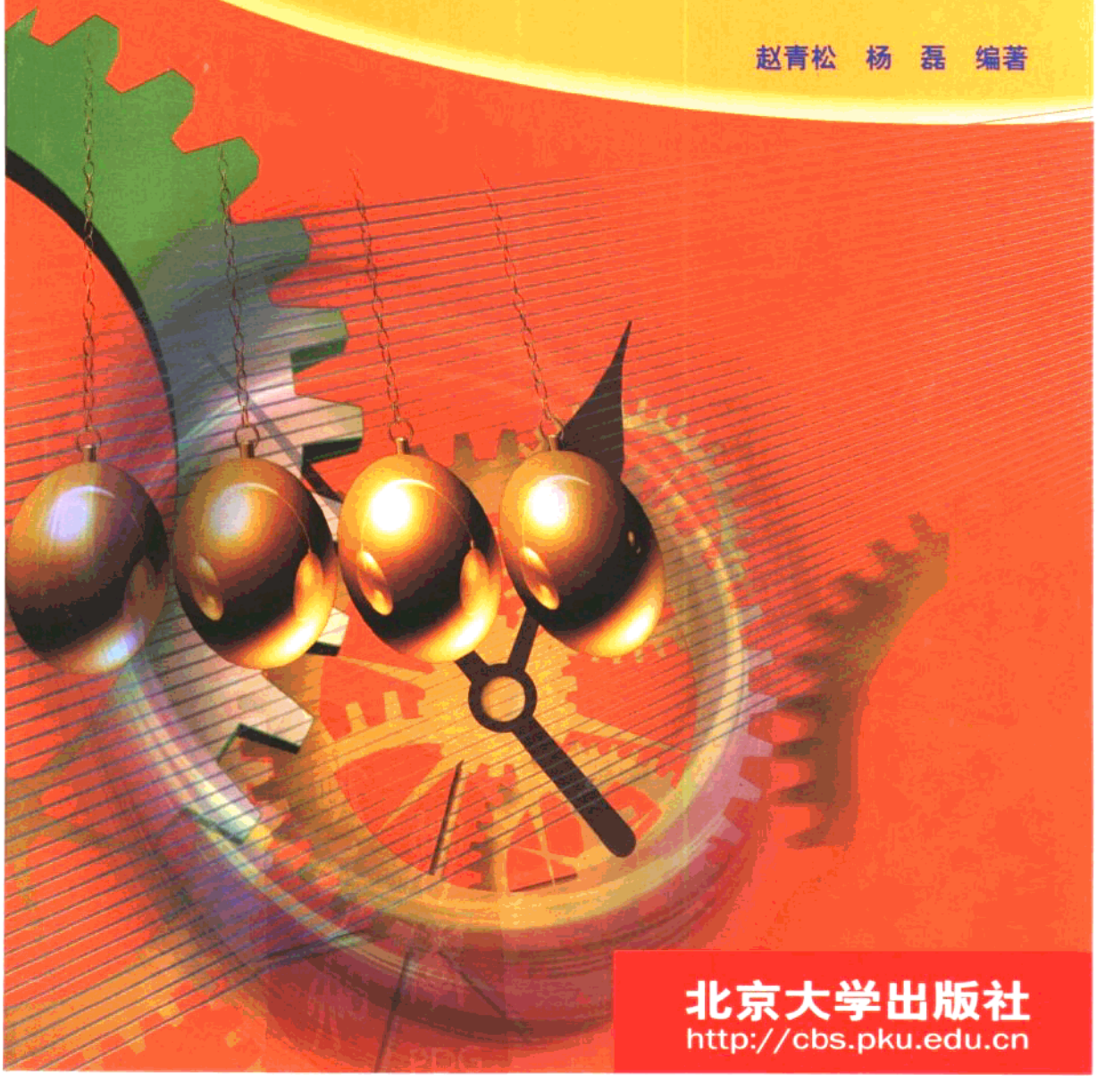
从C++、Java到C#

赵青松　杨　磊　编著



北京大学出版社
http://cbs.pku.edu.cn

前　言

在过去的20年中，C和C++已成为开发商用和企业软件时使用最广泛的编程语言之一。这两种语言为开发者提供了大量细致灵活的控制，这种灵活性是以生产的成本作为代价的。对C和C++程序员来说，理想的解决方法应该是快速地开发与访问所有潜在平台的能力相结合。开发环境应该完全与新的Web标准同步并容易与现存的应用系统集成，同时，C和C++程序员还希望能够在必要的时候在底层编写代码。

