高等院校信息技术规划教材

计算机专业英语

（第2版）

高凯 张永强 编著

清华大学出版社

高等院校信息技术规划教材

计算机专业英语

（第2版）

高凯 张永强 编著

清华大学出版社
北京

内 容 简 介

本书精选 C#、Java 以及 Android 移动开发平台、分布式系统、网络信息检索与搜索引擎技术、开源信息检索工具、人工智能、绿色计算、云计算、无线传感网、物联网、虚拟化技术、统一通信、下一代网络服务技术、ACM 图灵奖获奖科学家事迹简介、IT 职业规划等最新 IT 科技及相关内容，选材广泛，内容丰富，配有单词注解、难句翻译、相关专业知识说明、背景知识介绍等。读者在学习本书的同时，还可以了解科技英语中常用的语法，掌握常用的计算机专业英语词汇和表达方式，提高计算机专业知识的阅读和理解能力。

本书选材新颖、图文并茂，具有系统性、先进性、技术前瞻性，适合作为高等院校学生和专业技术人员学习计算机专业英语的教材及工具书，并可作为信息技术领域的技术人员和管理人员的参考用书。

图书在版编目(CIP)数据

计算机专业英语/高凯，张永强编著. —2 版. —北京：清华大学出版社，2015
高等院校信息技术规划教材
ISBN 978-7-302-40037-0

Ⅰ. ①计… Ⅱ. ①高… ②张… Ⅲ. ①电子计算机－英语－高等学校－教材 Ⅳ. ①H31

中国版本图书馆 CIP 数据核字(2015)第 077237 号

责任编辑：焦　虹　李　晔
封面设计：傅瑞学
责任校对：白　蕾
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954

印 装 者：三河市少明印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：19　　**字　　数**：436 千字
版　　次：2011 年 9 月第 1 版　2015 年 6 月第 2 版　　**印　　次**：2015 年 6 月第1 次印刷
印　　数：1～2000
定　　价：34.90 元

产品编号：062600-01

前言 foreword

随着我国信息化进程的加快，IT 业正在向国际化迈进。作为计算机和 IT 业的行业性国际通用语言，英语在业界发展和学术交流活动中的不可或缺作用已经被越来越多的科研人员、大中专院校师生所认同。无论是学习计算机技术还是使用计算机软硬件产品，都离不开计算机专业英语这个工具。在未来的激烈竞争中，既精通专业技术又具备较高专业英语水平的复合型人才，必将具有更大的优势。

计算机专业英语的学习有着和普通英语不一样的地方，专业英语突出表现在长句多、被动语句多、祈使语句多、专业术语多、合成新词多、一词多义现象较普遍等方面，而部分专业技术人员和大学生在学习计算机专业英语时不能很好地掌握学习方法和技巧，缺乏科技英语的阅读理解、翻译及写作能力，对新知识背景缺乏了解，导致学习效果不理想。如何更好地讲授计算机专业英语课程，让学生在学习到最新的专业知识的同时，更快地提高专业英语水平，以便能达到更好的教学效果，是教学中必须解决的问题。

在河北省重点学科“计算机软件与理论”以及河北省电子信息类本科教学高地重点专业建设项目的支持下，我们编写了本书。它是编者在精选英文时文和经典原版教材的基础上，结合教学和科研实践编写而成的，同时加大了最新的计算机领域热点知识和技术介绍，意在培养和提高学生用英语交流沟通的能力。本书每章由多篇关联文章组成，选材广泛，内容丰富，主要涉及 C#、Java 以及 Android 移动开发平台、分布式系统、网络信息检索与搜索引擎技术、开源信息检索工具、人工智能、绿色计算、云计算、无线传感网、物联网、虚拟化技术、统一通信、下一代网络服务技术、ACM 图灵奖获奖科学家事迹简介、IT 职业规划等内容。为了面向具有不同专业背景的读者，本书用批注的方式，穿插介绍一些相关的专业术语。同时，对较生僻的单词进行了注解，对重点句和难句进行了翻译，在章节最后还给出了相关专业背景知识说明等。这样，读者在阅读时可以基本

不用查阅字典就能了解最新的计算机发展技术及应用前景,而以中文形式表述的专业知识背景介绍则能帮助读者了解相关专业知识背景,启发读者理解篇章内容。本书章后附有针对课文内容的思考练习题和参考文献,帮助读者进一步理解课文内容。另外,书中大量新知识和实用技能的介绍,不仅有助于调动学生的学习热情,焕发学生求知欲,引导学生自发地去学习相关专业知识,还能通过对国外经典教材中相应内容的讲解,使学生学习计算机方面的常用词汇以及科技英语中常用的语法现象,掌握常用的计算机专业英语词汇和英语表达方式,能够分析专业英语的长句结构,从而准确把握文意,提高大篇幅英语文章的阅读速度,加强对计算机专业知识的阅读和理解能力。

