精通C#程序设计

南泰电脑　吕文达　编著

- 以超过200个详细解说、内容精辟的实用范例,贯穿全书所有主题。
- 完整地说明利用C#结合.NET Franework开发应用程序所具备的知识与相关程序设计技巧。
- 不同于其他探讨C#主题的技术书籍、避免难以理解的理论说明与繁杂的技术数据列表，以深入浅出的风格，针对.NET技术主题，翔实地说明与探讨。
- 精心设计、逐行说明的实用范例，可以快速引导读者熟悉C#与.NET Franework各种主题内容与实际应用。
- 读者在范例程序的演练过程中，可以强化开发.NET应用程序的程序编写技巧。

清华大学出版社

精通 C#程序设计

南泰电脑　吕文达　编著

清华大学出版社
北　京

内 容 简 介

本书以C#语言为基础，通过大量的范例及简明扼要的解析阐述开发各种.NET应用程序所必须掌握的技巧。本书共20章，主要介绍：C#基础概要，C#语言基础，类与方法，面向对象程序设计，运算符重载，数组与矩阵，集合，文字处理，异常处理，事件与托管，文件的输入/输出与数据流，多线程设计，窗口程序设计，数据库应用程序，绘图，组件、属性与映射，远程服务与应用程序定义域，网络应用程序，组件应用程序等内容。

本书的特色是：起点低、入门快，实例精。本书适合想要了解如何使用C#语言开发.NET应用程序的读者作为教材使用；不论读者是否具备程序设计的背景，都可以从本书中受益。

图书在版编目(CIP)数据

精通C#程序设计/南泰电脑，吕文达编著.—北京：清华大学出版社，2004.11
ISBN 7-302-09186-2

Ⅰ.精…　Ⅱ.①南…②吕…　Ⅲ.C语言—程序设计　Ⅳ.TP312

中国版本图书馆CIP数据核字（2004）第080727号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　**邮　　编**：100084
社 总 机：010-62770175　**客户服务**：010-62776969
责任编辑：桑任松
封面设计：陈刘源
印 刷 者：北京市清华园胶印厂
装 订 者：三河市化甲屯小学装订二厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：43.25　**字数**：1036千字
版　　次：2004年11月第1版　2004年11月第1次印刷
书　　号：ISBN 7-302-09186-2/TP·6464
印　　数：1～4000
定　　价：58.00元

前　言

C# 是一种应用于开发 .NET 应用程序的专属语言，它以 C 语言为基础，沿用其大部分的关键字、表达式以及运算符，而语法的设计却更为简洁易懂，同时以 C++ 对象模型，建构其面向对象的语言架构，支持完整的面向对象设计理论。

C# 本身的设计，考虑各种语言的特性，避免了现有程序语言的缺点，并且新增了多种出色的语言特性，如垃圾回收机制、类型安全以及异常处理等。

C# 保留了 C++ 的功能强大特性，兼具 Visual Basic 易于使用的优点，搭配 .NET 所提供的类库，改变了传统应用程序的开发模型，加上微软对于 .NET 平台的全力支持，C# 在可预见的未来将会扮演非常重要的角色。

关于本书

本书以 C# 语言为基础，说明开发各种.NET 应用程序所需了解的技巧，而其中最大的特色在于利用大量范例，精要地说明各种相关技术的实际应用过程。全书并没有太多关于理论的描述以及相关资料的列举，大部分的篇幅着重于范例的说明以及关键知识的探讨，最大的目的在于引领读者，利用消化作者整理过的信息，快速了解 C# 这个全新的程序设计语言以及实际应用于开发.NET 应用程序所需具备的重要知识。

适合对象

本书定位在想要了解如何使用 C# 开发 .NET 应用程序的初中级读者，其内容涵盖了最基础的语法说明，因此初学的读者研读本书并不会有任何困难，而具备 C++ 、Java 或是 Visual Basic 等程序设计语言背景的读者，在阅读时也会非常轻松地获得提高。不管怎样，由于 C#的语法使用及各种机制都相当一致，相信各种级别的读者，都能够在本书找到所需要的信息。