微软为了解决上述问题，提出了一种叫C#的语言（英文中读作：C sharp）。C#是一种现代的、面向对象的语言，它使开发人员能够在微软新的.NET平台上快速建立广泛的应用，其提供的工具和服务能充分发掘系统的计算和通信能力。

C#是一种简单可靠的、高级的、面向对象的程序设计语言。C#是在C和C++语言的基础之上开发的，所以它也是C语言家族中的一员，它和这两种语言有非常类似的结构和风格。

另外，C#和Java语言还有着千丝万缕的联系。微软产品经理Tony Goodhew说，C#是Java语言与C++语言的混合体。Java与C++用于撰写视窗应用软件，适合软件开发人员采用。同时Goodhew指出，C#虽然包含多种Java程序语言受欢迎的功能，但不是为了与Java竞争。但IBM软件部总经理Steve Mills却认为："C#是改了名称的Java，微软的Visual Basic不像Java那样横跨多重使用者平台、需要修改，C#可作为Java的替代选择。"

接触过这两种语言的程序员可以发现，C#和Java具有很多相似之处。如C#与Java都对指针的使用提出了严格的要求：在C++中，定义了指针之后，你可以自由地把它指向任何一个类型，包括做一些相当愚昧的事情，比如将一个整型指针指向一个双精度型指针，只要内存支持这一操作，它就会凑合着工作，而往往是这种情况导致了程序的错误。为了避免上述问题，C#实施了最严格的类型安全措施来保护它自身及垃圾收集器。

另外，自动内存管理机制是Java的特点之一：C++语言中的内存管理一直是程序员最头痛的问题，程序员往往忘了给变量分配内存，不过大多数程序员是在分配了内存后，如不再需要时却没有及时释放，甚至是根本就没有释放，结果就发生了内存泄露。而C#提供的内存碎片自动收集机制能自动收集程序中释放的内存，这就大大减轻了程序员自己管理内存的负担。

当然，C#语言也提供了Java具备、但C或C++语言欠缺的功能，例如更强的安全功能，以及清理应用软件所占电脑存储器的"垃圾收集"功能等。

有人曾经预言，"C#的推出将使Java的地位受到威胁。"在C#尚未充分展示魅力的今天，这样的预言能否实现，还是一个未知之谜。而对C++、Java及C#进行全面的比较，无疑是一件非常有意义的事情。

本书讲解了这三种语言各自的特点和它们之间的一些联系，从而使读者能够对当今最流行的计算机程序设计语言有一个全面的了解。

本书由赵青松、杨磊编著。另外，谢松县、郭建龙、云勇、刘恒、常慧、李斯、方元昆、简宁、管明治、陆安、吴光、柴米河、沙岭奇、李景、田豪、张量等也参加了本书的编写工作。

鉴于编者水平有限，错误之处在所难免，欢迎广大读者批评指正。如需进一步交流，请发邮件至 wdj1976@sina.com。

编　者

2002 年 4 月

责任编辑：杨锡林
封面设计：项　尧

从 C++、Java 到 C#

C++、Java 和 C# 是三种功能强大的的高级程序设计语言，在全球范围内得到广大程序员的喜爱和关注。作为面向对象和程序设计语言，它们彼此之间有着千丝万缕的联系，但是又有着各自的特点。本书就是在介绍三者各自特点的基础之上，比较和分析了它们之间的联系。

本书内容由浅入深，围绕面向对象程序设计语言的特点，通过大量的实例讲解，使读者既能了解和掌握这三种语言的基本知识，又能通过它们之间的对比和分析，进而对面向对象程序设计有一个更高层次的、全面的认识。

本书可作为 C++、Java 和 C# 初学者的入门教程，也可作为广大编程人员的参考书。

ISBN 7-301-05200-6

9 787301 052006 >

ISBN 7-301-05200-6/TP · 0597

定价：28 元

北京大学出版社

地址：北京市海淀区中关村北京大学校内
邮编：100871
电话：(010)62765013　（编辑部）
　　(010)62750672　（发行部）
E-mail:xxjs@pup.pku.edu.cn
http://cbs.pku.edu.cn