本书选材多为相关领域的经典原版资料,图文并茂,具有系统性、先进性、技术前瞻性等特点。每章按照 Section、Background Knowledge、Thinking and Exercising、Reference、Some Translations 的顺序排列,生僻词用脚注标于相应的页面下方,针对部分词汇的背景知识简介则以批注的形式列在相应的 Section 后面,部分难句翻译用尾注列于章尾。本书适合普通高校、实践及工程类院校学生在学习专业英语时选用,是高等院校学生和在职专业人员学习计算机专业英语的理想教材及工具书,也是 IT 领域技术人员和管理人员的自修参考用书。通过对本书的学习,读者不但能从中学到专业的英语知识,增加计算机英语词汇量,提高针对科技英语的翻译能力,还能了解相关的计算机背景知识,了解计算机及相关科技发展的轨迹,使学生在提高专业英语水平的同时,全方位拓展知识面,为进一步的科研和工作打下良好的基础。

本书由高凯、张永强主编,编著工作分工:高凯统稿,提出写作大纲并撰写第 1~5 章和第 13 章;张永强撰写了其余章节;张晴、高莘等协助完成书稿审校工作。

在本书的编写过程中,得到河北省重点学科"计算机软件与理论"以及河北省电子信息类本科教学高地重点专业建设项目的支持。同时,张冬雯、郭立炜、王俊社、周万珍、杨奎河、张坤、张晓明、仇晶、许云峰、高国江、乔世权、李媚等对本书的编写工作提出很多好的意见和建议,国外众多的计算机技术的经典教材、研究成果和相关网站也为本书提供了参考。本书的顺利完成得益于大量的相关工作及研究成果,其中一部分背景资料来源于百度百科知识,在此谨向提供这些作品的网站、作者和科研工作人员(特别是那些由于篇幅所限未能在参考文献中提及的作者)表示谢意;同时,对那些为本书提供帮助的老师、同仁和课题组成员致以诚挚的谢意和崇高的敬意。

由于我们的学识、水平均有限,书中存在不妥之处在所难免,恳请广大读者批评指正。

编 者

2015 年 2 月

目录 Contents

chapter 1

Programming and Mobile Development Platform[1]

In this chapter, we will present the coding knowledge and the popular mobile development platform. C# and the . NET framework will be present first, then the **comparability**[2] among C# and Visual Basic . NET, Java, C++ will be proposed. Java overview, including its history, its class libraries, its typical environment, will be present then. Android is a new open source software toolkit for mobile phones that was created by Google and the **Open Handset Alliance**. In a few years, it's expected to be found in millions of cell phones and other mobile devices, making Android a major platform for application developers.

Section 1 C# and the .NET Framework

The goal of C# is to provide a simple, safe, modern, object-oriented, Internet-centric, high performance language for . NET development. C# is a new language, but it draws on the lessons learned over the past three decades. In much the way that you can see in young children the features and personalities of their parents and grandparents, you can easily see in C# the influence of Java, C++, Visual Basic, and other languages.

The focus of this section is the C# language and its use as a tool for programming on the . NET platform. You learn C# specifically to create . NET applications. This section does not consider C# in a vacuum but places the language firmly in the context of Microsoft's . NET platform and in the development of desktop and Internet applications.

1. Overview of . NET

When Microsoft announced C# in July 2000, its **unveiling**[3] was part of a much larger

---

[1] 本章英文内容系选自参考文献[Liberty, 2001]、[Deltel, 2009]、[Burnette, 2009]中的内容,有部分删改

[2] comparability *n*. 相似性

[3] unveiling: *n*. 揭开面纱, 显露

event: the **announcement**❶ of the . NET platform. **The . NET platform is, in essence, a new development framework that provides a fresh application programming interface (API) to the services and APIs of classic Windows operating systems, especially Windows 2000, while bringing together a number of disparate technologies that emerged from Microsoft during the late 1990's**[i]. Among the latter are COM + component services, the ASP web development framework, a commitment to XML and object-oriented design, support for new web services protocols such as **SOAP**, **WSDL**, and **UDDI**, and a focus on the Internet, all integrated within the **DNA**❷ architecture.

