清华电脑学堂

Visual C++.NET 程序设计 实践教程

邵良杉 李白萍 王新海 等编著

- 总结了作者长期教学培训成果，难易适中，实用性强
- 系统全面地介绍了 Visual C++.NET 的技术要点
- 围绕丰富实验讲解了编程实践知识
- 精心编写了大量“实验指导”，引导学生深入练习编程实践
- 网站提供代码下载和课件支持

清华大学出版社

清華电脑学堂

Visual C++.NET 程序设计 实践教程

■ 邵良杉 李白萍 王新海 等编著

清华大学出版社
北 京

内 容 简 介

本书主要介绍 Visual C++.NET 编程知识。全书主要内容包括.NET Framework 中的公共语言运行时、类库、中间语言、JIT 编译器；托管 C++的数据类型、各种运算符以及程序控制语句。Visual C++.NET 中的函数使用，数组和指针的创建和使用。结构、枚举和联合等数据类型。

本书深入介绍了面向对象的程序设计知识，包括类的继承、虚函数、多态性、以及新增的接口与委托。之后，本书介绍了在 Visual C++.NET 中创建窗体的应用知识；Windows 文件系统操作知识。讨论了 ADO.NET 和主要的 ADO.NET 类。本书最后介绍了 Visual C++.NET 绘图技术和图形设备接口应用知识。

本书可以作为读者学习 Visual C++.NET 语言和面向对象开发的教程，适合作普通高校计算机专业和非计算机专业的程序设计教材，也可供自学读者使用。

图书在版编目（CIP）数据

Visual C++.NET 程序设计实践教程 / 邵良杉，李白萍，王新海等编著. —北京：清华大学出版社，2007.1
ISBN 978-7-302-14219-5

Ⅰ. V…　Ⅱ. ①邵…　②李…　③王…　Ⅲ. C 语言—程序设计－教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2006）第 142427 号

责任编辑：夏兆彦
责任校对：张　剑
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c－service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京密云胶印厂
装 订 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：25.5　**字　数**：604 千字
版　　次：2007 年 1 月第 1 版　　**印　次**：2007 年 1 月第1 次印刷
印　　数：1～5000
定　　价：39.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　产品编号：022165－01

前言

FOREWORD

Visual C++.NET 作为 Visual Studio.NET 的重要组成部分，它在保留原 Visual C++编程的基础上，加入了.NET 环境中一系列新特性。托管 C++语言不仅继承了传统的 C++的优点，并且具有.NET 开发语言的专长（如内存的自动管理），用于在.NET 平台上开发应用程序的程序开发语言。

Visual C++.NET 是一种强大的开发工具，在 Windows 图形用户界面应用程序、ASP.NET Web 应用、ADO.NET 数据库等方面有着广泛的应用。

本书内容

本书介绍 Visual C++.NET 编程知识。全书内容包括.NET Framework 中的公共语言运行时、类库、中间语言、JIT 编译器；托管 C++的数据类型、各种运算符以及程序控制语句。Visual C++.NET 中的函数使用，包括对函数的定义、调用函数以及特殊的函数调用——递归调用。数组和指针的创建和使用。

本书介绍了结构、枚举和联合等数据类型。深入介绍了面向对象的程序设计知识，包括类的继承、虚函数、多态性，以及新增的接口与委托。

之后，本书介绍了在 Visual C++.NET 中创建窗体的应用知识；Windows 文件系统操作知识。讨论了 ADO.NET 和主要的 ADO.NET 类。本书最后介绍了 Visual C++.NET 绘图技术和图形设备接口应用知识。

本书使用 Microsoft Visual Studio.NET 2003 作为开发工具。书中的 C++基础知识，可以满足程序初学者学习 C++，也可以作为对 C++在.NET 下全新认识的开始。

本书特色

本书通过实例介绍 Visual C++.NET 程序开发知识，具有实用性教程的特色。

- 本书汇总了多年的程序员职业教学培训经验，内容组织更合理，实例丰富全面。
- 本书使用 Visual C++.NET 语言开发了大量实例，读者可以通过这些丰富实例学习 Visual C++.NET 编程实践知识。
- 每章编写了大量“实验项目”，引导读者应用该章知识独立练习编程项目。
- 复习题可以帮助学生检查对 Visual C++.NET 开发理论知识的掌握程度。