从 C++、Java 到 C#
北京大学出版社

目　录

第1章 概 述

C++和Java是当前最为流行的编程语言，而C#则是微软新近推出的.NET编程语言。对这三种语言进行比较和鉴别，不仅有助于编程人员对它们的特点和应用有所把握，而且可以根据它们各自的优劣在编程时有所选择，更好地实现自身的需求。

1.1 技术背景

1.1.1 C++的技术背景

C语言是一种被广泛应用的程序设计语言，它除了具有规模小、速度快、结构化、程序化、结构灵活等特点之外，也存在着一些不足：

（1）非强制类型，这既是它的长处，也是它的一个不足。从技术上来讲，类型机制反映了一种程序设计语言对强制使用变量类型的严格程度。在C中可以将一个整数赋给一个字符变量，反之亦可。这就意味着编程人员应当正确地管理程序中的变量。对有经验的编程人员来说，这没有什么问题，但新手应加以注意，因为这样做可能带来副作用。

（2）缺乏运行时刻检查，在C中，因为缺乏运行时刻检查机制，会引起很多神秘和短暂的问题，它们常常不易被编程人员发觉。

（3）传统程序设计技术，也就是面向过程的结构化程序设计技术。在这种设计方式下程序通常包含一个主函数，以及一个或者若干个由主函数调用的函数（子例程）。这种程序设计方法是自顶向下的。在这种模式下，代码和数据是分离的，过程定义要对数据进行处理，但两者不会融为一体，这就给程序的维护带来了不便。当必须增加和删减程序代码时，往往要牵动整个程序。这种方式下的开发时间和调试时间都比较长。

C++语言基于C，是C的一个超级集合，保留了C的灵活性，并且具有强有力处理硬软件接口和低层系统程序设计的能力；C++保留了C语言的紧凑性和强有力的表达式。更为重要的是C++提供了支持面向对象程序设计和高层问题抽象的平台，在支持面向对象程序设计方面，C++比Ada更前进了一步。在简洁性和模块化方面，C++类似于Modula-2，而且保留了C的高效率和简洁性。

C++是一种混合性语言。它支持用面向对象的程序设计来求解问题，使之产生这个问题全新的面向对象解。实际上，C++语言表现出具有过程程序设计方法和新的面向对象设计方法。这种在C++中的双重性，对于C++初学者提出了一种特殊的挑战，即不仅要学习新语言，而且要学习新的问题求解的方法和掌握新的思维方式。

C++是由BCPL和Simula 67的某些灵感而产生的。它从Simula引进了子类（导出类）

和虚拟函数，借鉴了 AIgol 68 操作符重载能力和灵活地把说明紧挨着应用程序首次使用的地方的能力。像其他现代程序设计语言一样，C++语言进化和改良了我们所熟知的高级程序设计语言的某些最佳特征。当然，最接近于 C++的还是 C。

1980 年，C++由贝尔实验室的 Bjarne Stroustrup 创建。最初的动机是打算在效率上改进 Simula 67 语言并采用复杂事件驱动，当时的 C＋＋称为带类的 C（C-with-classes），缺少操作符重载、引用和虚拟函数等许多细节。

1983 年，C++首次推广到外界，到 1987 年夏天，C++仍然在演变。AT&T 决定性地提交出与 C 兼容的 C++，这样就保留了完整的 C 的几百万行代码及广泛的 C 的库和 C 的工具。鼓励 C 程序员学习 C++而不放弃他已经工作多年的过程程序设计语言。这对 C 程序员来说是个大好消息：当从 C 前进到 C++时，他们仍没有放弃在过去工作的领域，从而使语言在发展中保持了稳定性。事实上，C 语言本身影响了 C++的发展。ANSI C 标准草案中包含了某些 C++的关键特征，诸如函数原型，1990 年通过的 ISO C 中也是如此。

C++语言的标准化工作是从 1989 年开始的，现在已建立了灵活的面向对象语言的经验库，并且牢固地与多值继承、封装技术和其他重要领域建立了联系，而且取得新的突破性进展。

