C# Language Programming

顾 洪 李 慧 主编

C#语言程序设计

東南大學出版社
Southeast University Press

C# Language Programming

顾 洪 李 慧 主编

C#语言程序设计

东南大学出版社
·南京·

内 容 简 介

C#是目前主流的程序设计语言之一，本书以 Microsoft Visual Studio 2005 为平台，系统地介绍了 C#的数据类型、表达式、流程控制、面向对象编程、接口技术、线程技术和文件操作等内容。

本书注重理论性与实用性的结合，内容循序渐进，示例代码简洁易懂，每章内容都有相应的小结和习题，便于读者总结和练习，并免费提供源代码。

本书适合作为高等院校相关专业 C#课程教材，也可供广大 C#开发用户参考。

图书在版编目(CIP)数据

C#语言程序设计/顾洪，李慧主编. —南京：东南大学出版社，2009.1

ISBN 978-7-5641-1469-5

Ⅰ.C… Ⅱ.①顾… ②李… Ⅲ.C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2008)第 188277 号

C#语言程序设计

出版发行 东南大学出版社(南京市四牌楼 2 号 邮编 210096)
电 话 (025) 83793191(发行)；57711295(传真)
出 版 人 江 汉
责任编辑 马子康
经 销 全国新华书店经销
印 刷 南京京新印刷厂
版 次 2009 年 1 月第 1 版 2009 年 1 月第 1 次印刷
开 本 787mm×1092mm 1/16
印 张 14.25
字 数 370 千字
印 数 1—4 000 册
书 号 ISBN 978-7-5641-1469-5/TP・243
定 价 26.00 元

前　言

自2000年6月微软公司推出Microsoft.NET以来，.NET技术和相应产品已被广泛应用在信息技术领域的各个方面，C#语言越来越成为主流的开发语言。学习面向对象Windows应用程序设计应采用广泛使用且适合教学的工具，而C#语言是为.NET框架量身定做的新一代面向对象的语言，用该语言开发的.NET系统的商用软件较多，通过学习，毕业后学生可具有较好的工作基础。

编者在从事C#语言的教学过程中，很难为学生找到一本合适的C#语言教材。主要是因为市场上此类教材一般是把C#语言、Windows应用程序开发和ASP.NET数据库开发三者结合在一起介绍，由于内容较多，学时有限，很难将这三部分内容讲深入、讲透彻，而且部分图书的内容难度较大，不适合初学者使用。为了解决这个问题，使学生尽可能地掌握面向对象编程的思想和语言开发工具，作者编写了本书。本书采用实例教学法，对于每个重要的知识点都有相应的示例。这些示例简单明了、针对性强，可以帮助读者快速理解所学的内容。本书适合作为高等院校相关专业及编程基础教学培训机构的教材，也可作为.NET开发人员的参考书。

全书按照教材体例编写，共12章，各章配有小结，总结相关知识点，章末配有习题以帮助读者巩固所学的内容。第1章概述，简略介绍.NET技术和C#语言的特点；第2章通过一个简单的实例介绍C#语言程序的基本结构；第3章介绍C#语言中所支持的数据类型以及变量的定义等内容；第4章介绍数据类型间的转换，包括显式转换和隐式转换；第5章介绍C#中与表达式相关的基本内容；第6章介绍C#中的流程控制，这是编程必需的内容；第7章介绍面向对象程序设计的基本概念类及其相关的内容；第8章介绍类中的重要内容：方法；第9章介绍类的重要特性：继承；第10章介绍体现组件编程思想的接口；第11章介绍线程；第12章介绍用C#语言对文件对象进行处理等内容。

本书所有示例都使用C#语言编写，以Visual Studio.NET 2005作为开发环境。本书配有书中的程序源代码和习题解答，需要这部分内容的读者可访问网站：http://www.sju.js.cn。

本书由三江学院顾洪老师和金陵科技学院李慧老师主编，吴德、曹阳、王志瑞、殷美琴、朱金锋、黄慧、张琳、许晓茹等老师参与编写，全书由顾洪统稿。在全书的编写过程中，得到了三江学院计算机科学与工程学院王芝庆院长和金陵科技学院信息技术学院张燕院长的大力支持和具体指导，在此一并表示感谢。