读者对象

本书由多家院校的教师联合编写，在各家院校成熟教案及原有自编教材的基础上整合编写。作者均从事 Visual C++.NET 教学和开发工作，拥有丰富的 Visual C++.NET 开发案例和教学经验。本书共 14 章，需要 42 个课时。为了给教师授课提供方便，本书提

供了教学课件，读者可以从 www.tup.tsinghua.edu.cn 下载使用。

本书可以作为普通高校计算机相关专业 Visual C++.NET 编程初级教材，也可以作为接触过 Visual C++基础知识，需要深入学习 Visual C++.NET 开发的中级教材。编写过程难免会有错误，欢迎读者与我们联系，帮助我们改正提高。

编　者

目 录

CONTENTS

第 1 章　Visual Studio.NET 基础

本章介绍.NET 的设计原理，以及 Visual Studio.NET 各组成部分之间的相互关系，让用户能够理解 Visual Studio.NET 编译器的内部原理和 Visual Studio.NET 应用程序的组成部分。为了加深对.NET 的理解，本章使用诸如记事本之类的文本编辑器创建一个简单的应用程序，然后从命令提示窗口编译和执行该程序。最后本章将介绍 Visual Studio.NET 的集成开发环境（IDE）。

本章学习要点：

- 了解标准 C++与 VC++.NET 的区别
- 了解.NET Framework 和 Visual Studio.NET 之间的关系
- 理解.NET Framework 的构成
- 了解什么是程序集
- 理解命名空间
- 在文本编辑器中创建托管 C++应用程序
- 熟悉 Visual Studio.NET IDE 开发环境

1.1 C++、VC++.NET 和 VS.NET 简介

在介绍 VC++.NET 之前，我们首先需要弄清楚 C++、VC++.NET 和 VS.NET 之间的关系。首先介绍 C++语言。C++语言是 20 世纪 80 年代由 Bjarne Stroustrup 在 Bell 实验室开发的。该语言是作为 C 语言的改进而开发的，C 语言也是在 Bell 实验室产生的。C++中面向对象的特性使其比 C 语言强大得多。1997 年，美国国家标准协会（ANSI）的一个委员会——国际标准化组织（ISO）将 C++的一个版本作为标准，也就是常说的标准 C++。

Visual Studio.NET 则是由 Microsoft 开发的用于 Windows 应用程序的开发平台。在 Visual Studio.NET 环境中，可以使用 VB.NET、VC++.NET、C#.NET 和 J#.NET（其中“#”读做 Shape）等开发语言。其中 VB .NET，C#和 Visual J#是 3 种微软支持的用于.NET Framework 的主要开发语言，VC＋＋.NET 则是微软在标准 C++的基础，对其进行扩充使之更加适用于 Windows 程序开发。在 Visual Studio.NET 变革了 Windows 应用程序的开发，原因如下：

- Microsoft 公司提供了 4 种 Visual Studio.NET 语言：C++.NET、C#、Visual J#和 VB.NET。不管选择何种 Visual Studio.NET 开发语言，用来编写应用程序的集成开发环境都是相同的。实际上，Visual Studio.NET 允许用户使用一种语言创建一部分应用程序，然后使用另一种语言创建应用程序的其他部分。
- 每一种 Visual Studio.NET 语言都提供了相近的功能，因此，确定使用哪一种

Visual Studio .NET 语言不取决于特定语言的功能，而是取决于开发人员的熟练程度和喜好。

- 所有.NET 语言的性能都相近，因此选择何种语言无关紧要。

尽管使用 Visual Studio .NET IDE 可以很方便地创建大多数.NET 应用程序，但是也可以不使用 Visual Studio.NET IDE，而完全使用文本编辑器创建应用程序，然后通过从命令提示窗口调用合适的编译器来编译应用程序。实际上，甚至不需要在计算机上安装 Visual Studio.NET 就可以创建应用程序。原因很简单，有另外一种名为.NET Framework 的内部软件组件负责编译和执行应用程序。开发和测试应用程序唯一需要的软件组件就是.NET Framework。

