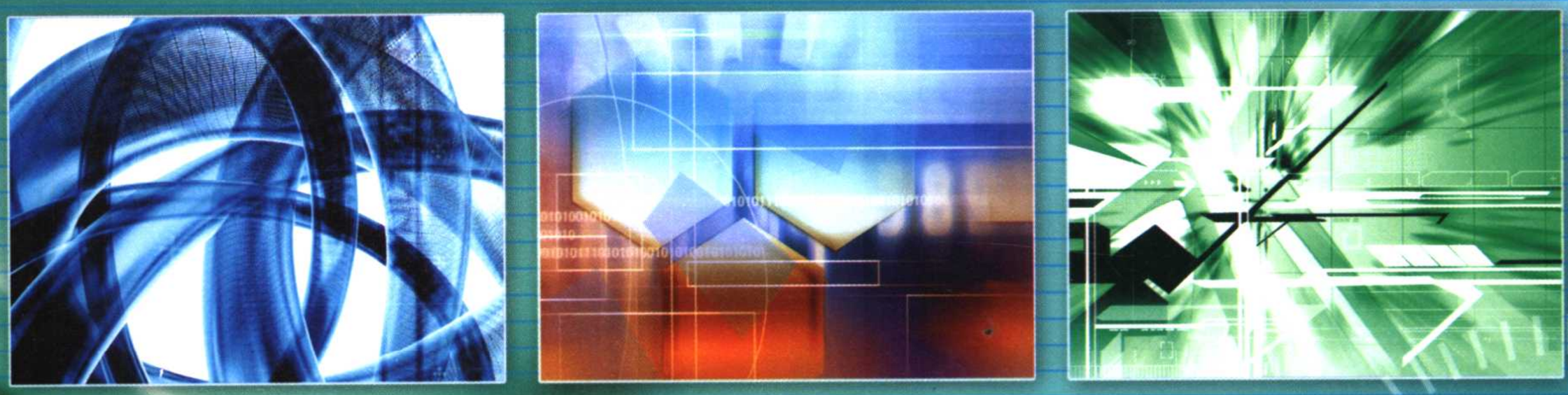
高等院校计算机技术系列教材

Visual C++.NET 程序设计

段 超 施平安 任卓谊 编著



冶金工业出版社

高等院校计算机技术系列教材

Visual C++.NET 程序设计

段　超　施平安　任卓谊　编著

北　　京

冶 金 工 业 出 版 社

内 容 简 介

本书是根据普通高等教育“十一五”国家级规划教材的指导精神而编写的。

.NET 是微软为适应 21 世纪软件发展而推出的一项重要战略。.NET 不仅完全面向对象、简单高效、支持多线程，而且更重要的是，它独立于任何特定的语言或平台。

本书较为全面地介绍了其中综合性最高、结构最复杂的软件开发工具：Visual C++.NET。全书共分为 18 章，内容包括：Visual C++的基本知识、面向对象编程的特性、.NET 框架的使用、数据访问、Web 服务的创建、Visual C++.NET 高级特性的其他应用等。本书在介绍各个主题时，不仅提供了大量的示例、表格、插图，而且还在各章后面安排了一定量的练习，以供读者在复习时使用。

本书可以作为计算机及相关专业的本科教材，也可作为从事 Visual C++.NET 程序开发人员的参考书。

图书在版编目（CIP）数据

Visual C++.NET 程序设计 / 段超等编著. —北京：冶金工业出版社，2005.12

ISBN 7-5024-3878-5

I. V… II. 段… III. C 语言—程序设计—高等学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字（2005）第 139383 号

出版人 曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）

责任编辑 程志宏

佛山市新粤中印刷有限公司印刷；冶金工业出版社发行；各地新华书店经销

2006 年 1 月第 1 版第 1 次印刷

787mm × 1092mm 1/16； 22.25 印张； 655 千字； 346 页

33.00 元

冶金工业出版社发行部 电话：（010）64044283 **传真：**（010）64027893

冶金书店 地址：北京东四西大街 46 号（100711） **电话：**（010）65289081

（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于本书

本书是根据普通高等教育“十一五”国家级规划教材的指导精神而编写的。

目前，全国各地本科院校普遍扩招，本科生人数迅速增长，这给他们的就业带来了巨大的压力。而当前本科生的就业情况还不如专科学生，究其原因，所用教材与实际应用脱轨是一主要因素。针对现有教材质量较差、品种单一、版本陈旧、实用性和可操作性不强等原因，肩负着应用型人才培养的高等本科院校急需一系列符合当前教学改革需要的教材。