内容架构

全书总共包含 20 章，分类如下：

C# 语言　这一部分包含 C# 语言的相关特性介绍，如第 1 章的 C# 基础概要、第 2 章的语言基础、第 3 章的类与方法说明、第 4 章面向对象程序设计、第 5 章运算符、第 6 章的数组、第 9 章的例外处理在程序调试中的应用以及第 10 章的委派与事件。

类库的应用　C# 本身强大的功能，有一大部分是利用结合 .NET 所提供的类库而来，本书介绍各种类库所提供的类，如第 7 章的集合类接口、第 8 章文字处理与规则表达式、第 10 章的委派与事件、第 11 章的文件 I/O 数据流、第 12 章多线程技术以及第 17 章的属性与映射，这几章说明了支持编写各种应用程序的特定类，提供实现的范例程序说明。

窗口应用程序　第 13 章窗口应用程序(I)、第 14 章窗口应用程序(II) 以及第 16 章绘图与 GDI+ ，这三章探讨窗口界面的应用程序开发以及绘图程序的相关应用说明。

数据库应用程序　NET 数据库应用程序的开发，由 ADO.NET 提供相关的支持。第 15 章提供了这方面的详细说明，此外对 SQL 语法也提供了相关的探讨。

网络与分布式应用程序 .NET 比起传统的 Win32 应用程序，最大的改进在于分布式应用程序的开发，本书对于在分布式环境下开发应用程序所需的相关技术说明，集中在第 18 章的远程服务 、第 19 章的网络程序以及第 20 章的组件应用程序中进行阐述。

目　录

第 1 章　C# 基础概要

C#是一种全新的面向对象语言，专门用于开发在.NET 平台上执行的应用程序，本书第 1 章，将从一个最简单的应用程序开始，介绍 C# 语言的基本结构及语法，通过各种范例的操作，引领读者进入 C# 程序语言的设计领域。

C# 融合了其他语言的优点开发而成，使用类似 C 的语法，所以可以在 C# 身上找到其他程序语言的影子，例如，C++、Java 以及 Basic 等，然而比起这些语言，C# 在许多方面被设计得更为出色，对于面向对象设计理论的支持也非常彻底；而通过与.NET Framework 的结合，与类库的支持，程序开发人员因此得以使用 C# 开发出更稳固、功能更强大的应用程序。

整个 C# 语言建立于.NET Framework 之上，本章将从 .NET Framework 的基础知识开始，说明 C# 与 .NET Framework 之间的关系，并且利用一个简单的范例程序，简要地说明各种 C# 语言的基本要素以及概念。

1.1　.NET Framework 与 C# 应用程序设计

在进入 C# 内容介绍之前，首先我们必须使用一些篇幅简要地说明.NET Framework 相关的基础知识，有了.NET Framework 的基本认识之后，将会有助于你快速地学习 C# 这个专为 .NET 平台所设计的全新语言；你很难单独学习 C# 而不去了解何谓 .NET Framework，C# 利用与.NET Framework 紧密的结合而得以轻易地开发功能强大的应用程序。

.NET Framework 是一种全新的运算平台，它提供了在 Windows 操作系统执行应用程序所需的服务，以及相关的应用程序接口，其中包含了两个主要部分，Common Language Runtime(公共语言运行库) 以及 .NET Framework 类库 (Class Library)，分别列举说明院如下：

(1) Common Language Runtime　简称为 CLR，其本身可以视为用于执行程序代码管理的机制，其中提供了各种服务，例如，多线程管理、内存管理以及类型检查等等，应用程序的源代码在这样的机制下，确保了其安全性与正确性。

Common Language Runtime 是 .NET Framework 的基础，控制和管理程序代码的运作，建立在 CLR 上所执行的程序代码被称为托管 (Managed) 的程序代码。一般来说，C# 所编写的应用程序均必须在 CLR 的环境下执行，也因此受到 CLR 的监控，本身均为托管代码。