1.2 .NET Framework 与 Visual Studio.NET 之间的关系

Visual Studio.NET 程序开发环境的核心是名为.NET Framework 的软件组件。要想开发和运行.NET 应用程序，必须在计算机上安装.NET Framework 组件。.NET Framework 包含把.NET 应用程序转换为可执行文件所需的所有编译器。程序开发人员甚至可以在文本编辑器中创建应用程序，不必使用 Visual Studion.NET IDE。但是 Visual Studio.NET IDE 提供了一个健壮的集成开发环境，可以用来开发、测试和部署使用 Visual C++、C#、Visual J# 或者 VB.NET 编写的应用程序。尽管不使用 Visual Studio.NET IDE 也可以编写应用程序，但是使用 Visual Studio.NET IDE 会得到更高效率。图 1-1 演示了.NET Framework 和 Visual Studio.NET 之间的关系。

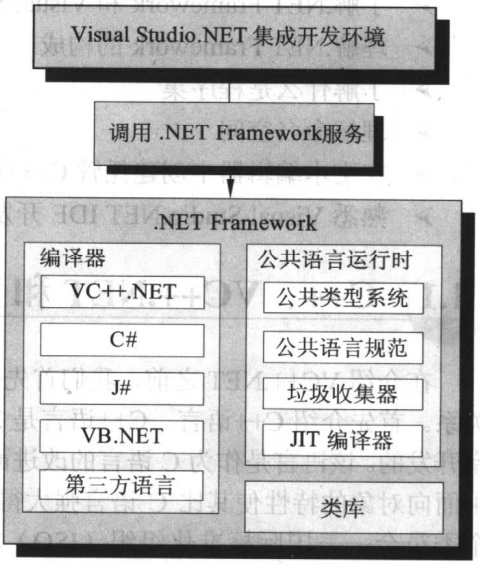

图 1-1 .NET Framework 和 Visual Studio.NET 之间的关系

如图 1-1 所示，Visual Studio.NET IDE 依赖于.NET Framework 提供的服务。这些服务包括微软开发的语言编译器或者第三方提供的语言编译器。这些语言编译器是.NET Framework 自身的组成成分，它不属于 Visual Studio.NET IDE。Visual Studio.NET 提供了大量的工具来调用某一种安装的编译器。下面将重点介绍.NET Framework 的构成。

1.3 .NET Framework

不管在 Visual Studio.NET 中使用何种.NET 开发语言，都必须安装.NET Framework。这是因为它们都依赖.NET Framework 来编译和执行应用程序。.NET Framework 定义了语言之间互操作的规则，以及如何把应用程序编译成为可执行代码。.NET Framework 还

负责管理任何 Visual Studio .NET 语言创建的应用程序的执行。

1.3.1 公共语言运行时

.NET Framework 为所有的.NET 开发语言提供了一个公共的运行环境，而不是为每一种语言提供各自不同的运行环境。这个公共的运行环境称为 Common Language Runtime（CLR），即公共语言运行时。CLR 除了在运行时管理代码的执行之外，CLR 也为所有适应 CLR 的语言提供了一套公共的服务。下面列出了 CLR 提供的一些重要服务。

- **公共类型系统（Common Type System，CTS）** 它定义了所有.NET 语言的标准数据类型及其格式。例如，CTS 定义了整型是 32 位大小，还指定了整型值的内部格式。
- **公共语言规范（Common Lauage Specification，CLS）** 它定义了各语言间互操作性的规则。由于 CLS 定义了规则，任何一种.NET 开发语言创建的类就可以由其他.NET 语言使用。
- **JIT 编译器（Just-In-Time 编译器）** 当.NET 应用程序第一次编译时，编译成一种可以由所有.NET 语言共享的中间语言。在应用程序执行时，再由 JIT 编译器把中间语言转换为可以在目标计算机上执行的可执行文件。
- **代码管理** 在创建和销毁对象时，由 CLR 负责分配和释放内存。
- **垃圾收集器（Garbage Collector，GC）** 释放无用对象所占用的内存。

下面我们分别介绍各个服务。

1. 公共类型系统

公共类型系统（CTS）定义了一套标准的数据类型和一套创建新数据类型的标准，使得 CLR 可以在用不同开发语言开发的应用程序之间管理这些标准化类型，并且可以在不同计算机之间以标准格式进行数据通信。下面描述 CTS 的功能。