由于作者水平有限，不当之处在所难免，恳请读者批评指正。

编　者

2008年12月28日

目　录

第1章 概 述

1.1 .NET概述

Microsoft Visual Studio.NET是Microsoft公司为适应Internet高速发展的需要而推出的新的开发平台。2003年，Microsoft公司发布了VS.NET 2003，提供了在Windows操作系统下开发各类基于.NET框架1.1的全新的应用程序开发平台；2005年底，Microsoft公司又发布了基于.NET框架2.0的VS 2005开发平台，植入了适用于大型团队开发的各种优秀的复杂功能，并于2006年1月发布了VS 2005 Professional简体中文版。

.NET首先是一个开发平台，它定义了一种公用语言子集Common Language Subset (CLS)，这是一种为符合其规范的语言与类库之间提供无缝集成的混合语。.NET统一了编程类，提供了对下一代网络通信标准可扩展标记语言Extensible Markup Language(XML)的完全支持，使应用程序的开发变得更容易、更简单。Microsoft.NET计划还将实现人机交互方面的革命，微软将在其软件中添加手写和语音识别的功能，让人们能够与计算机进行更好的交流，并在此基础上继续扩展功能，增加对各种用户终端的支持能力。最为重要的是.NET将改变因特网的行为方式，软件将变成为服务。与Microsoft的其他产品一样，.NET与Windows平台紧密集成，并且与其他微软产品相比更进一步，由于其运行库已经与操作系统融合在了一起，因此从广义上把它称为一个运行库也不为过。

简而言之，.NET是一种面向网络、支持各种用户终端的开发平台环境，微软的宏伟目标是让Microsoft.NET彻底改变软件的开发方式、发行方式、使用方式等等。.NET的核心内容之一，就是要搭建第三代因特网平台，这个网络平台将解决网站之间的协同合作问题，从而最大限度地获取信息。在.NET平台上，不同网站通过相关的协定联系在一起，网站之间自动交流、协同工作，提供最全面的服务。

总之，.NET战略是一场软件革命，.NET对最终用户来说非常重要，因为计算机的功能将会得到大幅度提升，同时计算机操作也会变得非常简单。尤其是用户将完全摆脱人为的硬件束缚，可以自由冲浪于因特网的多维时空，自由访问、自由查看、自由使用自己的数据，不再束缚于便携式电脑的方寸空间，而是可通过任何桌面系统、任何便携式电脑、任何移动电话或PDA进行访问，并可对其进行跨应用程序的集成。.NET对开发人员来说也十分重要，因为它不但会改变开发人员开发应用程序的方式，而且使得开发人员能创建出全新的各种应用程序，大幅提高软件生产率。

1.2 .NET与C#

1.2.1 支持多种编程语言的.NET结构框架

多年以前，当微软的组件对象模型 Component Object Model(COM)尚未推出时，软件的复用性对于开发人员仅仅是一种美好的憧憬，成千上万的程序员为了处理通信、接口和不同语言间的冲突而通宵达旦地艰辛劳动，但却收效甚微。COM 的出现改变了这一切，通过将组件改变为通用集成型的构件，开发人员正逐渐地从过去的繁复编程事务中解脱出来，可以选择自己最得心应手的编程语言进行编程。然而，软件组件与应用程序之间的联合仍然是松散的，不同的编程语言与开发平台限制了部件间的互用性，其结果是产生了日益庞大的应用程序与不断升级的软硬件系统。举个很简单的例子，只用五行 C 语言代码就能编写出的一个简单程序，若使用 COM 来编写，可能需要几百行代码，COM 在带来巨大价值的同时，也大大增加了开发费用，而 .NET Framework 的出现使得一切问题都迎刃而解。实际上，在 .NET Framework 中，所有的编程语言，从相对简单的 JScript 到复杂的 C++语言，一律是等同的。Framework 框架是开发人员对编程语言命令集的称呼，.Net 框架的意义就在于用统一的命令集支持任何的编程语言，.NET 将使编程人员梦想的语言互用性变成为近在眼前的现实。想想看，一个在 Visual Basic(VB)中定义的类能够在另一种与它完全不同的语言环境中使用、调试甚至继承，这将给编程带来多大的便利。.NET 框架是 .NET 平台的基础架构，其强大功能来自于公共语言运行时 Common Language Runtime(CLR)和类库，包括将 Windows Forms ADO.NET 和 ASP.NET 紧密结合在一起，这提供了不同系统之间交叉与综合的解决方案和服务。.NET 框架创造了一个完全可操控的、安全的和特性丰富的应用执行环境，这不但使得应用程序的开发与发布更加简单，并且成就了众多种类语言间的无缝集成。