.NET 的推出，受到了业界的广泛关注。本书为了使大家能够更好地学习掌握有关.NET 编程方面的知识而编写，以帮助那些有志投身于软件业的学生们或有志青年。

二、本书结构

本书共分两大部分，第一部分（第 1 章到第 7 章）讲授 Visual C++.NET 程序设计的基础知识，主要包括 Visual C++.NET 集成开发环境、语言特性、托管和程序集等；第二部分（第 8 章到第 18 章）讲授 Visual C++.NET 程序设计的中级知识，主要包括面向对象编程、委托和事件、异常、图形用户界面设计、多线程、XML、数据库访问和 Web 服务等。具体结构如下：

第 1 章：概述。主要介绍.NET 框架的作用及其在软件发展中的地位，简要地介绍.NET 框架的一些新概念，并使大家初步地了解.NET 框架类库。

第 2 章：变量与运算符。主要介绍变量声明、变量赋值；如何使用基本数据类型；变量类型间的转换；如何为类添加成员变量；指针、结构、枚举和常量等其他变量类型；算术运算符、关系运算符和逻辑运算符的用法；按位运算符、sizeof 运算符和其他运算符的用法；如何进行类型转换；运算符的优先级次序和结合方向。

第 3 章：控制结构。主要介绍各种循环结构和选择结构的使用方法；break 语句和 continue 语句。

第 4 章：函数。主要介绍如何使用函数来模块化地构造程序；如何声明函数原型，定义函数体；作用域规则；函数之间传递信息的机制；递归的概念，以及如何区分递归和迭代；定义和使用重载函数。

第 5 章：数组。主要介绍 C++本地数组和托管数组的概念；声明、初始化数组，并能够引用数组中的元素；如何将数组传递给函数；基本的排序和查找技术；声明和操作多维数组。

第 6 章：托管编程入门。主要介绍语言的互操作性、垃圾收集机制、托管类型与值类型、装箱操作的概念、属性的概念和如何定义属性。

第 7 章：程序集。主要介绍程序集的概念及其主要优点；如何使用 ILDasm.exe 工具查看程序集；如何区分私有程序集和使用共享程序集；如何创建、安装和使用共享程序集。

第 8 章：类和对象。主要介绍类的相关概念；成员访问限定符的作用；如何使用属性来保持对象的一致性；this 指针的概念和用法。

第 9 章：面向对象编程。主要介绍继承和多态的概念；基类和派生类的概念和相互关系；protected 等成员访问限定符；基类和派生类中使用构造函数和析构函数；抽象类和具体类的概念及其区别。

第 10 章：委托和事件。主要介绍委托的概念；如何创建和使用委托；事件的概念；如何创建和使用事件。

第 11 章：异常处理。主要介绍异常和错误处理；如何抛出、捕获和处理异常；try 块、catch 块和 finally 块的作用和用法；如何创建自定义异常类型。

第 12 章：图形输出。主要介绍设备环境和图形对象；如何操作颜色和字体；绘制线条、矩形、字符串、图像等基本图形；利用 Image 类操作和显示图像；利用 Graphics 绘制较为复杂的图形。

第 13 章：图形用户界面。主要介绍如何建立图形用户界面；窗体的概念；如何创建和使用一些基本的窗体控件；如何处理键盘事件和鼠标事件；创建和使用菜单、工具栏、状态栏、对话框，以及如何根据需要创建和使用对话框。

第 14 章：文件处理。主要介绍文件的创建、读写和更新操作；System::IO 命名空间；文本的输入/输出；二进制数据的输入/输出。

第 15 章：多线程。主要介绍多线程的概念及其工作原理；如何创建、管理和销毁线程；什么是线程的生命周期；线程同步；线程优先级和线程调度。

第 16 章：XML。主要介绍 XML 的基本组成；XML 命名空间；如何创建架构；如何读写 XML 文档；如何使用 XSL 转换 XML 文档。

第 17 章：ADO.NET。主要介绍关系数据库模型；如何使用 SQL（结构化查询语言）进行数据查询；System::Data 命名空间中操纵数据库的接口和类；ADO.NET 对象模型；System::Data::OleDB 命名空间常用类型的使用。