其他无法在 CLR 的环境下执行的程序代码，我们则将其称为未托管 (Unmanaged) 的程序代码，例如，.NET 出现之前的 Visual Basic 6.0、C++ 等等，均属于这一类的程序代码。

(2) 类库 (Class Library)　包含了以面向对象理论设计，提供开发应用程序所需功能的各种类型，其涵盖的功能相当广泛，从最基本的数据流 I/O、创建丰富用户接口所需的应用

程序接口、多线程应用程序的编写，到提供复杂的分布式远程服务应用程序开发，程序开发人员几乎都可以在其中找到相关的支持。

C# 利用.NET Framework 所提供类库的支持，得以开发出各种功能丰富的应用程序，当你继续往下研读本书所探讨的各章节内容时，你将会发现除了 C# 语言基本语法、内容结构以及面向对象理论的相关探讨，其他章节所涉及的内容都与类库所提供的特定类型脱不了关系；类库所包含的各种不同类型，能够提供编写应用程序所需的功能，通常你所必须做的只是引用所需的类型，运用其提供的方法与属性成员，建立解决特定问题的应用程序。

从本章开始，你会看到全书均是以此种模式，编写范例应用程序。例如，当你想要开发一个具有多线程功能的应用程序时，可以直接引用类库所提供的 Thread 类完成所需的功能；如果想创建窗口界面，则必须利用 Form 类以及相关的可视化控件来完成；甚至当你想要开发一套数据库系统时，类库也创建了一组功能非常强大的 ADO.NET 类库，提供开发人员一切所需的数据库应用程序接口；很快你会发现，.NET Framework 提供你创建应用程序所需的一切基础，当 C# 语言的基本语法学习告一段落后，你的大部分精力将会投注在了解如何使用类库里所提供的各种类。

1.2 类　　库

前述提及类库包含了功能广泛的各类类型，在稍后的内容，我们将会对其意义以及在 C# 语言里所扮演的角色，做进一步说明。你现在所必须知道的是，类型为创建于.NET 平台上执行的应用程序的基础元素，整个类库包含的类型，按其功能，大致可以分为以下几类：

- 提供基础数据的相关类型　其中包含了类型之间的转换操作、格式化以及字符串数据处理等相关功能。
- 使用于应用程序的异常处理机制　用于在应用程序执行期间提供一致性的错误处理机制，其中包含了异常返回以及错误的捕捉。
- 执行期查看加载的类型信息　提供执行期间，应用程序加载类型信息的查看与创建类型执行实例 (Instance)，并且针对这些执行期间动态产生的类型实例，进行调用 (Invoke) 以及存取操作。
- 封装数据结构　提供如对象集合、数组、字典以及散列表等相关数据结构的实现。
- I/O 流的操作　允许针对数据流和文件进行读取和写入操作，其中文件包含了目录路径、磁盘盘符等。数据流对于各种存储器，支持缓冲写入和读取字节的操作，其中包含了如网络流、内存流等数据流等等。
- 提供数据存取　允许程序设计人员使用其所提供的数据存取服务，进行各种类型数据库的数据维护操作，这方面的功能由一组 ADO.NET 所提供的组件来完成。
- 制作丰富的用户界面　包含制作窗口界面、绘图功能的相关类型。

1.3　C# 程序语言

C#是一种专为.NET Framework 所设计的面向对象程序语言，使用 CLR 提供的服务，创建解决特定问题的应用程序，它是 C++ 经过改良，以面向对象设计理论为基础发展而来的，特别是在语法上，C# 大致承袭了 C++ 的风格，另一方面也同时弥补了 C++ 的欠缺和不足的功能，比如增加了类型安全、版本控制、事件和内存回收等机制。