Microsoft says it is devoting 80% of its research and development budget to . NET and its associated technologies. The results of this commitment to date are **impressive**❸. For one thing, the scope of . NET is huge. The platform consists of four separate product groups:

(1) A set of languages, including C# and Visual Basic . NET; a set of development tools, including Visual Studio . NET; a comprehensive class library for building web services and web and Windows applications; as well as the *Common Language Runtime* (CLR) to execute objects built within this framework.

(2) A set of . NET Enterprise Servers, formerly known as SQL Server 2000, Exchange 2000, BizTalk 2000, and so on, that provide specialized functionality for relational data storage, email, B2B commerce, etc.

(3) An offering of commercial web services, recently announced as Project Hailstorm; for a fee, developers can use these services in building applications that require knowledge of user identity, etc.

(4) New . NET-enabled non-PC devices, from cell phones to game boxes.

2. The . NET Framework

Microsoft . NET supports not only language independence, but also language integration. This means that you can inherit from classes, catch exceptions, and take advantage of **polymorphism**❹ across different languages. The . NET Framework makes this possible with a **specification**❺ called the Common Type System (CTS) that all . NET components must obey. For example, everything in . NET is an object of a specific class that derives from the root class called System. Object. The CTS supports the general concept of classes, interfaces, **delegates**❻ (which support callbacks), reference types, and value types.

---

❶ announcement: *n.* 宣告

❷ DNA: 此处不是指脱氧核糖核酸, 是 Distributed N Architexture 的缩写, 微软. NET 架构的前身

❸ impressive: *adj.* 印象深刻的

❹ polymorphism: *n.* 多态性

❺ specification: *n.* 规范, 说明书

❻ delegate: *n.* 委托

Additionally, . NET includes a Common Language Specification (**CLS**), which provides a series of basic rules that are required for language integration. The CLS determines the minimum requirements for being a . NET language. Compilers that **conform**❶ to the CLS create objects that can **interoperate**❷ with one another. The entire Framework Class Library (FCL) can be used by any language that conforms to the CLS.

The . NET Framework sits on top of the operating system, which can be any **flavor**❸ of Windows, and consists of a number of components. Currently, the . NET Framework consists of:

(1) Four official languages: C#, VB . NET, Managed C++, and JScript . NET.

(2) The Common Language Runtime (**CLR**), an object-oriented platform for Windows and web development that all these languages share.

(3) A number of related class libraries, collectively known as the Framework Class Library (**FCL**).

Figure 1-1 breaks down the . NET Framework into its system architectural components.

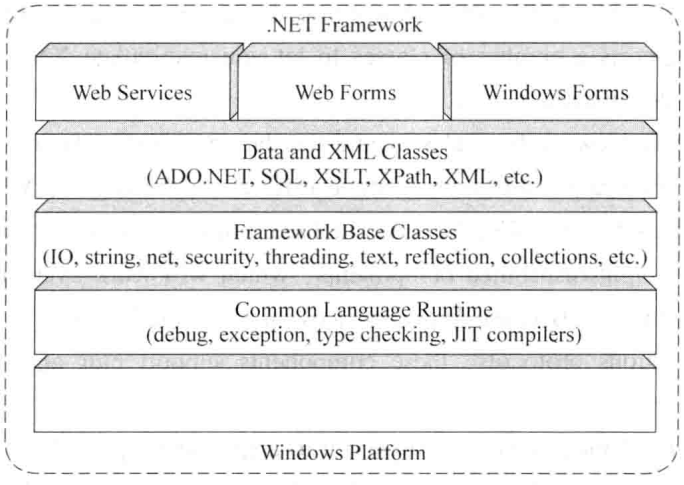

Figure 1-1 NET Framework architecture

The most important component of the . NET Framework is the CLR, which provides the environment in which programs are executed. The CLR includes a virtual machine, **analogous**❹ in many ways to the Java virtual machine. At a high level, the CLR activates objects, performs security checks on them, lays them out in memory, executes them, and garbage-collects them. (The Common Type System is also part of the CLR.)

In Figure 1, the layer on top of the CLR is a set of framework base classes, followed by

---

❶ conform: *v.* 遵守

❷ interoperate: *v.* 互操作

❸ flavor: *n.* 特色

❹ analogous: *adj.* 类似的，相似的

an additional layer of data and XML classes, plus another layer of classes intended for Web Services, Web Forms, and Windows Forms. Collectively, these classes are known as the Framework Class Library (FCL), one of the largest class libraries in history and one that provides an object-oriented API to all the functionality that the . NET platform **encapsulates**❶. With more than 5,000 classes, the FCL facilitates rapid development of desktop, client/server, and other web services and applications.