第 18 章：Web 服务。主要介绍 Web 服务的概念；如何创建 Web 服务和 Web 服务客户端；ATL Server 的概念；如何使用 ATL Server 创建基于 Web 的应用程序；如何使用 ATL Server 创建 Web 服务。

三、本书特点

本书系统、全面地研究和借鉴了国外相关教材先进的教学方法，结合国内院校教学实际和先进的教学成果，根据教育部“十一五”国家级规划教材应用型本科教育的指导思想编写，具有实用性和可操作性，与时俱进，与当前就业市场结合得更加紧密。

本书内容丰富、条理清晰、难易适中，以循序渐进、深入浅出的方式引导读者逐步掌握.NET 程序设计精髓。本书注意理论与编程实践的结合，通过各种不同层次的例子和完整的应用程序使读者加深对.NET 基本理论和思想的理解。

四、本书适用对象

本书可以作为计算机及相关专业的本科教材，也可作为从事 Visual C++.NET 程序开发人员的参考书。

我们想借这个难得的机会感谢教研室的全体同仁。每当遇到困惑与他们进行探讨时，他们总是无私地将自己的经验、知识和设想提供给我们，和他们在一起讨论.NET 问题使我们受益匪浅。此外，还要感谢出版社的同志们，正是他们严谨的作风和细致的工作，为本书提出了大量的宝贵意见，使得我们能够更加精益求精地完成这本书。

尽管我们已经倾心相注，精心而为，但难免存在疏忽纰漏，恳请广大读者不吝指教和改正，我们会全力改进，以期在后续工作中得以完善，联系方法如下：

电子邮箱：service@cnbook.net

网址：www.cnbook.net

本书的电子教案、源代码及习题参考答案可在该网站免费下载。此外，该网站还有一些其他相关书籍的介绍，可以方便读者选购参考。

编　者

2005 年 9 月于广州

目　录

第1章 概 述

学习目标

- 知道.NET 框架的作用及其地位
- 初步接触.NET 框架类库

2000 年 6 月，微软公司宣布了.NET 战略。.NET 平台对早期的开发平台进行了显著地改进，.NET 框架采用 CLR，使得采用.NET 平台所支持的不同编程语言开发的应用程序能够相互通信。

微软公司的.NET 战略为我们展示了一个全新的境界，它极力倡导在软件开发过程中，尽情地享受 Internet 和 Web 服务。.NET 的一个关键特性在于，它独立于任何特定的语言或平台。.NET 并不要求程序员使用某一种或某几种特定的程序语言；相反，开发人员可以使用任何语言来创建一个.NET 应用程序，只要这些语言与.NET 平台相兼容即可。这样，多个程序员在使用.NET 框架开发同一个软件项目时，可以采用自己最为精通的.NET 语言来编写代码。

.NET 的一个关键部分是 Web 服务。Web 服务实际上是一种应用模式，用于通过 Internet 向客户提供功能。客户或其他应用程序可以将 Web 服务的实例用作可重用的软件构建单元。Web 服务扩展了软件重用的概念，它允许程序员在开发 Web 应用程序时将主要精力集中在自己擅长的领域，而不必亲自实现应用程序的每个部分。软件公司可以购买网上发布的 Web 服务，并将时间和精力花在开发自己的产品上。程序员可以使用数据库、安全性、身份验证、数据存储等领域的 Web 服务来创建一个应用程序，而不必知道这些软件组件的内部实现细节。

XML 和 SOAP 是实现 Web 服务应用通信的保证。XML 将数据进行半结构化，SOAP 是 Web 服务在实际应用通信时所用的协议。由于 XML 和 SOAP 是行业标准而非微软的私有标准，因此使用 Web 服务生成的应用程序越来越流行。

本章将讨论.NET 主要组成部分。

1.1 .NET 框架

.NET 框架是构建并运行应用程序的平台，它主要由公共语言运行库（CLR）和.NET 框架类库（FCL）组成。CLR 抽象了操作系统服务，并用作托管应用程序的执行引擎。FCL 为开发各种应用程序提供了一套可重用的类。

在编写.NET 应用程序时，一般只需使用 FCL。当然，也可以调用 Windows API 或 COM 对象，但是这样必须使用非托管代码（不需要 CLR 支持的本机代码），这种转换会降低系统性能。