1.2.2 面向.NET的全新开发工具C#

当今，C 和 C++一直是最有生命力的程序设计语言，这两种语言为程序员提供了丰富的功能，高度的灵活性和强大的底层控制能力，而这一切都不得不在效率上作出不同程度的牺牲。如果用户使用过包括 C 和 C++在内的多种程序设计语言，相信会深刻体会到它们之间的区别。比如与 Visual Basic 相比，Visual C++程序员为实现同样的功能就需要更长的开发周期。由于 C 和 C++既为用户带来了高度的灵活性，又使用户必须要忍受学习的艰苦和开发的长期性，许多 C 和 C++程序员一直在寻求一种新的语言，以期在开发能力和效率之间取得更好的平衡。今天，人们改进开发出了许多语言以提高软件生产率，但这些或多或少是以牺牲 C 和 C++程序员所需要的灵活性为代价，这样的解决方案限制了程序员能力的发挥，且它们不能很好地与原有的系统兼容；更令人头痛的是，它们并不总是与当前的 Web 应用结合得很好。理想的解决方案是将快速的应用开发与对底层平台所有功能的访问紧密结合在一起，程序员们需要一种环境，它与 Web 标准完全同步，并且具备与现存应用间方便地进行集成的能力；除此之外，它允许程序员在需要时使用底层代码。

针对该问题，微软的解决方案是设计出一种称为C#的程序语言。C#是一种现代的、面向对象的程序开发语言，它使得程序员能够在新的微软.NET平台上快速开发种类丰富的应用程序，而.NET平台提供了大量的工具和服务，能够最大限度地发掘和使用计算及通信能力。由于一流的面向对象的设计，因此从构建组件形式的高层商业对象到构造系统级应用程序，都会发现C#是最合适的选择。使用C#语言设计的组件能够用于Web服务，这样通过Internet可以被运行于任何操作系统上任何编程语言所调用。不但如此，C#为C++程序员提供快捷的开发方式，它并没有丢掉C和C++的基本特征。由于C#与C和C++有着很大程度上的相似性，熟悉C和C++的开发人员很快就能精通C#。

1.3 C#语言的特点

C#在带来对应用程序的快速开发能力的同时，并没有牺牲C与C++程序员所关心的各种特性，它忠实地继承了C和C++的优点。即使是一位新手，C#也不会给用户带来任何其他的麻烦，快速应用程序开发Rapid Application Development(RAD)的思想与简洁的语法将使用户迅速成为一名熟练的开发人员。

正如前文所述，C#是专门为.NET应用而开发出的语言，这从根本上保证了C#与.NET框架的完美结合，在.NET运行库的支持下，.NET框架的各种优点在C#中表现得淋漓尽致。下面先来看看C#的一些突出的特点，相信在以后的学习过程中，将会深深体会到#(SHARP)的真正含义。

1.3.1 简洁的语法

C#用真正的关键字换掉了那些把活动模板库Active Template Library(ALT)和COM搞得乱糟糟的伪关键字，如OLE_COLOR、BOOL、VARIANT_BOOL、DISPID_XXXXX等等，每种C#类型在.NET类库中都有新名字，需要理解的仅仅是名字嵌套而已。语法中的冗余是C++中常见的问题，比如const和#define各种各样的字符类型等等，C#对此进行了简化，只保留了常见的形式，而将别的冗余形式从它的语法结构中清除了出去。

1.3.2 面向对象设计

C#具有面向对象语言所应有的一切特性：封装、继承与多态，这并不出奇；然而，通过精心地面向对象设计，从高级商业对象到系统级应用，C#是建造广泛组件的绝对选择。在C#的类型系统中，每种类型都可以看作一个对象。C#只允许单继承，即一个类不会有多个基类，从而避免了类型定义的混乱。C#中没有全局函数，没有全局变量，也没有全局常数，一切的一切都必须封装在一个类之中，这样代码将具有更好的可读性，并且减少了发生命名冲突的可能。整个C#的类模型是建立在.NET虚拟对象系统Visual Object System(VOS)的基础之上，其对象模型是.NET基础架构的一部分，而不再是其本身的组成成分，在下面将会谈到，这样做的另一个好处是兼容性强。借助于从VB中得来的丰富的RAD经验，C#具备了良好的开发环境，结合自身强大的面向对象功能，C#使得开发人员的生产效率得到极大地提高。