- CTS 定义了所有应用程序使用的主要.NET 数据类型，以及这些类型的内部格式。例如，CTS 指定了诸如 int 和 single 这样的基本数据类型的大小和格式。
- CTS 指定了如何为结构和类分配内存。
- CTS 允许不同语言所开发的组件可以互操作。
- CTS 实施类型安全性。类型安全性禁止一个应用程序使用为另一个应用程序分配的内存。

虽然在.NET Framework 里，通过 CTS 使得语言的互操作变得很容易，但仅靠它还是不足以保证用不同语言开发的应用程序之间的便利交互。为此，CLR 提供了另一种叫作公共语言规范的组件。

2. 公共语言规范

公共语言规范（CLS）是公共类型系统的子集。CLS 和 CTS 一起定义了允许不同编程语言使用的标准集，使得由这些编程语言编写的应用程序可以互操作。

CLS 和.NET 自身都依赖于 Windows 应用程序编程接口（Windows Application

Programming Interface，API）提供的低级服务。API 提供了诸如文本框、按钮、滚动条、列表框和组合框这样的基本控件类。还提供了 Windows 服务来管理文件、进程和内存。

CLS 定义了所有基于.NET Framework 的语言都必须支持的最小功能集。CLS 定义的规则可以概括如下：

- CLS 定义了变量的命名标准规则。例如，与 CLS 兼容的变量名都必须以字母开始，并且不能包含空格。
- CLS 定义了基本数据类型，如 Int32，Int64，Single，Double 和 Boolean。
- CLS 禁止无符号数值数据类型。有符号数值数据类型的一个数据位用来指示数值的正负。无符号数据类型没有保留这个数据位。
- CLS 定义了对基于 0 的数组的支持。
- CLS 指定了函数参数列表的的规则，以及参数传递给函数的方式。例如，CLS 禁止使用可选的的参数。
- CLS 定义了事件名和参数传递给事件的规则。
- CLS 禁止内存指针和函数指针。

除了上述标准之外，CLS 还定义了其他标准，这里不做深入介绍。

任何语言都可以扩展基本的 CLS 要求。例如，有些语言（如 C#）支持无符号整型。不鼓励使用非标准的功能，因为这样做就妨碍了语言之间的互操作性。完全符合 CLS 的语言称为兼容 CLS 的语言。

3. JIT 编译器

在传统的应用程序开发中，程序设计人员所开发的应用程序直接被编译成针对某种机器的可执行文件。然而，不同厂商生产的 CPU 体系结构也不相同，也就是说，它们具有不同的寄存器，并且执行一套自己特有的指令。例如，Macintosh CPU 的指令集与 Intel Pentium 的指令集就不同。这就造成了相同的应用程序不能在不同的机器上运行，即程序不具有可移植性。为了克服该问题，Apple 公司开发了 Virtual PC 应用程序，它允许苹果机用户通过把 Pentium CPU 指令转换为 Macintosh CPU 指令，从而运行 Windows 应用程序。这种类型的应用程序通常称为模拟器（emulator）。

尽管这些软件模拟器有效地解决程序的可移植性，但是它们的执行速度慢于本机的执行速度。另外，软件模拟器也无法完全利用 CPU 的细微的增强功能，例如从 Pentium III 到 Pentium IV 的增强功能，模拟器就无法加以利用。

.NET Framework 解决了不同硬件体系结构之间程序不可移植的缺点。它的解决方法是把应用程序编译为一种称为 Microsoft Intermediate Language（MSIL）的独立于硬件的语言，即中间语言。

中间语言的格式通常不依赖于特定的 CPU 体系结构。也就是说，中间语言不直接引用 CPU 寄存器或者执行 CPU 指令，而当用户执行中间语言格式的应用程序时，另一个名为 Just-In-Time（JIT）编译器的应用程序进一步将中间语言转换，生成目标 CPU 可以执行的本地可执行文件。

除了不同 CPU 厂商会制造各种不兼容的硬件之外，同一个 CPU 厂商，也常常会制造出支持特殊指令的新 CPU。这样把应用程序编译为独立于 CPU 的中间语言，当用户执行

应用程序时，再把中间语言优化为特定 CPU 可以执行的可执行文件。因此，微软为每一种目标 CPU 提供了不同版本的 JIT 编译器，以便利用各种 CPU 的独特功能提高性能。

4. 执行代码管理

CLR 执行的代码称为托管代码（managed code）。托管代码的作用之一就是防止一个应用程序干扰另一个应用程序的运行。这个过程称为类型安全性。使用类型安全的托管代码，一个应用程序就不会覆盖另一个应用程序分配的内存。创建托管代码需要如下 4 步。

