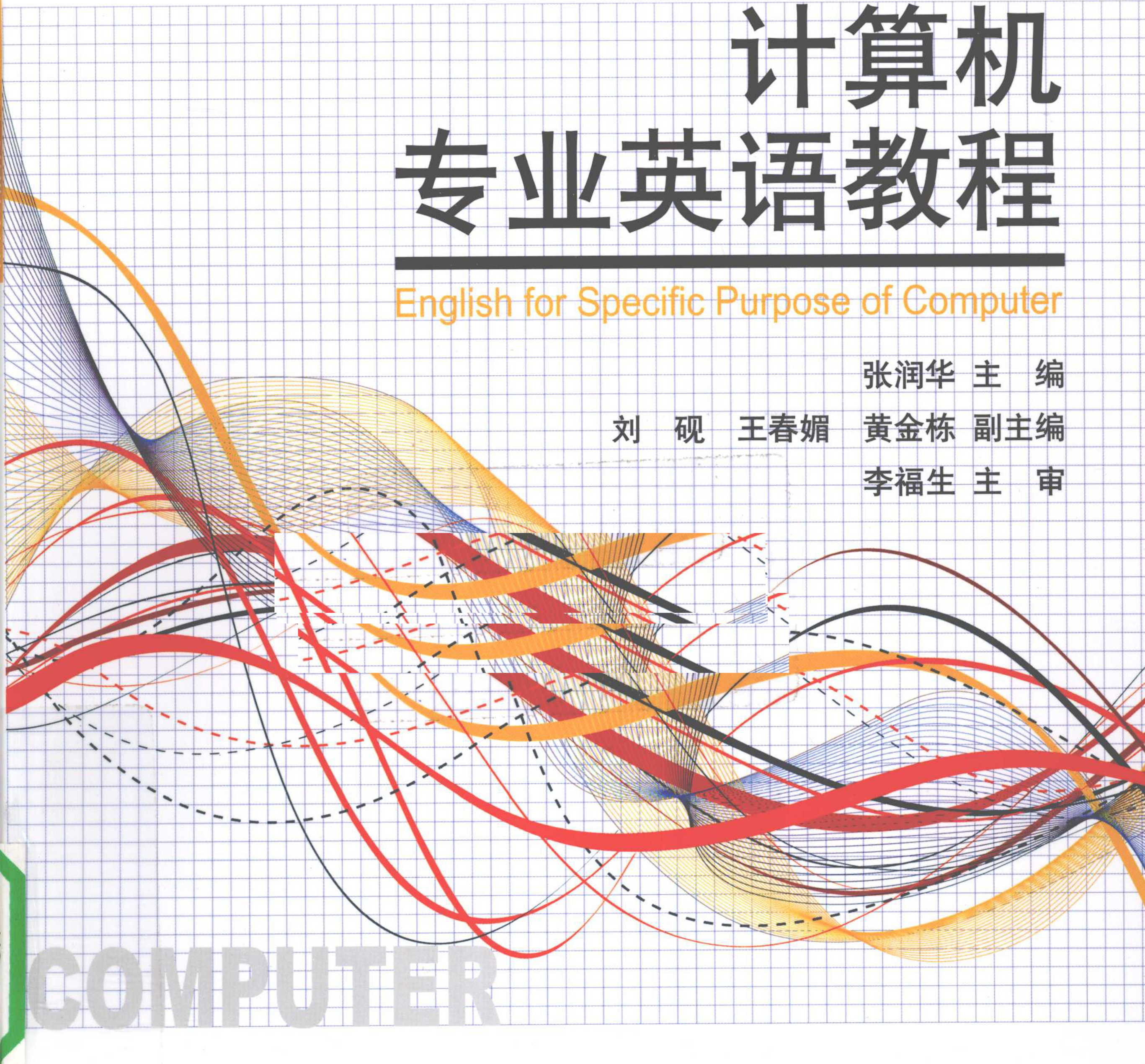
21世纪高等院校计算机系列课程教材

计算机专业英语教程

English for Specific Purpose of Computer

张润华 主　编
刘　砚　王春媚　黄金栋 副主编
李福生 主　审



北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

21 世纪高等院校计算机系列课程教材

计算机专业英语教程

主　编　张润华

副主编　刘　砚　王春媚　黄金栋

主　审　李福生

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内容简介

本书是高等院校教材，也可供计算机专业人员及其他有兴趣的读者学习参考。

本教材的编排思路是按照学生了解、使用计算机的过程来进行的。认知过程及各单元内容如下：认识计算机；了解计算机；操作计算机（操作系统的使用）；控制计算机（编程技术）；了解网络和 Internet（网络基础、上网冲浪）；保护计算机（预防病毒、查杀病毒）；计算机技术应用（数据库及其他数字应用技术）；了解计算机发展趋势。