.NET 主要是生成、部署和运行 Web 服务的平台，但.NET 框架同样支持其他的编程模型。除了 Web 服务外，还可以利用它来编写控制台应用程序、GUI 应用程序、Web 应用程序，甚至是 Windows 服务。

以下的几个小节将讨论并介绍.NET 框架的有关重要概念。

1.1.1 公共语言运行库（CLR）

CLR 是.NET 框架的心脏。开发人员编写的每段框架代码，不是运行在 CLR 中，就是得到 CLR 的许可后在其外部运行，如果没有 CLR 的支持，就什么也不能运行。

CLR 位于操作系统之上，并为宿主托管应用程序提供虚拟环境。当运行托管可执行文件时，CLR 加载包含可执行文件的模块，并运行其中的代码。为 CLR 编写的代码称为托管代码。为.NET 进行编

译实际上是为 CLR 进行编译，因为编译得到的代码并不是 CPU 所能识别的机器指令，而是一种称为“微软中间语言（MSIL，简称为 IL）”的新语言。MSIL 为一个虚拟的 CPU 定义了一套处理指令，已编译为 MSIL 的代码必须进一步编译为本机机器指令后才能在 CPU 上运行。

CLR 提供了一个实时编译器，用来将 IL 代码编译为本机机器代码。编译后的代码就会进行缓存，在程序中断时，这些代码就可能会从系统中删除。整个程序不会一次性地进行全部编译，而是将它们分成小的程序段，在需要的时候进行单独地编译。一个正在运行的程序在其生命周期内也许不会使用到所有代码，就像在一次运行中，用户不会用到应用程序的所有功能一样。

由于实时编译器将 MSIL 指令转换为本机代码，因此它扮演了代码验证的角色，可以确保代码是类型安全的。实际上，不可能执行超出它存取访问的指令。在托管程序中，永远不会遇到无效指针的问题，因为 CLR 会在无效指针被使用之前抛出一个异常。不能将类型转化为它不属于的东西，因为这样的操作不是类型安全的。除了消除一些困扰应用程序的最常见的错误以外，代码验证机制使得运行企图破坏主机系统的恶意代码变得非常困难。

代码验证机制也是使 CLR 能够在单个进程中宿主多个应用程序的主要技术，它将程序进程划分为虚拟区域、应用程序域。Windows 通过将程序宿主在不同的进程中来隔离它们。而这种在每个进程中宿主一个应用程序的模型会消耗较大的内存。当一个服务器应用程序需要同时伺服上千个用户时，内存消耗的问题就非同小可。

在某些情况下，CLR 不会为每个应用程序启动一个新进程，而是启动一个或多个进程，在应用程序域中宿主各个应用程序。应用程序域就像进程一样安全，因为它们形成了托管应用程序不能跨越的边界。而且，应用程序域的效率要高于进程，因为一个进程可以宿主多个应用程序域，并可以将库文件加载到应用程序域中，由其中的应用程序共享。

C/C++程序员经常会因为内存泄漏而感到苦恼。内存泄漏是由于没有及时将不再使用的资源返回给系统而造成的。CLR 为这一问题提供了解决方案：垃圾收集机制。垃圾收集机制负责跟踪有效的引用，并对引用进行计数。它会定时对所有的对象引用进行检测，如果检测到当前未被引用的对象，CLR 就会将其从内存中释放。在垃圾收集器完成了释放对象所占资源的工作后，它会用压缩托管堆的方式来整理垃圾，从而可以做到为新的请求分配连续的内存空间。

由于有了垃圾收集器，只由托管代码组成的应用程序不会引起内存泄漏。

1.1.2 托管模块

当使用能够生成 MSIL 的编译器生成程序时，编译器会产生一个托管模块。托管模块实际是一个需要 CLR 才能执行的标准 Windows 可移植可执行文件。托管模块的扩展名通常为 EXE、DLL 或 NETMODULE。

在托管模块中有以下四个重要组成部分：

（1）Windows 可移植文件表头（PE 文件表头）。

（2）包含标识托管模块信息的 CLR 头。

（3）描述模块内容及其外部依赖项的元数据。

（4）由源文件生成的 MSIL 指令。

每个托管模块均包含描述模块内容的元数据。元数据是必需的，每个与 CLR 兼容的编译器都必须生成元数据，这意味着每个托管模块都是自我描述的。由于有了元数据，CLR 就能判断出它加载的每个托管模块中包含哪些类型。借助于元数据，在程序编辑器中输入类实例的名称时，Visual Studio.NET 集成开发环境会显示一个与上下文相关的可用方法和属性列表，我们将这种特性称为智能感应。