C++语言支持大型软件系统的开发。在语言中增加了强类型检测以及与 C 语言比较，减少了在 C++软件系统的开发性错误，并提供了对多用户程序设计。

1.1.2　Java 的技术背景

Java 诞生于 1991 年，它是由 Sun Microsystems 公司的一个开发小组开发出来的，该小组隶属于一个最初名为 Green 的项目组。该项目组试图开发一种能够应用于消费性电子产品工业中的独立于平台的软件技术。这个小组创建的语言起初叫做 Oak。

与此同时，一股新的潮流席卷美国，它就是“Web 漫游”。World Wide Web（环球网，WWW），这个应用于 Internet 的数百万个相互链接的 HTML 文档上的名字，突然在大众中变得流行起来。这是因为引入了一种由 NCSA 开发的名为 Mosaic 的图形 Web 浏览器。该浏览器简化了 Web 上的浏览操作，方法是通过把文本和图形组合到一个界面中，无需用户学会许多令人困惑的 UNIX 和 DOS 命令。使用 Mosaic 使得在 Web 上的浏览操作变得非常容易。

从 1994 年起，Oak 技术开始应用于 Web 上。1994 年，两名 Sun 公司的开发人员创建了 HotJava 的第一个版本，当时它被命名为 WebRunner，是一个用于 Web 的图形浏览器，至今仍在使用。该浏览器完全由 Oak 语言编写，后来改称 Java。不久之后，Java 编译器由最初的 C 语言代码改用 Java 语言进行了重写，这证明了 Java 可以作为一种应用程序语言而有效地使用。在 1995 年 5 月的 SunWorld 95 会议期间，Sun 正式推出了 Java。

Web 漫游在数百万的计算机用户间变得极为流行。然而，在 Java 出现之前，Internet 上信息的内容都是一系列乏味的 HTML 文档。Web 用户如饥似渴地期盼着一种交互式的、只需考虑所使用的软硬件平台即可执行的、能够访问多机种网络而不会把病毒传播给多机种网络计算机的应用程序。Java 能够创建这类应用程序。

1.1.3　C#的技术背景

C 和 C++作为开发商用和企业软件时使用最广泛的编程语言之一，为开发者提供了大

量的细致灵活的控制，但这种灵活性是以生产的成本作为代价的。为此微软提出了一种叫 C# 的语言。C#是一种现代的、面向对象的语言，它使开发人员能够在微软新的 .NET 平台上快速建立广泛的应用，其提供的工具和服务能充分发掘系统的计算和通能力。

C#语言将作为 Microsoft Visual Studio 7.0 的一部分而推出，同时这个开发环境还支持 Visual Basic、Visual C++以及脚本语言 VBScript、Jscript 等。微软的 Next Generation Windows Services（NWGS）平台提供了一个执行引擎和一个丰富的类库来支持上述语言的解释和执行。对于 C#开发人员来说，这就意味着即使 C#是一种新的语言，它也可以使用 Visual Basic、Visual C++所使用的类库，而自身并不用包含类库。

因为其优良的面向对象设计，在构建从高级业务对象到系统级应用的各种不同组件时，C# 是一个首要的选择。使用简易的 C# 语言构造，组件可以被转换为 Web 服务，从而允许从运行在任何操作系统上的任何语言中跨越 Internet 调用它们。

不仅如此，C#的设计为 C++程序员带来了快速的开发能力，而不用牺牲 C++已有的功能和控制能力。通过这种继承，C#高度地保持了与 C 和 C++的一致。开发者只要熟悉 C 和 C++语言就可以快速地掌握 C#，并写出更多的 C#应用程序。

1.2 语 言 特 性

1.2.1 C++的语言特性

C++是从 C 语言发展起来的，在其发展过程中就包含了对 C 的一些特性的改进：

（1）注解。C++中引入了一种新的注解到行末的分界符“//”。

（2）枚举名。一个枚举的名即一个类型名，这就简化了原先枚举类型的表示符，不必在枚举类型名前加一个 enum。

（3）结构名或者类名。结构或者类的名即是类型名，这是 C++发展而来的，不需在一个结构或者类名前加限制符 struct 或者 class。

（4）分程序声明。C++允许将声明放在分程序内和代码语句之后，这就允许编程人员将标识符在离它的首次使用点最近的地方加以说明。它甚至允许在循环控制变量结构中的形式定义被声明。