首先，必须选择一种合适的开发语言，它能够生成适合的 CLR 执行的代码，并且能够使用.NET Framework 提供的服务。目前，微软提供了 4 种兼容.NET Framework 的语言：VB.NET，C#，Visual J#和 C++。同时也可以使用第三方供应商提供的兼容.NET Framework 的语言。

其次，把应用程序编译为独立于机器的中间语言。

然后，在执行应用程序时，把中间代码转换成本机可执行文件。本机可执行文件可以在目标 CPU 上执行。这个过程由 JIT 编译器完成。

最后，应用程序执行时，会调用.NET Framework 和 CLR 提供的服务。

图 1-2 演示创建托管代码的过程。

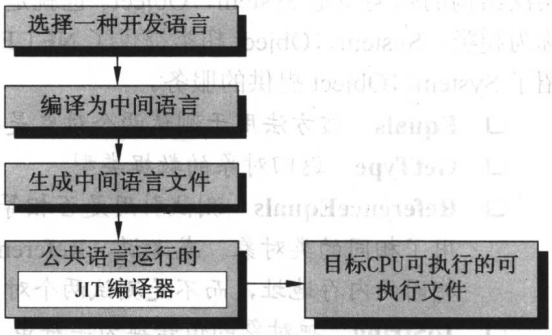

图 1-2 创建和执行托管的可执行文件

并不是 Visual Studio.NET 生成的所有可执行文件都依赖于 CLR 提供的服务，这种不依赖于 CLR 的可执行文件称为未托管的可执行文件或未托管代码。这里我们需要重点介始微软提供的 4 种.NET Framework 开发语言。

- **Visual C#.NET** Visual Studio.NET 系列提供了一种新的面向对象的语言——Visual C#.NET。Visual C#.NET 结合了 Visual C++和 Visual Basic 两者的优点。Visual C#.NET 允许创建托管和非托管代码。
- **Visual Basic.NET** Visual Basic 毫无疑问是到目前为止最容易的编程语言。Visual Basic 在.NET 中进行显著改进，最显著的变化是 Visual Basic.NET 现在是纯面向对象的，从而完全支持面向对象的特征。
- **Visual J#.NET** Visual J#.NET 是为 Java 语言开发人员提供的开发工具。Java 程序开发人员使用它可以很容易在.NET Framework 上创建托管应用程序。
- **Viual C++.NET** Visual C++.NET 是唯一允许创建与.NET Framework 不兼容应用程序的语言。这就意味着还可以创建传统的 Windows 应用程序(使用 MFC(微软基本类库))。除此之外，ViualC++.NET 也可以创建托管应用程序。在本书中，我们主要介绍 Visual C++.NET 创建托管应用程序的知识。

5. 垃圾回收

CLR 提供的一个重要功能就是内存管理。当应用程序（进程）启动时，CLR 为该进

程分配连续内存区域，该区域称为托管堆（managed heap）。当 CLR 从类创建对象时，将会从托管堆中为这些对象分配内存。如果 CLR 一直分配内存，这必然会耗尽可用的内存资源。因此，CLR 会定期使用另一个称为垃圾回收器（Garbage Collector，GC）的组件从托管堆中释放内存。GC 靠定期运行来释放托管应用程序（进程）中不再需要的内存。垃圾回收只对托管应用程序才有效。

需要注意的是，GC 是自动运行的，不需要人工干预。

1.3.2 .NET Framework 类库

.NET Framework 类库是.NET Framework 的另一个重要组成部分。类库中包含了.NET Framework 中定义的所有类。这些类中既有用于帮助创建 Windows 和 Web 应用程序的类（基础类），同时也包括了用于进行其他数据处理的类。

.NET Framework 中所有的类和用户创建的类都被组织成层次结构。.NET Framework 层次结构的基类型是 System::Object。也就是说，System::Object 位于层次结构树的根，称为超类。System::Object 超类提供了.NET Framework 中所有类型的基本功能。下面介绍了 System::Object 提供的服务。