C# 被设计成为比起过去你所学习的程序语言（如果有的话），例如 C++，功能更为强大，并且具有如 Visual Basic 等程序语言容易使用的优点，由于其完整的面向对象特性以及本身与类库的紧密结合，使得开发功能强大的商业级应用程序更为容易；C# 应用程序本身由一个以上分开的程序代码文件所组成，其扩展名为 "cs"。最简单的 C# 程序代码文件可以使用记事本之类的文本编辑器，经过将其保存为 ".cs " 扩展名的文件来完成。

C# 编译器提供将程序文件编译成为可执行文件（扩展名为 .exe）的功能，C# 编译器名称为 csc.exe ，你可以在命令行输入可执行文件名称启动这个编译器，进行程序的编译操作。

以下内容，我们将从一个简单的程序范例开始，介绍使用 C# 程序语言所必须了解的相关知识，并且说明其基本语法。

1.3.1　第一个 C# 应用程序

编写一个简单的 C#应用程序并不困难，如同一般介绍程序语言的书籍一样，我们使用一个简单的 SayHello 应用程序，作为 C# 程序语言探讨的起点。打开 Notepad 文本编辑器，输入以下范例程序代码：

```
// SayHello.cs
//------------------------------------------------------------
//  第 1 个 C# 应用程序
//------------------------------------------------------------

class SayHello
{
    static void Main()
    {
        System.Console.WriteLine(" Hello Welcome ") ;
    }
}
```

完成后，将其扩展名改为 ".cs"，并且以 "SayHello" 名称存盘。接下来，将 C# 程序代码文件编译成为可执行文件。你可以有几种方法完成编译。操作，其中简单的方法便是使用命令行，然而，前提在于使用的计算机上必须安装了 .NET Framework。你可以从微软公司的网站取得完整的.NET Framework，根据其提示，完成安装操作。现在假设你的计算机上已完成相关的环境设置，在命令行窗口输入以下的命令 ：

```
csc SayHello.cs
```

其中"csc" 是 C# 编译器 csc.exe 的名称，SayHello.cs 则是所要编译的 C# 程序代码文件，执行此编译指令，产生一个扩展名为 "exe" 的 SayHello 可执行文件，用鼠标双击可执行文件，得到执行结果如图 1.1 所示。

一个能够在主控台输出指定文字信息的应用程序就这样完成，这个程序虽然简单，但是其中包含了编写 C#应用程序所需了解的一些基础知识。我们现在利用这个范例程序代码，说明 C# 程序语言的一些重要基础知识。

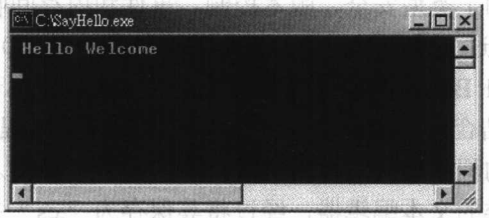

图 1.1

1.3.2 程序解析

1. 注释

注释被应用于程序语言中，说明注记程序代码的内容。善用注释，可以使应用程序增强可读性，尤其是当你所开发的应用程序项目愈来愈大的时候，注释能够帮你很快地了解自己先前写下的程序代码内容所代表的真正意义。因此，为你所编写的程序代码提供充分的注释，是一种良好且必须的程序设计编写习惯。由于注释只是用于说明程序代码的相关文字，因此编译器不会去理会注释的内容，C#提供了与 C++ 语言相同的注释符号，例如，上述范例程序中，一开始的程序代码：

```
// SayHello.cs
//-------------------------------------------------------------
//  第 1 个 C# 应用程序
//-------------------------------------------------------------
```

这 4 行程序代码均以两条正斜线开始，将其标示为注释，C# 另外提供了一种标示整段的注释符号，可以将上式修改如下：

```
/* SayHello.cs
-------------------------------------------------------------
  第 1 个 C# 应用程序
-------------------------------------------------------------  */
```