（5）匿名联合。不具有名字的联合可以在定义一个变量或域的任何地方定义。利用这一特性，可以使一个结构的两个或多个域之间共享内存，从而节省内存。

（6）显式类型转换。可以用一个预定义类型或者用户定义的类型的名作为一个函数，将数据从一种类型转换至另一种类型。在特定的情况下，这种显式类型转换机制可以用来作为强制类型转换的替代。

（7）函数重载。在 C++中，利用说明符重载后，函数就可以使用相同的名字，并可根据参数的个数和类型来区别每个重载的函数。

（8）缺省的函数参数值。在 C++中，可以为函数的尾部参数赋予缺省值。有了这一特性后，可以用少于形式参数个数的实参来调用函数，所缺少的尾部参数即取它们的缺省值。

（9）inline 说明符。利用 inline 说明符可以指示编译程序在一个函数被调用处对该函数

进行扩展替换。

除此之外，C++在面向对象方面有了很大的增强：

（1）类构造和数据封装性。类构造是面向对象程序设计的基本工具。类定义可以封装所有的数据声明、初始值以及操作集，以实现数据抽象。对象可以被声明为属于某个给定的类，还可以向对象发送消息。此外，一个指定类的对象可以包含其自身的、关于该类的数据表示的私有部分和共有部分。

（2）struct 类。C++中的 struct 是类定义的子集，没有私有或保护部分。这种子类可以包含数据和函数。

（3）构造方法和析构方法。构造方法和析构方法用来对某一指定类的一个对象内定义的数据进行初始化。当一个对象被声明时，初始化方法就被激活。当从声明该对象所在作用域内退出后，析构方法就自动对该对象占用的内存进行释放。

（4）消息。对象是面向程序设计的基本成分。对对象的操作是通过向它们发送消息来完成的。对象也就是声明为某一个给定类的变量，只需利用与调用函数类似的机制，即可向对象发送消息。可以向一个对象发送的消息集在有关该对象的类描述中指定。在响应一条消息时，对象要根据消息的性质来确定采用哪一种动作比较合适。

（5）友元。数据隐藏和数据封装意味着不可以直接访问对象内部的结构。对于类之外的任何函数来说，类的私有部分通常是完全不可见的。但是，C++也确实允许方法或类之外的其他函数被声明为类的一个友元。友元关系透过类的保护墙，允许访问类的私有数据和方法。

（6）运算符重载。在 C++中，程序人员可以取编译程序员提供的预定义运算符和函数集，或取用户定义的运算符和函数，并赋予它们多重含义。

（7）派生类。派生类可以视为指定类的子类，从而形成了一个抽象层次。派生类对象通常继承其父类的全部或部分方法，还可以将这些继承来的方法与自身所包含的新方法综合起来。所有的子类对象都包含来自父类中的数据域，以及它们自己的私有数据。

（8）使用虚函数的多态性。多态性涉及一种父类和子类的树结构。树中的每个子类可以接受一条或多条相同的消息。在这棵树中，当某个类的一个对象接受一条消息时，该对象就要确定对于指定子类的一个对象来说适合于该消息的特定应用。

1.2.2　Java 的语言特性

Java 是一种优秀的编程语言。它具有以下特点：

1. 简单

Java 最初是为对家用电器进行集成控制而设计的一种语言，因此它必须简单明了。Java 语言的简单性主要体现在以下三个方面：

（1）Java 的风格类似于 C++，因此，C++程序员可以很快就掌握 Java 编程技术。

（2）Java 摒弃了 C++中容易引发程序错误的地方，如指针和内存管理。

（3）Java 提供了丰富的类库。

2. 面向对象

面向对象可以说是 Java 最重要的特性。Java 语言的设计完全是面向对象的，它不支持类似 C 语言那样的面向过程的程序设计技术。Java 支持静态和动态风格的代码继承及重用。

3．分布式

Java 包括一个支持 HTTP 和 FTP 等基于 TCP/IP 协议的子库。因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全相同。

4．健壮

Java 致力于检查程序在编译和运行时的错误。类型检查帮助检查出许多开发早期出现的错误。Java 自己操纵内存减少内存出错的可能性。Java 还实现了真数组，避免了覆盖数据的可能。

5．结构中立

另外，为了建立 Java 作为网络的一个整体，Java 将它的程序编译成一种结构中立的中间文件格式。只要有 Java 运行系统的机器都能执行这种中间代码。Java 源程序被编译成一种高层次的与机器无关的 byte-code 格式语言，这种语言被设计在虚拟机上运行，由机器相关的运行调试器实现执行。