- **Equals** 该方法用于测试两个对象是否包含相同的数据。
- **GetType** 返回对象的数据类型。
- **ReferenceEquals** 测试引用是否相等。也就是说，它测试两个对象变量是否引用了相同的类对象。或者说，ReferenceEquals 测试了两个对象变量是否引用了相同的内存地址，而不是测试两个对象变量包含相同的数据。
- **ToString** 把对象的值转换为字符串。

由于 Object 是.NET Framework 中所有类的父类，其他类都从该类派生而来（关于派生随后的章节将详细说明）。因此，在.NET Framework 中，所有的类都支持上述的方法。

1.4 程序集概述

程序集是.NET 应用程序的主要特征之一。一个程序集是类型（比如类）和其他资源的集合，这些资源共同作用以实现所需要的功能。另外，每一个程序集中都包含了一个程序集清单。程序集清单描述了程序集以及组成程序集的各个模块。程序清单常常称为程序集元数据。下面列出了程序集清单的其本元素。

- 程序集包含一个描述性名称和可选的位置信息。描述性名称和位置信息称为程序集的身份。
- 清单描述了程序集的内容。程序集内容描述了程序集提供的类型（如类）。
- 一个程序集的运行可能会依赖于一个或者多个其他程序集。因此，程序集清单中还包含了一个依赖关系列表。
- 清单中还包含许可。程序集许可指出谁可以使用程序集的内容。

除程序集元数据外，程序集通常还包含另一种元数据，即类型元数据。类型元数据描述了程序集中的类型以及这些类型的成员。例如，如果一个程序集中包含一个名为

AccessData 的类，它带有 Read 和 Write 方法，这些信息就会保存在程序集的类型元数据中。除了程序集元数据和类型元数据外，程序集还包含实际的中间代码。最后，程序集还包含可选的资源。可选资源包括位图、图标和光标等应用程序需要的支持文件。

.NET Framework 类库也由许多程序集组成。下面列出.NET Framework 类库中的少数几个程序集及其作用，这些程序集几乎都可用于图形用户界面程序的开发。

- 程序集 System.dll 定义主要数据类型和最基本的类型，如 System.Object。
- 程序集 System.Data.dll 定义组成 ADO.NET 的组件。ADO.NET 提供了在 Visual Studio.NET 中访问数据库的服务。
- 程序集 System.Drawing.dll 包含了用于向输出设备（如屏幕和打印机）绘制各种图形（矩形、直线等）的组件。
- 程序集 System.Windows.Forms.dll 包含用来实现 GUI 应用程序使用的窗体的组件，以及创建这些窗体的控件。

.NET Framework 类库包含大约 100 种不同的程序集（库文件），这些程序集定义了组成.NET Framework 类库中的组件。

1.5 命名空间

前面说过，.NET Framework 的类库中包含了大量的类，大约包含了 3500 个类，所以在程序设计人员需要快捷的方法找到所需要的类。为此.NET Framework 被分为许多命名空间，而在一个命名空间中包含了功能相似的类。

命名空间实际上也是分层的，这意味着一个命名空间可以包含另一个命名空间，而后者又包含了更为类似的类。每一个类一定完全属于一个命令空间，它不会同时属于多个命名空间。

下面列出了几个在应用程序中常用的命名空间。

- System 命名空间包含了定义数据类型、事件和事件处理程序的基本类。
- System::Data 命名空间包含提供数据访问功能的命名空间和类。这些命名空间和类构成了 ADO.NET。
- System::Drawing 命名空间包含了提供与 Windows 图形设备接口（Windows Graphical Device Interface，GDI）的接口类。这些类定义了各种绘图类型，如矩形和直线。
- System::IO 命名空间包含了从其他数据源读取和写入顺序文件和数据的类。
- System::Windows::Forms 命名空间定义了包含工具箱中的控件及窗体自身的类。
- System::Net 命名空间包含了用于网络通信的类或命名空间。

命名空间是.NET Framework 类库中类逻辑组织形式，它的物理组织形式以程序集的形式存放。一个程序集可以包含一个或多个命名空间。例如，System 和 System::IO 命名空间都保存到 System.dll 程序集中。另外，一个命名空间也可以存在于多个程序集。

在.NET Framework 中，一个命名空间中包含了一种或者多种类型，int 或者 String 被认为是类型。每一个类也被视为类型。例如，System 命名空间包含名为 Int32 的类型，它代表一个整数值。在.NET Framework 中类型分为两类，即值类型和引用类型。