1.3.3 与 Web 的紧密结合

.NET 中新的应用程序开发模型意味着越来越多的解决方案需要与 Web 标准相统一，例如超文本标记语言(Hypertext Markup Language，HTML)和可扩展标记语言(Extensible Markable Language，XML)，由于历史的原因，现存的一些开发工具不能与 Web 紧密地结合，简单对象访问协议(Simple Object Access Protocol，SOAP)的使用使得 C# 克服了这一缺陷，大规模、深层次的分布式开发从此成为可能。由于有了 Web 服务框架的帮助，对程序员来说，网络服务看起来就像是 C# 的本地对象，程序员们能够利用他们已有的面向对象的知识与技巧，开发 Web 服务；仅需要使用简单的 C# 语言结构，C# 组件将能够方便地为 Web 服务，并允许它们通过 Internet 被运行在任何操作系统上的任何语言所调用。举个例子，XML 已经成为网络中数据结构传送的标准，为了提高效率，C# 允许直接将 XML 数据映射成为结构，这样就可以有效地处理各种数据。

1.3.4 安全性与错误处理

语言的安全性与错误处理能力是衡量一种语言是否优秀的重要依据，任何人都会犯错误，即使是最熟练的程序员也不例外。忘记变量的初始化、对不属于自己管理范围的内存空间进行修改，这些错误常常产生难以预见的后果，一旦这样的软件被投入使用，寻找与改正这些简单错误的代价将会是让人无法承受的。C# 的先进设计思想可以消除软件开发中的许多常见错误，并提供了包括类型安全在内的完整的安全性能。为了减少开发中的错误，C# 会帮助开发者通过更少的代码完成相同的功能，这不但减轻了编程人员的工作量，同时更有效地避免了错误发生。

.NET 运行库提供了代码访问安全特性，它允许管理员和用户根据代码的 ID 来配置安全等级。在缺省情况下，从 Internet 和 Intranet 下载的代码都不允许访问任何本地文件和资源。比方说，一个在网络上的共享目录中运行的程序，如果它要访问本地的一些资源，那么异常将被触发，若拷贝到本地硬盘上运行，则一切正常。内存管理中的垃圾收集机制减轻了开发人员对内存管理的负担，.NET 平台提供的垃圾收集器(Garbage Collection，GC)负责资源的释放与对象撤销时的内存清理工作。

C# 中变量是类型安全的。不能使用未初始化的变量，对象的成员变量由编译器负责将其置为零，当局部变量未经初始化而被使用时，编译器将作出提醒。C# 不支持不安全的指向，不能将整数指向引用类型。

1.3.5 版本处理技术

C# 提供内置的版本支持来减少开发费用。使用 C# 将会使开发人员更加方便地开发和维护各种商业应用。升级软件系统中的组件、模块是一件容易产生错误的工作，在代码修改过程中，可能对现存的软件产生影响，并很有可能导致程序的崩溃，为了帮助开发人员处理这些问题，C# 在语言中内置了版本控制功能。例如，函数重载必须被显式地申明，而不能像在 C++ 或 Java 中经常发生的那样被不经意地运行，这可以防止代码级错误和保留版本化的特性。另一个相关的特性是接口和接口继承的支持，这些特性可以保证复杂的软件被方便地开发和升级。

1.3.6 灵活性和兼容性

在简化语法的同时，C#并没有失去灵活性，尽管它不是一种无限制的语言，比如它不能用来开发硬件驱动程序，在默认的状态下没有指针等等。但是，在学习过程中将发现，它仍然是那样的灵巧。如果需要，C#允许将某些类或者类的某些方法申明为非安全的，这样一来将能够使用指针、结构和静态数组，并且调用这些非安全的代码，不会带来任何其他的问题。此外，它还另外提供了一个手段来模拟指针的功能：delegates 代理。再举一个例子，C#不支持类的多继承，但是通过对接口的继承，将获得这一功能。正是由于其灵活性，C#允许与C风格的、需要传递指针型参数的API进行交互操作，DLL的任何入口点都可以在程序中进行访问。C#遵守.NET公用语言规范(CLS)，从而保证了C#组件与其他语言组件间的互操作性，元数据(Metadata)概念的引入既保证了兼容性，又实现了类型安全。