6．安全

Java 的安全性可从两个方面得到保证。一方面，在 Java 语言里，像指针和释放内存等 C++功能被删除，避免了非法内存操作。另一方面，当 Java 用来创建浏览器时，语言功能和浏览器本身提供的功能结合起来，使它更安全。

7．高性能

如果解释器速度不慢，Java 可以在运行时直接将目标代码翻译成机器指令。Sun 用直接解释器一秒钟内可调用 300,000 个过程。翻译目标代码的速度与 C/C++的性能没什么区别。

8．多线程

Java 的多线程功能使得在一个程序里可同时执行多个小任务。因为 Java 实现了多线程技术，所以比 C 和 C++更健壮。多线程带来的更大的好处是更好的交互性能和实时控制性能。

1.2.3　C#的语言特性

1．生产力和安全

新的 Web 经济要求所有商业部门比从前更快地响应竞争的威胁。要求开发者在更短的时间内生产发行更多的程序版本，而不是单一纪念物似的版本。C# 的设计考虑到了这些因素，它的设计帮助开发者用更少的代码行和更少的出错机会做更多的事情。

（1）遵循新的 Web 设计标准

新的应用程序开发模型意味着越来越多的解决方案需要使用新的 Web 标准，如：HTML、XML 和 SOAP（Simple Object Access Protocol）。现存的开发工具都是在 Internet 之前或之初开发的。因此，它们总是不能很好地适应新的 Web 技术。

C#程序员可以用一个扩展的框架来构建微软.NET 平台上的应用。C#包括内建的支持使任何组件转换为一个能在 Internet 上运行任何平台上的任何应用中被调用的服务。

对程序员来说，更出色的是这个 Web 服务框架能使现存的 Web 服务看起来像本地的 C#对象，因此允许开发者权衡利用它们已有的面向对象的编程技能与现存的 Web 服务。

当然，还有更多的特点使得 C#成为主要的 Internet 开发工具。例如，XML 是新出现的在 Internet 中传递结构化数据的方法。这种数据集通常很小。为了提高性能，C#允许 XML 数据被直接映射到一个结构数据类型而不是一个类。在数据量不大时，这是一种更有效的处

理方式。

（2）消除重要的编程错误

即便是专家级的 C++程序员都避免不了最简单的错误，如忘记初始化一个变量，这些简单的错误常常导致不可预见的问题，它们长时间隐蔽，不易发现。一旦程序投入生产运行后，要排除哪怕是最简单的编程错误，都要付出昂贵的代价。

C#现代的设计排除了大多数普通的 C++编程错误。例如：

- 垃圾清理减轻了程序员自己管理内存的负担。
- 在 C#中的变量是自动被环境初始化的。
- 变量类型是安全的。

（3）依赖内建的转换支持降低开发成本

更新软件组件是一个容易出错的任务。修订代码版本无意中可能会改动程序的语义。为了帮助程序员解决这个问题，C#语言中包含了转换支持。例如，与 C++和 Java 不同，方法重载必须显式进行；这有助于防止代码错误和保留转换的灵活性。有关的特点还有本地的接口和接口继承支持。这些特点进化并发展了复杂的框架。

总之，这些特点使一个工程的后继版本的开发方法更加健壮，同时为成功的应用降低了整体的开发成本。

2. 功能、表现和灵活性

随着公司制作商业计划的水平越来越高，抽象的商业过程与实际软件实现之间的紧密连接已成一种必然趋势。但大多数语言工具没有一种简单易行的方式来链接业务逻辑和代码。例如，开发者可能使用代码注释来说明哪个类构成主要的抽象业务对象。

C#语言允许分类，扩展能应用于任何对象的元数据。一个工程的构建者能够定义特定领域的属性并将它们应用到任何语言的类元素、接口等等。然后开发者编程测试每个元素的属性。这就简化了自动化工具的编写，而自动化工具将保证每个类或者接口被正确表示为特定抽象商业对象，或简化了创建基于对象特定属性的过程。紧密连接定制元数据和程序代码有助于加强意料的程序行为与实际的实现之间的联系。