这种形式的注释由“/*”以及“*/”两个符号所组成，落在这两个符号里的文字内容，将会被当作注释，而与程序代码的内容无关。

2. 类声明

SayHello 范例程序代码的注释后面，紧接着为类的声明。类在 C# 中被用来定义用户自行创建的引用类型 ，这听起来有点复杂，本章稍后的内容我们将会对其作进一步的说明，现在你只要了解 C# 应用程序均是由不同的类型所组成，而类则是一种特定类型即可。应

用程序的相关程序代码，都必须被写在类里面，而要创建一个类，首先必须声明此类，如范例中的程序代码：

```
class SayHello
```

这一行语句声明一个称为 SayHello 的新类，并且用括号“{ }”界定其程序代码的内容，在主控台输出“ Hello Welcome ”信息的相关程序代码均被写在这里面。大括号中创建了一个程序代码块，其中存在着一行以上的程序代码语句，形成了一个程序的逻辑单位，这些单位在程序执行的时候，根据程序员组织的方式进行运算，让你能够有效且清楚地执行程序运算。

3. 程序入口点 Main()

括号{ }里的程序代码，包含了一个函数 Main，其代表一个类所能提供的方法。方法定义了类所能执行的工作，也决定了类的特性。一个类可以有数种不同的方法，方法的详细内容我们将在下一章进行说明，一个名称为 Main 的函数，是一个特殊的方法，代表了程序的入口点，也就是程序一开始所执行的方法，整个应用程序只能有一个命名为 Main 的函数，以下为此范例的 Main 函数：

```
static void Main()
{
    System.Console.WriteLine(" Hello Welcome ") ;
}
```

程序本身由 Main 这个方法开始执行，因此，所有的 C# 应用程序均必须含有这个名称的函数，你必须将程序一开始所要执行的程序代码，放在这个函数里面，整个函数本身也是一个独立的程序代码块，用大括号所创建。

C# 是一种区分大小写的程序语言，因此，你必须在名称的第 1 个字母，明确地使用大写“M”为此方法的名称，而所有的程序代码，也需要注意其大小写之间的区分，而大部分情况下，编译器在程序编译期间，也会指出这一类的错误。

4. 关键字

关键字为程序语言预先定义的保留字，每一个关键字均有其所代表的特殊意义，上述程序代码的第 1 行定义了名称为 Main 的这个函数，其前使用了两个关键字。第一个关键字为 static，这个关键字指定其所定义的方法调用，不需要产生类的实例。C# 是一个面向对象的语言，所有方法函数均必须写在类里，并且通过所产生的类实例对象所调用。static 提供一种类似过程式语言的公共类成员的功能，你不需要产生类实例对象，便能够直接执行其中的函数，而由于 Main 是程序一开始执行的地方，此时还没有任何的对象被创建。因此，Main 总是被声明为 static，这其中的相关细节，我们在下一章作较为详细的说明。

第 2 个关键字 void 则指定此函数不返回任何值，同样，若是你所设计的方法有返回值，则必须以返回值的类型声明定义这个方法函数，在下一章会有相关内容的探讨。

5. 输出文字的代码

最后，我们来看看函数中真正将文字输出到主控台的程序代码，也就是以下这一行：

```
System.Console.WriteLine(" Hello Welcome ") ;
```

这一程序代码很简单，只是单纯地将“Hello Welcome”欢迎信息输出到主控台，并且以分号作为结束，C# 语言以分号作为每一段独立程序代码表达式的分隔符，一段独立的程序代码本身允许进行某种独立的程序运算。

6. 其他重要元素

这段程序代码里面，包含了几个重要的元素，列举说明如下：

- System 命名空间(namespace)，定义一个逻辑区域，用于分隔组织相关的类型以避免混淆，这个命名空间为.NET 所提供的核心命名空间，编写 .NET 应用程序所需的基础核心类均在这个命名空间中。
- Console 这是一个类库里所默认的类，提供简单的主控台文字输入/输出模式。
- WriteLine() 这是 Console 类提供的一个方法，这个方法将指定的信息输出到主控台。