本书素材均选取计算机科学各个领域的最新信息，内容新颖，覆盖面广、系统性强、可读性高。能帮助读者掌握计算机专业术语、培养和提高读者阅读及翻译计算机专业英语文献资料的能力。

图书在版编目（CIP）数据

计算机专业英语教程/张润华主编. —北京：北京理工大学出版社，2010. 4

ISBN 978 -7 -5640 -2969 -2

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

中国版本图书馆 CIP 数据核字（2010）第 007241 号

出版发行 / 北京理工大学出版社
社　　址 / 北京市海淀区中关村南大街 5 号
邮　　编 / 100081
电　　话 / (010)68914775(办公室)　68944990(批销中心)　68911084(读者服务部)
网　　址 / http: // www. bitpress. com. cn
经　　销 / 全国各地新华书店
印　　刷 / 天津紫阳印刷有限公司
开　　本 / 787 毫米 × 1092 毫米　1/16
印　　张 / 9. 25
字　　数 / 179 千字
版　　次 / 2010 年 4 月第 1 版　　2010 年 4 月第 1 次印刷
印　　数 / 1 ~ 2000 册　　责任校对 / 陈玉梅
定　　价 / 20. 00 元　　责任印制 / 边心超

前　言

在高等教育中，学生的基本素质和实际工作能力的培养受到了空前重视。对非英语专业的学生而言，英语水平和能力的培养不仅是文化素质的重要部分，在很大程度上也是能力的补充和延伸。在此背景下，教材是教学的基本要素之一，因此计算机专业英语教学的教材作用尤为突出。

作为计算机专业的学生，已经学习了公共英语，打下了扎实的基础，同时也已经学习了部分计算机专业基础课，了解了计算机相关技术，具有了一定的计算机专业背景知识。如何帮助学生记忆大量计算机专业词汇，看懂程序使用过程中的提示信息及硬件的说明文件，翻译计算机相关资料是本教材要解决的问题。

本教材由 10 个单元组成，内容涉及初识计算机、计算机硬件、Windows 操作系统功能介绍、编程技术、网络基础、互联网、数据库、视频编辑、计算机发展趋势等。内容既注重对专业知识面的覆盖，又着重反映最新的计算机发展及应用。目的是为学生提供阅读素材，通过大量阅读，提高学生阅读英语资料的能力。

每单元分为三部分：Lesson A 为精读课文，包括内容引导及提示、课文、词汇注释、练习、专业背景知识、参考译文等；Lesson B 为阅读材料，形式多样，帮助学生扩展视野；Lesson C 偏重语法及翻译技巧的介绍，同时给出了相关知识点的翻译练习，帮助学生切实提高计算机英语的翻译能力。其中精读课文和阅读材料多选自英美计算机相关网站，内容具有一定概括性，文字流畅，难度适中，能体现现代专业英语的篇章结构特点和词汇特点，使学生在学完本教程之后能顺利阅读计算机专业的英语资料。

本书由张润华担任主编，刘砚、王春媚、黄金栋为副主编，来自企业一线的李福生担任主审，主要的参编人员有王伟、刘晓英、李金勇等。

编者深知自己才疏学浅，知识有限，书中肯定有不少疏漏和错误，恳请读者及各位专家学者批评指正。对此，编者表示衷心的感谢。

编　者

目　　录

Unit 1 Begin with Computer Science

Lesson A Introduction of Computer Science

Ⅰ. *Getting started*

Think about these questions while reading: When did Computer Science began and how? Who helped Computer Science develop? What is Computer Science?

Ⅱ. *Text*

Theoretical[1] Computer Science has its roots in mathematics, where there was a lot of discussion of logic. It began with **Pascal and Babbage**[2] in the 1800's. Pascal and Babbage eventually tried to come up with computing machines that would help in calculating **arithmetic**[3]. Some of them actually worked, but they were **mechanical**[4] machines built on physics, without a real theoretical background.

Another person in the 1800's was a man named George Boole, who tried to **formulate**[5] a **mathematical**[6] form of logic. This was eventually called Boolean Logic in his honor, and we still use it today to form the heart of all computer hardware. All those **transistors**[7] and things you see on a **circuit board**[8] are really just physical **representations**[9] of what George Boole came up with.