1.1.3 元数据

元数据是一种二进制信息，用以对存储在公共语言运行库可移植可执行文件（PE 文件）或存储在

内存中的程序进行描述。将用户的代码编译为 PE 文件时，便会将元数据插入到该文件的一部分中，而将代码转换为 Microsoft 中间语言（MSIL）并将其插入到该文件的另一部分中。在模块或程序集中定义和引用的每个类型和成员都将在元数据中进行说明。当执行代码时，运行库将元数据加载到内存中，并引用它来发现有关代码的类、成员、继承等信息。

元数据以非特定语言的方式描述在代码中定义的每一类型和成员。元数据存储以下信息：

（1）程序集的说明。

标识（名称、版本、区域性、公钥）。

导出的类型。

该程序集所依赖的其他程序集。

运行所需的安全权限。

（2）类型的说明。

名称、可见性、基类和实现的接口。

成员（方法、字段、属性、事件、嵌套的类型）。

（3）属性。

修饰类型和成员的其他说明性元素。

对于一种更简单的编程模型来说，元数据是关键。该模型不再需要接口定义语言（IDL）文件、头文件或任何外部组件引用方法。

元数据允许.NET 语言自动以非特定语言的方式对其自身进行描述，而这是开发人员和用户都无法看见的。另外，通过使用属性，可以对元数据进行扩展。

1.1.4 MSIL 指令

MSIL 为处理器定义了本机指令集。编写.NET 框架程序无需通晓 MSIL，就像编写 Windows 程序无需精通 x86 汇编语言一样。但是，如果掌握一些 MSIL 的基本知识，当.NET 框架类库中的某个方法没有如期运行而又想知道为什么时，这些知识将对你有很大的帮助。

MSIL 总共包括 100 多个不同的指令。一些是芯片指令集中的典型低级指令，诸如将两个值相加的 ADD 指令；另一些较高级的指令是硬件指令集中的典型指令，诸如 THROW 抛出异常。

MSIL 使用基于堆栈的执行模型。X86 处理器将数值加载到寄存器之后对它们进行操作，而 CLR 将数值加载到计算堆栈中进行操作。要将两个值相加，CLR 将它们复制到堆栈中，调用 ADD，然后从堆栈中获得结果。将数值从内存复制到堆栈的过程称为加载，从堆栈复制到内存的过程称为存储。

表 1-1 列出了大部分常见的 MSIL 指令，并进行了简要的说明。

表 1-1 常用的 MSIL 指令

指令	说明
BOX	将值类型转换成引用类型
CALL	调用方法；如果方法是一个虚方法，则忽略其虚拟性
CALLVIRT	调用方法；如果方法是一个虚方法，则不忽略其虚拟性
CASTCLASS	将对象转换为另一种类型
LDC	向堆栈加载数值常量
LDARG[A]	向堆栈加载参数或参数地址
LDELEM	向堆栈加载数组元素
LDLOC[A]	向堆栈加载局部变量或局部变量地址
LDSTR	向堆栈加载字符串
NEWARR	创建新数组
NEWOBJ	创建新对象

续表 1-1

指令	说明
RET	从方法调用中返回
STARG	将堆栈中的值复制到参数中
STELEM	将堆栈中的值复制到数组元素中
STLOC	将堆栈中的值传递给局部变量
THROW	抛出异常
UNBOX	将引用类型转换为值类型

1.2 .NET 框架类库（FCL）

使用 C 语言的 Windows 程序员偏爱于使用 Windows API 和第 3 方 DLL 提供的函数，C++程序员倾向于使用他们自己创建的类库或 MFC，VB 程序员则更愿意使用 Visual Basic API，它是对底层操作系统的抽象。

现在有了.NET 框架类库，这意味着你可以忘记过去的一切。FCL 包含多达 7000 个类型（类、结构、接口、枚举和委托）。有些 FCL 类型包含 100 多个方法、属性和其他成员，因而学习 FCL 并非一件容易的事情。但是，大家应该看到，所有的.NET 语言都使用相同的 API：FCL，这就是巨大的进步，从这个角度来看，掌握 FCL 丝毫不会浪费你的宝贵时间。