1.4 公用语言运行时环境与公用语言规范

了解了.NET的结构之后，来看看.NET利用其结构创造的运行环境。公用语言运行时环境，是C#及其支持的.NET平台开发工具的运行基础。具体来说，它为用户的应用提供了以下益处：跨语言集成、跨语言异常处理、内存管理自动化、强化的安全措施、版本处理技术、组件交互的简化模型。

1.4.1 理解 CLR

.NET提供了一个运行时环境，叫做公用语言运行时，它管理着代码的执行，并使得开发过程变得更加简单，这是一种可操控的执行环境。在这种环境下的跨语言集成、跨语言异常处理、增强的安全性、版本处理与开发支持、简单的组件交互模型以及调试服务等技术的实现，需要运行时环境能够向可操控代码提供服务。语言编译器需要产生一种元数据，它将提供在使用语言中的类型、成员、引用的信息，元数据与代码一起存储，每个可加载的CLR映像均包含了元数据，运行时环境使用元数据定位并载入类，在内存中展开对象实例、解决方法调用、产生本地代码、强制执行安全性，并建立运行时环境的边界。运行时环境自动处理对象的展开与引用，当对象不再被使用时运行时环境负责它们的释放，被运行时环境这样的生命期管理的对象称为可操控代码。自动内存管理消除了内存溢出，同时也解决了其他一些常见的语法错误，如果代码是可操控的，仍然可以在需要的时候使用非可控代码，或者在.NET应用中同时使用可控与非可控代码。CLR使设计跨语言的组件与应用变得更加容易，以使不同语言设计的对象能够彼此间进行通信，并且它们的行为能够紧密地综合与协调。举个例子，定义了一个类，可以在另一种不同的语言中从该类中派生一个类或者调用它其中的一个方法，也可以向另一种语言中类的方法传递该类的一个实例。这种跨语言的集成之所以可能，是因为以运行时为目标的语言编译器与工具使用一种运行时所定义的公用类型系统，它们通过遵守运行时的规则、公用语言规范来定义新的类型生成，使用、保持并绑定类型。作为元数据的一部分，所有可控组件携带了关于它们所依赖的组件与资源的信息，运行时环境使用这些信息来保证组件或应用具有需要的所有东西的特定版

本，其结果是代码将不会因为版本冲突而崩溃，注册信息与状态数据不再保存在难以建立与维护的注册表中，所定义的类型及附属信息作为元数据被保存，这使得复制与移动组件的复杂程度得到降低。

1.4.2 可操控执行的含义

前面的叙述中，多次提到了可操控这一概念，这意味着它指向的对象在执行过程中完全被运行时环境所控制，在执行过程中，运行时环境提供以下服务：自动内存管理、调试支持、增强的安全性及与非可操控代码的互操作性。例如，COM 组件在可控执行过程中的第一步是选择源代码的生成工具，如果希望应用 CLR 提供的优势，就必须使用一种或多种以运行时为目标的语言编译器；例如 VB、C#、VC 的编译器；或者一种第三方编译器，如 PERL 或 COBOL 编译器。由于运行时是一种多语言执行环境，它支持众多的数据类型和语言特性，使用的语言编译器决定将使用运行时的哪一部分功能子集，所以在代码中使用的语法由编译器决定，而不是运行时环境。如果组件需要被其他语言的组件完全使用，那么必须在组件的输出类型中使用 CLR 所要求的语言特征；当完成并编译代码时，编译器将它转换为微软中间语言 Microsoft Intermediate Language(MSIL)，同时产生元数据。当要执行代码时，这种中间语言被即时 Just In Time(JIT)编译器编译成为本地代码。如下：JIT 编译器将在编译过程中对中间语言进行类型检查，一旦失败，在代码执行中将会触发异常。

1.4.3 CLR 的突出特色

跨语言集成的能力

CLR 包含了一个丰富的语言特性集，保证了它与各种程序设计语言的兼容性。

内存管理自动化

在执行过程中管理应用程序的资源是一项单调而困难的工作，它会将程序员的注意力从本应解决的问题中引开，而垃圾收集机制完全解决了程序员在编程过程中头痛的问题，跟踪内存的使用，并知道何时将它们释放。在面向对象的环境中，每种类型都标识了对应用有用的某种资源，为了使用这些资源，需要为类型分配内存。在应用中访问一种资源要通过以下步骤：