Computer Science, however, hit the golden age with John von Neumann and Alan Turing in the 1900's. Von Neumann formulated the theoretical form of computers that is still used today as the heart of all computer design: the separation of the CPU, the RAM, the BUS, etc. This is all known **collectively**[10] as Von Neumann **architecture**[11].

Alan Turing, however, is famous for the theoretical part of Computer Science. He invented something called the Universal Turing Machine, which told us exactly what could and could not be computed using the standard computer architecture of today. This formed the basis of Theoretical Computer Science.

Ever since Turing formulated this extraordinary concept, Computer Science has been dedicated to answering one question: "Can

⌘ 批注：

1	理论上的，假设的
2	Pascal Blaise（1623—1662）法国数学家、物理学家、哲学家 Charles Babbage（1792—1871）英国数学家
3	算术，运算
4	机械的，力学的
5	构想出；写出公式
6	数学的
7	晶体管
8	电路板
9	陈述，表现
10	全体地，共同地
11	体系结构

we compute this?" This question is known as **computability**[12], and it is one of the **core disciplines**[13] in Computer Science. Another form of the question is "Can we compute this better?" This leads to more complications, because what does "better" mean?

So, Computer Science is partly about finding efficient **algorithms**[14] to do what you need.

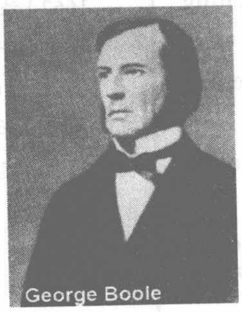
Still, there are other forms of Computer Science, answering such related questions as "Can we compute thought?" This leads to fields like **Artificial Intelligence**[15].

Computer Science is all about getting things done, to find **progressive solutions**[16] to our problems, to fill gaps in our knowledge. Sure, Computer Science may have some math, but it is different from math. All in all, Computer Science is about exploring the **limitations**[17] of humans, of expanding our horizons and having some fun at the same time.

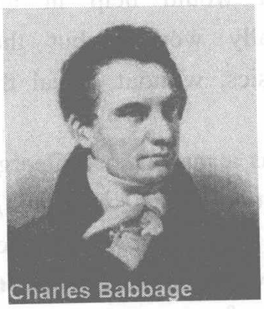
12	可计算性
13	核心学科
14	算法
15	人工智能
16	（问题的）逐步解决
17	局限，极限

Ⅲ. *Exercises*

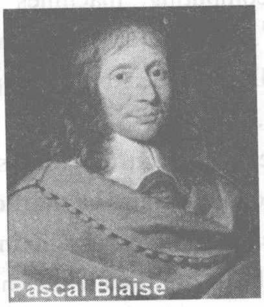
Match the person with the related events or items:



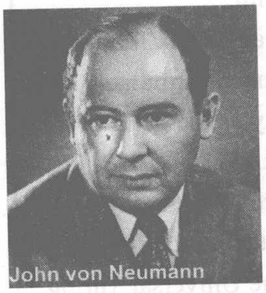
a



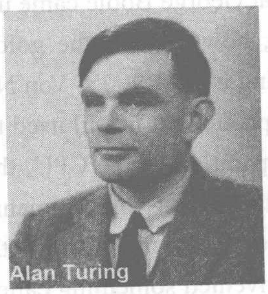
b



c



d



e

(1) He invented the Universal Turing Machine. (　　)

(2) He tried to formulate a mathematical form of logic. (　　)

(3) He formulated the theoretical form of computers that is still used today as the heart of all

computer design. (　　)

(4) A programming language is named in honor of him. (　　)

(5) He is credited with inventing the first mechanical computer. (　　)

Ⅳ. *Answers to Exercises*

e, a, d, c, b

Ⅴ. *Position Knowledge*

1. Turing and A.M.Turing Award

阿兰·麦席森·图灵（Alan Mathison Turing，1912.6.23—1954.6.7），英国数学家、逻辑学家，被视为计算机之父。1931 年进入剑桥大学国王学院，毕业后到美国普林斯顿大学攻读博士学位，“二战”爆发后回到剑桥，后曾协助军方破解德国的著名密码系统 Enigma，帮助盟军取得了“二战”的胜利。