为了更快地学习和掌握 FCL，微软将 FCL 划分为一个树状的命名空间层次结构。FCL 一共大约有 100 多个命名空间。每个命名空间包含类和其他可完成同类任务的类型。例如，System::IO 命名空间包含文件输入/输出的类，System::Windows::Forms 命名空间包含创建和管理各种 GUI 控件的类。大家可以在.NET 框架 SDK 在线帮助文档中找到 FCL 中所有命名空间的完整列表。

表 1-2 列出了几个常见的 FCL 命名空间。

表 1-2 部分 FCL 命名空间

命名空间	描述
System	核心数据类型和辅助类
System::Data	ADO.NET 数据访问类
System::Collections	哈希表、可调整大小的数组和其他容器
System::Drawing	GDI+图形类
System::IO	执行文件和流的输入/输出的类
System::Net	支持网络通信的类
System::Web	提供 ASP.NET 和 Web 窗体支持的核心结构
System::Web::Services	编写 Web 服务的类
System::Xml	提供对 XML 的支持的类
System::Windows::Forms	用于创建 Windows 窗体应用程序的类
System::Threading	创建和管理线程的类
System::Security	提供有关 CLR 安全性的类
System::Security::Cryptography	提供加密服务的类
System::Web::UI	ASP.NET 的核心类
System::Web::UI::WebControls	ASP.NET 服务器控件

很明显，试图通过一章（甚至是一整本书）就能涵盖 FCL 所有内容的想法是幼稚的。通过本书的学习，大家将会熟悉许多 FCL 类库中的命名空间，但这是远远不够的。只有在不断应用中才能全面地掌握 FCL。

1.3 例子

下面这个例子也非常简单，它不仅在窗体（Form）中显示字符串“欢迎进入.NET 世界!”，还将显示一幅图像作为背景。

（1）创建一个名为 Chapter01Sample1 的 C++控制台应用程序。因为该例子将要使用窗体，并进行绘图操作，所以必须引入相应的 dll 文件。在 Chapter01Sample1.cpp 文件中“#using <mscorlib.dll>”语句的后面添加如下代码：

```
#using <System.dll>
#using <System.Windows.Forms.dll>
#using <System.Drawing.dll>
```

然后，在“using namespace System;”语句的后面添加如下代码：

```
using namespace System::ComponentModel;
using namespace System::Windows::Forms;
using namespace System::Drawing;
```

以使程序能够使用这些命名空间中的类。System::Windows::Forms 命名空间包含着创建窗体所要用到的类，而 System::ComponentModel 命名空间提供了托管类，用来管理组件和控件的设计时和运行时行为，在处理窗体时会经常用到它。System::Drawing 命名空间封装了基本的 GDI+绘图功能，提供了简单的面向像素的二维图形操作功能。

（2）创建一个新类 Sample01，并使其派生于 Form 类。该类的完整代码如下所示：

```
__gc public class Sample01 : public Form{
private:
    System::Drawing::Font *pFont;              //字体
    System::Drawing::Bitmap *pBmp;             //位图
public:
    Sample01()
    {
        //设置窗体属性
        Text = S"Chapter01 Sample";
        ClientSize = System::Drawing::Size(400,300);
        FormBorderStyle = FormBorderStyle::Fixed3D;

        pFont= new System::Drawing::Font("宋体",20,FontStyle::Regular,GraphicsUnit::Pixel);
        pBmp = new System::Drawing::Bitmap(S"Sample01.jpg");
        //订阅 Paint 事件
        Paint += new PaintEventHandler(this,&Sample01::Form_Paint);
    }

    void Form_Paint(Object* pSender,PaintEventArgs* pe)
    {
        //窗体 Paint 事件的处理函数
        Graphics *graphics = pe->Graphics;
        graphics->DrawImage(pBmp,0,0);
        graphics->DrawString(S"欢迎进入.NET 世界! ",pFont,Brushes::Red,100,10);
    }
};
```