可管理的类型安全环境对大多数企业应用是适合的。但是现实世界中的经验告诉我们，有些应用延续需要“本地”代码，其原因要么是因为性能问题，要么是因为与现存的应用程序接口问题。这样的情况强迫开发者使用 C++来开发，即使他们喜欢使用生产性更好的开发环境。

C#通过下面的方式解决了这个问题：包含组件对象模型（COM）的本地支持和基于 Windows APIs 的支持。

在 C# 中，每一个对象自动地成为一个 COM 对象。开发者不再需要显式地实现 IUnknown 和其他 COM 接口，而是将它们内置了。同样，C#编程能在本地使用现存的 COM 对象，与它们用什么语言创建的无关。

对于需要这种特性的开发者来说，C# 包含了一个特别的特性：那就是一个程序可以调用任何本地 API。在一个特别标记的代码块中，允许开发者使用指针和传统的 C/C++特点，如自己管理内存和指针算法。

相对于其他开发环境，这是一个突出的优点。它意味着 C# 程序员能建立他们自己现存的大量 C 和 C++代码，而不用丢弃它们。

COM 支持和本地 API 访问支持，目的是向开发者提供了重要的能力和控制，而不用离

开 C#环境。

具体来说则包括如下一些特点：

（1）现代性

你在学习 C#上所付出的努力将是一项巨大的投资，因为 C#被设计成为开发.NET 应用的首选语言。你会发现许多在 C++中必须由你自己来实现或者干脆没有的特征，都成为基础 C#的一部分了。

小数类型对于企业级编程语言来说是很受欢迎的一个附加类型。你可以使用一个新的 DECIMAL 数据类型来进行货币计算。如果你不喜欢这个简单类型，可以很容易地为你的应用特别设计一个新的类型。

在 C#中，指针不再是你的编程武器了，所以你也就不必负责整个内存的管理了。.NET 平台环境提供了自动内存管理机制负责你的 C#程序的内存管理工作。

对于 C++程序员来说，异常处理是 C#的一个主要特性，但这决不是什么新鲜事。然而，和 C++不同的是，异常处理是跨语言的（运行环境的另一特点）。在 C#之前，你必须对付离奇的 HRESULTs，现在不必要了，稳健的基于异常处理的错误处理机制能做好这些。

（2）面向对象

C#作为一种新的语言理所当然地支持面相对象的所有关键概念：封装、继承和多态性。整个 C#的类模型是建立在.NET 虚拟对象系统（VOS，Virtual Object System）之上的。对象模型是基础构架的一部分，而不是编程语言的一部分。

在面向对象的程序设计中，没有全局函数、变量或者常数。这些必须被封装在一个类中，或者作为一个实例成员（通过类的一个实例对象来访问），或者作为一个静态成员（通过类型来访问），这会使你的 C#代码具有更好的可读性，并且减少了发生命名冲突的可能性。

在类中定义的方法缺省情况下不是虚拟的（不能被派生类所覆盖），这一做法的要点是可以去掉另一个产生错误的来源——对方法的错误覆盖。如果一个方法可以被覆盖，那么它必须是有一个显式 virtual 修饰符。这样不但可以减少虚函数表的长度，还能保证正确的版本处理行为。

对于 C++程序员而言，已经熟悉了使用访问权限修饰符为类的成员指定不同的访问级别。C#也支持私有（private）、保护（protected）和公共（public）访问权限修饰符，而且增加了第四个：internal。在实际的程序设计中，很多情况下只需要从一个类派生，从多个基类派生所带来的问题比这种做法所能解决的问题更多，这就是 C#只允许一个基类的原因，但是完全可以用接口来实现同样的多重继承的功能。

（3）类型安全性

在 C++中，定义了指针之后，你可以自由地把它指向任何一个类型，包括做一些相当愚昧的事情，比如将一个整型指针指向一个双精度型指针，只要内存支持这一操作，它就会凑合着工作，而往往是这种情况导致了程序的错误。

为了避免上述问题，C#实施了最严格的类型安全来保护它自身及垃圾收集器。在 C#中，必须遵守关于变量的一些规定：

不能使用未初始化的变量。对于对象的成员变量，编译器负责把它们置零。局部变量需要你自己处理。如果使用了未初始化的变量，编译器就会提醒你。这样做的好处是：你可以避免使用了一个未被初始化的变量带来的严重后果。

C#不支持不安全的指向。不能将整数指向引用类型，并且 C#会时时验证指向的有效性。