1936 年，图灵向伦敦权威的数学杂志投了一篇论文，题为《论数字计算在决断难题中的应用》。在这篇开创性的论文中，图灵给“可计算性”下了一个严格的数学定义，并提出著名的“图灵机”（Turing Machine）的设想。“图灵机”不是一种具体的机器，而是一种思想模型，可制造一种十分简单但运算能力极强的计算装置，用来计算所有能想象得到的可计算函数。“图灵机”与“冯·诺伊曼机”齐名，被永远载入计算机的发展史中。1950 年 10 月，图灵又发表了另一篇题为《机器能思考吗》的论文，成为划时代之作。也正是这篇文章，为图灵赢得了“人工智能之父”的桂冠。

图灵奖，是美国计算机协会（ACM）于 1966 年设立的，又叫“A.M. 图灵奖”，是计算机界最负盛名的奖项，有“计算机界诺贝尔奖”之称。专门奖励那些对计算机事业做出重要贡献的个人。其名称取自计算机科学的先驱、英国科学家阿兰·图灵，设立这个奖的目的之一就是纪念这位科学家。获奖者的贡献必须是在计算机领域具有持久而重大的技术先进性的。大多数获奖者是计算机科学家。图灵奖对获奖者的要求极高，评奖程序也极严，一般每年只奖励一名计算机科学家，只有极少数年度有两名以上在同一方向上做出贡献的科学家同时获奖。目前图灵奖由 Intel 公司和 Google 公司赞助，奖金为 250 000 美元。截止到 2005 年，获此殊荣的华人仅有一位，他是 2000 年图灵奖得主姚期智。

2. Pascal and Niklaus

Pascal Blaise，是法国数学家、物理学家、思想家。16 岁时发现著名的帕斯卡六边形定理，17 岁时写成《圆锥曲线论》（1640），1642 年他设计并制作了一台能自动进位的加减法计算装置，被称为是世界上第一台数字计算器，为以后的计算机设计提供了基本原理。

为了纪念这位法国数学家，瑞士 Niklaus Wirth 教授于 20 世纪 60 年代末设计并创立了一种计算机通用的高级程序设计语言，便以 Pascal 来命名。Pascal 语言现已成为使用最广泛的基于 DOS 的语言之一，其主要特点有：严格的结构化形式；丰富完备的数据类型；运行效率高；查错能力强。

3. John von Neumann and ENIAC

1944 年夏的一天，正在火车站候车的冯·诺伊曼巧遇戈尔斯坦，并同他进行了短暂的交谈。当时，戈尔斯坦是美国弹道实验室的军方负责人，他正参与 ENIAC 计算机的研制工作。

在交谈中，戈尔斯坦告诉了冯·诺伊曼有关 ENIAC 的研制情况。具有远见卓识的冯·诺伊曼为这一研制计划所吸引，并意识到了这项工作的深远意义。

几天之后，冯·诺伊曼专程来到莫尔学院，参观了这台尚未完工的庞大机器，并以其敏锐的眼光，一下子抓住了计算机的灵魂——逻辑结构问题，令年轻的 ENIAC 的研制者们敬佩不已。因实际工作中对计算的需要以及把数学应用到其他科学问题的强烈愿望，使冯·诺伊曼迅速决定投身到计算机研制者的行列。对业已功成名就的他来说，这样做需要极大的勇气，因为这是一个成败未卜的新征途，一旦失败，会影响他已取得的名誉和地位。冯·诺伊曼却以其对新事物前途的洞察力，毅然决然地向此征途迈出了第一步，于 1944 年 8 月加入莫尔计算机研制小组，为计算机研制翻开了辉煌的一页。

冯·诺伊曼以其非凡的分析、综合能力及雄厚的数理基础，集众人之长，提出了一系列优秀的设计思想，在他和莫尔小组其他成员的共同努力下，只经历了短短的十个月，人类在数千年中积累起来的科学技术文明，终于结出了最激动人心的智慧之花——一个全新的存储程序通用电子计算机方案（EDVAC 方案）诞生了。