下面我们将详细地分析一下这些代码的作用。

关键字__gc 是 C++的一个托管扩展，它简化了与.NET 框架组件的交互，通过在关键字 class 前放置__gc，将使这个类成为一个托管类。这样，该类的实例只能在托管堆上进行创建，而不能在堆栈上进行创建，否则，编译器将会发出错误（C3149）。在实例化托管类后，.NET 运行环境将管理该类对象的生命周期。在对象超出其生命周期时，它所占用的内存将由垃圾自动收集器自动收回。可以使用如下语法来实例化托管类：

```
Sample01 sample = __gc new sample();
```

去掉__gc 修饰符也是可以的，这时编译器仍然使用托管的 new 操作符来进行实例化，因为所创建的 Sample01 类已经声明为一个托管类。

Sample01 类的父类 Form 类共有 39 个属性，本例子只设置了其中的 3 个：Text、ClientSize 和 FormBorderStyle。Text 属性是指显示在窗体标题栏中的标题文本；FormBorderStyle 属性指定窗体边界风格，从 FormBorderStyle 枚举中取值；ClientSize 属性指定窗体的客户区大小（该区域是指除标题栏和边界之外的窗体其他部分）。有兴趣的读者，可以查阅联机文档，以获得更详细的信息。

Sample01 类声明两个实例变量。Font 类型的 pFont 和 Bitmap 类型的 pBmp，分别用于显示文本和图像。Font 类表示字体，它是 System::Drawing 命名空间的一部分。在本例调用的 Font 构造函数中，第 1 个参数指定字体的名称，第 2 个参数指定字体的大小，最后一个参数表示度量字体大小的单位，本例使用像素来度量字体的大小。Bitmap 类表示位图，该类可以处理几种常见的图像格式，诸如.bmp、.jpg 和.gif。本例创建一个 Bitmap 对象 pBmp，并从 Sample01.jpg 文件中读取数据。

为了使程序在运行时能够一直显示文本和图像，我们必须处理 Paint 事件。窗体的状态在发生变化时，操作系统都会向它发送一个 Paint 事件，以刷新窗体。由于 Windows 只是刷新窗体上的组件，诸如窗体本身、标题栏、最小化按钮等等，所以窗体客户区中所绘图形的刷新工作需要大家自己来完成。下面的代码是添加 Paint 事件的处理函数：

```
Paint += new PaintEventHandler(this,&Sample01::Form_Paint);
```

这行代码尽管非常简单明了，但是它却隐含着.NET 中非常重要的概念：委托和事件。现在大家只需知道，这行代码的作用是将 Sample01 类的 Form_Paint 方法作为该类 Paint 事件的处理函数。至于.NET 的事件机制，本书将在第 10 章予以详细介绍。

注意：在进行绘图操作之前，必须获取设备环境（device context）的一个实例，即 Graphics 对象。Form_Paint 函数通过调用 DrawImage 和 DrawString 方法来显示图形和文本。

（3）在_tmain 函数中添加以下代码：

```
Console::WriteLine(S"Chapter01 Sample");
```

和

```
Application::Run(new Sample01());
```

Application 类是 System::Windows::Forms 命名空间的一部分，包含着用来管理 Windows 应用程序的静态方法，而 Run 方法是其中之一，该方法用来创建和显示一个窗体，并且在当前线程启动一个消息循环。

大家可以试着调试这个程序，并在自己感兴趣的地方设置断点，以加深理解。

最后，编译并执行这个程序。

小结

2000 年 6 月，微软公司宣布了.NET 战略。.NET 的一个关键特性在于，它独立于任何特定的语言或平台。开发人员可以使用任何语言来创建一个.NET 应用程序，只要这种语言有.NET 版本。

.NET 的一个关键部分是 Web 服务。Web 服务实际上是一种应用模式，用于通过 Internet 向客户提供服务。客户或其他应用程序可以将 Web 服务用作可重用的软件构建单元。

.NET 框架是构建并运行应用程序的平台，它主要由公共语言运行库（CLR）和.NET 框架类库（FCL）组成。CLR 抽象了操作系统服务，并用作托管应用程序的执行引擎。FCL 为开发各种应用程序提供了一套可重用的类。

CLR 是.NET 框架的心脏。开发人员编写的每段框架代码，不是运行在 CLR 中，就是得到 CLR 的许可后在其外部运行。托管应用程序要经过两个步骤，才能编译为可被处理器识别的机器代码：

（1）程序代码被编译为 MSIL 代码，MSIL 定义了用于 CLR 的指令。

（2）实际运行时，CLR 使用实时编译器将 MSIL 编译为机器代码。