- 为类型分配内存；
- 初始化内存，设置资源的初始状态并使其可用；
- 通过访问该类型的实例成员来访问资源；
- 卸下将被清除的资源状态；
- 释放内存。

这一看似简单的过程在实际的编程中是产生程序错误的主要来源之一，更可怕的是，内存中的错误往往导致不可预见的结果。

CLR 要求所有的资源从可操控的堆(一种内存结构)中分配。当一个进程被初始化后，CLR 保留了一个未被分配的地址空间，这一区域叫做可操控堆，在堆中保持了指向下一个将被分配给对象的堆地址的指针 NEXT。初始状态下，该指针保留地址空间的基地址。一个应用使用新的操作产生对象时，首先检查新对象需要的字节大小是否会超出保留空间，如果对象大小合适，NEXT 指针将指向堆中的这个对象，该对象的构造器被调用，新的操作返

回对象的地址。地址空间不够大时，堆将发现这一点，通过将新对象的大小与 NEXT 指针相加，并与堆的大小进行比较，调用垃圾收集器。在这里 CLR 引入了代的概念。代指堆中对象产生的先后，这样垃圾收集器在将发生溢出时回收属于特定的代的对象，而不是回收堆中的所有对象。

即时编译

在各种语言的编译器对源代码进行编译之后，在 CLR 环境中产生的是中间代码，出于兼容性与跨语言集成的考虑，其内容虽然有效，但在转化为本地代码之前它本身是不可执行的，这就是 JIT 编译器需要完成的工作。

这里需要说明一个问题：为什么要即时编译，而不是一次性地将中间代码文件进行编译？原因在于效率。在大型的应用中，很少会用到程序的全部功能，这种边执行边编译的措施比一次性的完全编译效率更高。

解决版本与发布问题

在当前以组件为基础的系统中，开发人员和用户对于软件版本和发布中存在的问题已经十分熟悉了：当安装一个新的应用之后，很可能发现原本正常的某个应用程序奇怪地停止了工作。绝大多数开发人员将时间花在了确保所有注册表入口的一致性，以便激活 COM 类上，这就是所谓的 DLL 地狱。.NET 平台通过使用集合来解决这一问题，在这里集合是一个专有名词，指类型与资源的发布单元，在很大程度上它等同于 DLL。正像.NET 用元数据描述类型一样，它也用元数据描述包含类型的集合。通常说来，集合由四个部分组成：集合的元数据、集合的内部清单、元数据描述的类型、实现类型的中间语言代码和一组资源。在一个集合中，以上四个部分并不是都必须存在，但是必须包含类型或资源，这样集合才有意义。

在.NET 中，一个基本的设计方针是使用孤立的组件。一个孤立的集合的含义是一个集合只能被一个应用所访问；在一台机器上，它不被多个应用共享，也不会受其他应用程序对系统的更改的影响。孤立赋予了开发人员在自己开发的程序中对代码的完全控制权，任何共享代码都需要被明确地标识。但.NET 框架也支持共享集合的概念，一个共享集合指在一台机器上被多个应用共享的集合，共享集合需要遵循严格地命名规定。.NET 应用程序间的共享代码是明确定义的，共享集合需要一些额外的规则来避免今天遇到的共享冲突问题、共享代码必须有一个全局唯一的名称，系统必须提供名称保护，且每当引用共享集合时，CLR 将对版本信息进行检查。.NET 框架允许应用或管理员在明确说明的版本政策下重写集合的版本信息。

1.4.4 公用语言规范

使被不同语言的编译器所编译的对象能够相互理解的唯一方法是所有在互操作过程中涉及的数据类型和语言特性对所有的语言来说是公共的。为了这个目的，公用语言运行时环境标识了一组语言特征的集合，称为公用语言规范(CLS)。如果组件在应用程序接口(Application Program Interface，API)中仅使用 CLS 的特征语言，包括子类，那么该组件能够被任何支持 CLS 的语言所编译的组件访问。所有支持 CLS 并仅使用 CLS 中的语言特征的组件，称为符合 CLS 的组件。设计公用语言规范时遇到的一个最主要的挑战是选择适当的语言特性子集的大小，它应具有完全的表达能力，又应足够小，使得所有的语言都能够容