4. Charles Babbage

查尔斯·巴贝奇（Charles Babbage，1792—1871）：英国数学家和发明家、现代自动计算机的创始人，科学管理的先驱者。

巴贝奇的第一个贡献是制作了一台“差分机”。所谓“差分”的含义，是把函数表的复杂算式转化为差分运算，用简单的加法代替平方运算。1812 年，20 岁的巴贝奇从法国人杰卡德发明的提花编织机上获得了灵感，差分机设计闪烁出了程序控制的灵光——它能够按照设计者的旨意，自动处理不同函数的计算过程。巴贝奇耗费了整整十年光阴，于 1822 年完成了第一台差分机，它可以处理 3 个不同的 5 位数，计算精度达到 6 位小数，能当即演算出好几种函数表。由于当时工业技术水平极低，第一台差分机从设计绘图到机械零件加工，都是巴贝奇亲自动手完成。成功的喜悦激励着巴贝奇，他连夜奋笔上书皇家学会，要求政府资助他建造第二台运算精度为 20 位的大型差分机。然而，第二台差分机在机械制造过程中，因为主要零件的误差达不到每英寸千分之一的高精确度，以失败告终，但他把全部设计图纸和已完成的部分零件送进伦敦皇家学院博物馆供人观赏。

Ⅵ. *Translation*

理论计算机科学源于数学这一富含逻辑思辨的学科。它诞生于 19 世纪，与帕斯卡（Pascal）和巴贝奇（Babbage）的贡献密不可分。当时，这两位数学家试图设计出一些计算器以提高算术运算的速度。这些计算器中的一部分是完全可以用于计算工作的，但这些计算器仅仅是基于物理学而设计出的机械装置，缺乏真正的理论基础。

在同一时代，还有一个叫乔治·布尔（George Boole）的人，致力于定义一套逻辑的代数系统。后人为了纪念他为此做出的贡献，将这套系统命名为“布尔逻辑”。时至今日，这套理论仍然在计算机硬件构建中体现了重要价值。各类晶体管以及电路板上的各种元件实际上都是布尔逻辑理论的物理实现。

然而，直到 20 世纪，约翰·冯·诺依曼（John von Neumann）与阿兰·图灵（Alan Turing）的出现，才使计算机科学真正进入了黄金时代。冯·诺依曼建立的计算机理论模型一直沿用至今，被视为计算机设计的核心思想，包括中央处理器、存储器、总线等，这些部件被统称

为“冯诺依曼结构”。

阿兰·图灵则由于在计算机科学理论方面的突出贡献为自己赢得了极大声誉。他发明了一种名为“通用图灵机”的模型，这个理论模型可以确切的判定当前标准结构计算机的计算能力范畴，为理论计算机科学奠定了坚实的基础。

自图灵提出这套卓越的理论模型后，计算机科学便开始致力于解决“该计算是否可行”，即“可计算性”的问题，并作为核心学科纳入计算机科学范畴。“可计算性”问题又衍生出另一个关于“是否可以将计算优化”的讨论，而这将是十分复杂的，毕竟“优化”意味着什么呢？

因此，在某种程度上，计算机科学就是一门为各类难题寻求高效题解的学科。

计算机科学还包含很多其他的范畴，以解决类似“计算机是否可以思考”的问题，这也使其衍生出“人工智能”等学科领域。

计算机科学是解决问题的科学，是积极寻求题解、填补人类知识空白的科学。虽然它起源于数学，但又不同于数学。总之，计算机科学致力于探索人类的极限，开阔人类的视野，同时又为人们带来欢愉。

Lesson B How to Study Computer Science

Ⅰ. *Task Introduction*

Before Reading: To study Computer Science, what do you need to learn, and what don’t you?

If you’re an **aspiring**[1] computer science student or someone who wants to **switch**[2] fields into **CS**[3], you’re in luck; there’s a lot of information **available**[4] on the Internet. CS is a large and rapidly-expanding field; once you've become confident in your abilities to program moderate-sized projects, a lot of topics open up to you. But what do you really need to learn about, and what don’t you?

A lot of it depends on what you want to get out of your study. If you want to become a researcher, you’ll most likely need to know more of the theory of computer science than if you want to become a **programmer**[5]. There are, however, some basic skills that will help nearly everyone in the field.

Learn Multiple[6] Programming Languages

No matter what you want to do in computer science, you’ll likely do some of it by writing computer programs. Not all languages are created equal, but most of them have some strengths. You’ll want to learn a systems language like C or C++. This will give you several advantages: first, you’ll understand **memory allocation**[7]; second,

⌘ 批注：

1	有志气的，有抱负的
2	改变，转变
3	Computer Science
4	可利用的
5	程序员
6	多种，多样的
7	内存分配

you'll understand more about how the system is designed; and finally, you'll be able to communicate with other programmers more easily. You can see this article for more details on the advantage of learning C.

But you want to learn a more **flexible**[8] language for daily **chores**[9]-for instance, a **scripting language**[10] like Perl or Ruby will help you quickly create interesting programs and test ideas.