1.4 命 名 空 间

C# 是一种面向对象程序语言，通过各种具有不同功能的对象组合成所需的应用程序。这些对象均由类所产生，使用 C# 语言设计应用程序，主要工作都集中在程序设计的设计类与使用类。而随着类的增加，你必须避免应用程序中相同类名称所造成的冲突，命名空间被设计用于解决这样的问题；例如，以下两个同样名称的类 FileIORW，分别被放置于不同的命名空间：

```
namespace  Console.IO
{
    class  FileIORW
    {
            进行主控台的 I/O   ……
    }
}

namespace  Windows.IO
{
    class  FileIORW
    {
            进行窗口程序的 I/O   ……
    }
}
```

当你编写程序代码使用 FileIORW 类时，若是直接使用其类名称，编译器并不知道你是需要 Console 版本的 FileIORW 类，还是 Windows 版本的 FileIORW，因此你必须使用“.”运算符连接命名空间与类名称，以区分不同版本的 FileIORW 类，如下：

```
Console.IO.FileIORW
```

以上代码表示使用 Console 版本的 FileIORW 类。接下来的一行代码则表示使用 Windows 版本的 FileIORW 类。

```
Windows.IO.FileIORW
```

利用这种方式避免名称的冲突虽然是个不错的主意，但是却非常容易造成冗长的程序代码，如果你编写大型的应用程序项目，这将是个很大的困扰。C# 使用 Using 关键字处理这样的问题，你可以在开始编写程序前，利用 Using 关键字指定所要引用的命名空间，而接下来的程序代码，便可以直接使用类名称。如上述的 Say Hello 范例，Console 本身为一个位于命名空间 System 的类，因此，你可以在程序一开始的时候，指定所要引用的命名空间。如下：

```
Using System
```

而在 SayHello 类的 Main 函数里，只需要直接写下 Console 的名称，引用其方法即可，而不需要写下完整的命名空间名称，此时的程序代码如下：

```
Console.WriteLine(" Hello Welcome ") ;
```

查看这段程序代码，你可以看到 System 命名空间的名称被拿掉了。

.NET Framework 提供了一组类库，其中包含了预先设计好的用于创建 .NET 平台应用程序所需的各种类，这些类按其特性及功能，被分类放置在不同的命名空间，供 C# 程序语言所使用，其中最重要的是 System 命名空间，这个命名空间包含了各种创建.NET 应用程序所需的核心基础类，本书往后的内容将会陆续介绍提供其他功能的命名空间。

1.5　主控台应用程序

当上述 SayHello 范例被执行的时候，出现一个 DOS 界面的应用程序执行画面，并且显示问候的信息，像这一类的应用程序，我们将其称为主控台应用程序 (Console Application)。主控台应用程序并不提供用户接口，由命令行或是 DOS 窗口与用户沟通，Console 类提供编写主控台应用程序所需的功能，本书在第 13 章和 14 章将会探讨如何编写窗口应用程序 (也就是一般应用程序的图形化用户接口) 及其相关的技术细节。为了与主控台应用程序作清楚的区别，稍后本章的内容，也将会对其作一简单介绍。

C# 是一种功能非常完整的面向对象语言，除了语言本身支持面向对象理论，并结合 .NET 类库所提供的功能，实现应用程序的操作。这些功能均被封装于不同的类，而其中的 Console 类，是.NET Framework 提供的类库中，我们所接触到的第 1 个类，它提供从主控台读、写字符的基本功能。本章与以后的几个章节，将大量使用这个类，进行程序范例的实现说明，并且示范各种 C# 应用程序的编写。

Console 类提供 WriteLine 方法，允许你将指定的字符输出到主控台，并且用 ReadLine 方法读取用户在主控台写入的数据，实现最基本的数据输入/输出 (I/O) 操作，以下的范例说明如何应用 Console 类，实现简单的主控台 IO 应用程序。

1.5.1　范例及其解析

范例 1.2　ConsoleIO

概述　示范主控台的基本输入与输出。