The set of framework base classes, the lowest level of the FCL, is similar to the set of classes in Java. These classes support **rudimentary**❷ input and output, string manipulation, security management, network communication, thread management, text manipulation, reflection and collections functionality, etc.

Above this level is a tier of classes that extend the base classes to support data management and XML manipulation. The data classes support persistent management of data that is maintained on databases. These classes include the Structured Query Language (SQL) classes to let you manipulate persistent data stores through a standard SQL interface. Additionally, a set of classes called ADO. NET allows you to manipulate persistent data. The . NET Framework also supports a number of classes to let you manipulate XML data and perform XML searching and translations.

Extending the framework base classes and the data and XML classes is a tier of classes geared toward building applications using three different technologies: Web Services, Web Forms, and Windows Forms. Web Services include a number of classes that support the development of lightweight distributed components, which will work even in the face of firewalls and **NAT** software. Because Web Services employ standard HTTP and SOAP as underlying communications protocols, these components support plug-and-play across cyberspace.

Web Forms and Windows Forms allow you to apply rapid application development techniques to building web and windows applications. It usually means simply draging and dropping controls onto your form, double-clicking a control, and writing the code to respond to the associated event.

3. C# Language

The C# language is **disarmingly simple**❸, with only about 80 keywords and a dozen built-in datatypes, but C# is highly expressive when it comes to implementing modern programming concepts. C# includes all the support for structured, component-based, object-oriented programming that one expects of a modern language built on the shoulders of C++ and

---

❶ encapsulates: *n.* 胶囊,封装

❷ rudimentary: *adj.* 根本的

❸ disarmingly simple: 使人倍感轻松的

Java.

The C# language was developed by a small team led by two distinguished Microsoft engineers, Anders Hejlsberg and Scott Wiltamuth. Hejlsberg is also known for creating Turbo Pascal, a popular language for PC programming, and for leading the team that designed Borland Delphi, one of the first successful integrated development environments for client/server programming.

At the heart of any object-oriented language is its support for defining and working with classes. Classes define new types, allowing you to extend the language to better model the problem you are trying to solve. C# contains keywords for declaring new classes and their methods and properties, and for implementing **encapsulation**, **inheritance**, **and polymorphism**❶, the three **pillars**❷ of object-oriented programming.

In C# everything **pertaining to**❸ a class declaration is found in the declaration itself. C# class definitions do not require separate header files or Interface Definition Language (IDL) files. Moreover, C# supports a new XML style of inline documentation that greatly simplifies the creation of online and print reference documentation for an application.

C# also supports interfaces, a means of making a contract with a class for services that the interface **stipulates**❹. In C#, a class can inherit from only a single parent, but a class can implement multiple interfaces. When it implements an interface, a C# class in effect **promises**❺ to provide the functionality the interface specifies.

C# also provides support for structs, a concept whose meaning has changed significantly from C++. In C#, a struct is a restricted, lightweight type that, when **instantiated**❻, makes fewer demands on the operating system and on memory than a conventional class does. A struct can't inherit from a class or be inherited from, but a struct can implement an interface.

C# provides component-oriented features, such as properties, events, and declarative constructs (called attributes). Component-oriented programming is supported by the CLR's support for storing metadata with the code for the class. The metadata describes the class, including its methods and properties, as well as its security needs and other attributes, such as whether it can be serialized; the code contains the logic necessary to carry out its functions. A compiled class is thus a self-contained unit; therefore, a hosting environment that knows how to read a class' metadata and code needs no other information to make use of it. Using C# and the CLR, it is possible to add custom metadata to a class by creating custom attributes. Likewise, it is possible to read class metadata using CLR types that support reflection.

---

❶ encapsulation, inheritance, and polymorphism: 封装、继承和多态
❷ pillar: *n.* 柱子, 支柱
❸ pertaining to: 与……有关
❹ stipulate: *v.* 规定, 保证
❺ promise: *v.* 承诺
❻ instantiated: *v.* 实例化

An **assembly**❶ is a collection of files that appear to the programmer to be a single dynamic link library (DLL) or executable (EXE). In .NET, an assembly is the basic unit of reuse, versioning, security, and deployment. The CLR provides a number of classes for manipulating assemblies.

4. C# Versus Visual Basic .NET

The **premise**❷ of the .NET Framework is that all languages are created equal. To **paraphrase**❸ George Orwell, however, some languages are more equal than others. C# is an excellent language for .NET development. You will find it is an extremely **versatile**❹, robust and well-designed language. It is also currently the language most often used in articles and tutorials about .NET programming.

It is likely that many VB programmers will choose to learn C#, rather than upgrading their skills to VB .NET. This would not be surprising because the transition from VB6 to VB .NET is, **arguably**❺, nearly as difficult as from VB6 to C#—and, whether it's fair or not, historically, C-family programmers have had higher earning potential than VB programmers. As a practical matter, VB programmers have never gotten the respect or **compensation**❻ they deserve, and C# offers a wonderful chance to make a potentially **lucrative**❼ transition.

5. C# Versus Java

Java programmers may look at C# with a mixture of **trepidation, glee, and resentment**❽. It has been suggested that C# is somehow a "rip-off" of Java. I won't comment on the religious war between Microsoft and the "anyone but Microsoft" crowd except to acknowledge that C# certainly learned a great deal from Java. But then Java learned a great deal from C++, which owed its syntax to C, which in turn was built on lessons learned in other languages. We all stand on the shoulders of giants.

C# offers an easy transition for Java programmers; the syntax is very similar and the semantics are familiar and comfortable. Java programmers will probably want to focus on the differences between Java and C# in order to use the C# language effectively.

6. C# versus C++

While it is possible to program in .NET with C++, it isn't easy or natural. **Frankly**❾,

---

❶ assembly: *n.* 程序集
❷ premise: *n.* 前提
❸ paraphrase: *v.* 解释
❹ versatile: *adj.* 通用的
❺ arguably: *adv.* 可论证地
❻ compensation: *n.* 补偿
❼ lucrative: *adj.* 有利的
❽ trepidation, glee, and resentment: 颤抖、高兴和怨恨
❾ frankly: *adv.* 坦白地

having worked for ten years as a C++ programmer and written a dozen books on the subject, I'd rather have my teeth drilled than work with managed C++. Perhaps it is just that C# is so much friendlier. In any case, once I saw C# I never looked back.

批 注

Open Handset Alliance：开放手机联盟包括手机制造商、手机芯片厂商和移动运营商等，是谷歌公司于2007年11月宣布组建的一个全球性组织，它支持谷歌发布的手机操作系统或应用软件，共同开发基于Android的开放源代码的移动系统。

SOAP：SOAP指简单对象访问协议(Simple Object Access Protocol)，是一种轻量的、简单的、基于XML的协议，允许Java对象和COM对象在分布式的、分散的、基于Web的环境中彼此通话。更一般地，SOAP允许任何类型的对象(或代码)在任何平台上，以任何一种语言相互通信。

WSDL：WSDL是Web Services Description Language的缩写，是用来描述Web服务和说明如何与Web服务通信的XML语言，用于描述Web Service及其函数、参数和返回值。一些开发工具既能根据Web Service生成WSDL文档，又能导入WSDL文档，生成调用相应Web Service代码。

UDDI：UDDI是Universal Description Discovery and Integration的缩写，意为统一描述、发现和集成的协议，它使得商业实体能够彼此发现，定义它们怎样在Internet上互相作用并在一个全球的注册体系架构中共享信息。

CLS：公共语言规范CLS和通用类型系统一起确保语言的互操作性。CLS是一个最低标准集，所有面向.NET的编译器都必须支持它。

CLR：CLR是公共语言运行时，用于把.NET的语言翻译为机器可以执行的语言，负责资源管理(内存分配和垃圾收集)，并保证应用和底层操作系统之间必要的分离。

FCL：FCL是.NET框架类库，不论在C#、J#、VB.NET或其他.NET开发语言中使用的.NET提供的类都是FCL中的类，也就是说，.NET的开发其实是和语言无关的。不同语言编写的程序由于使用同一类库(等于被翻译成了同一种中间语言Common Intermediate Language，即CIL)，使得各种不同语言写的程序集可以通用。

NAT：即Network Address Translation，是一个IETF(Internet Engineering Task Force，Internet工程任务组)的标准，允许一个整体机构以一个公用IP地址出现在Internet上，是一种把内部私有网络IP地址翻译成合法网络IP地址的技术。即在局域网内部使用内部地址，而当内部节点要与外部网络进行通信时就在网关将内部地址替换成公用地址，从而在外部公网上正常使用。NAT可以使多台计算机共享Internet连接，这一功能很好地解决了公共IP地址紧缺的问题。

Section 2 Java Overview

1. History of Java

Perhaps the microprocessor revolution's most important contribution to date is that it made possible the development of personal computers, which now number in the hundreds of millions worldwide[ii]. Personal computers have had a profound impact on people and the way organizations conduct and manage their business.

Many people believe that the next major area in which microprocessors will have a pro-