8 灵活的，可变通的
9 日常事务，例行工作
10 脚本语言

Finally, once you've mastered a language or two, expand your horizons with a **functional language**[11] like Scheme, ML, or Haskell. This will improve your understanding of programming languages and broaden your horizons about the possibilities.

11 函数式语言

A key thing to remember when learning new languages is that all languages offer the same power-you can do anything in one language that you can in another-but some languages make it easier to do certain things. For instance, if you want to read data from a text file, Perl is a great language. If you want to write an AI engine, then you might be **better off with**[12] Scheme.

12 较……更好

Learn to Design

Whether you want to work as a software engineer or as an **academic**[13], you're going to have to design programs in some form or another. Learning good design **principles**[14] early will make your life easier. The key thing to remember about design is that the goal is to catch the problems before you've **committed to**[15] a solution that won't let you fix them. You don't have to do all of your design up front, but if you don't, then you'll want to leave more **flexibility**[16] later.

13 专业学者
14 原则，原理
15 致力于
16 灵活性，变通性

Of course, some amount of design is absolutely **crucial**[17] or you'll simply have no idea what should be flexible and what can be hard **coded**[18]. Overly **modular**[19] designs can be as deadly and difficult to **maintain**[20] as extremely inflexible designs. One way of looking at the issue is that **modularity**[21] is powerful because it makes it easier to replace a bad idea with a good one. But if you know what the good idea is going to be anyway, then modularity doesn't help you, and because it takes more effort, it can hurt you.

17 决定性的，关键性的
18 编码，编程序
19 模块化
20 保持，继续
21 模块化，模块性

A good way to learn design is to practice on well-known systems projects, like writing an **interpreter**[22] or web server. These kind of projects have the advantage of having well-known **implementions**[23] that you can look at once you realize that there are problems with your original design. Whatever you do design, you definitely want to

22 解释程序，解释器
23 实现

implement at least parts of your designs or you'll never really come to understand the drawbacks in your ideas. It's **running into**[24] these drawbacks that will teach you the most.

24	遇到，碰见

Learn Basic Algorithms and Data Structures[25]

25	数据结构

There are a variety of important algorithms and data structures that are the **lingua franca**[26] of computer science. Everyone needs to know what a <>or **binary tree**[27] is because they get used all the time. Perhaps just as important are fundamental algorithms like binary search, graph searching algorithms, **sorting algorithms**[28], and **tree-based searches**[29] such as **minimax**[30]. These algorithms and their **variants**[31] show up a lot, and people will generally expect that any computer scientist will understand how they work.

26	通用语言
27	二叉树
28	排序算法
29	基于树的搜索
30	极大极小（树）
31	变体

Learn Basic Theory

There are a couple of things you should definitely be aware of: you should understand how to represent numbers in different **bases**[32] and how to **manipulate**[33] **boolean**[34] **expressions**[35] using boolean logic. Both of these tools will come in handy in a variety of circumstances, especially when reading other people's code or trying to **clarify**[36] your own. (Boolean logic is especially useful for formulating clear **conditional statements.**[37])

32	基数
33	操作，生成
34	布尔（型）的
35	表达式
36	阐明
37	条件语句

Even more important, you should have a good sense of the limits of current computers. In particular, it really helps to understand the ideas in algorithmic efficiency and Big-O **notation**[38]. Knowing these topics makes it clearer why certain programs and algorithms will take a very long time to run, and how to **spot**[39] them. It also makes it **significantly**[40] easier to **optimize**[41] your program when you know which algorithms to choose.

38	符号
39	看出，发现
40	显著地
41	优化

Finally, it's useful to have a sense of the absolute limits of what computers can do; it turns out that there are just some things that are not possible for a computer to do. Sometimes it can be **tempting**[42] to try to write a program that does just that!

42	吸引人的，诱人的

Ⅱ. *Have a try*

1. **There are many different kinds of programming languages you might want to learn. Read the** "Learn Multiple Programming Languages" **part and complete the chart.**

	Systems language	*Scripting language*	*Functional language*
Example			
Advantages			

2. Check (√) whether the sentences are true or false. Correct the false sentences.

(1) Not every language could do anything you want. (　　)

(2) If you want to be a software engineer, you're not going to have to design programs. (　　)

(3) When designing, if you could catch the problems first, you won't need to fix them so often after you've committed to a solution.(　　)

(4) Learn basic theory of Computer Science could help you when reading other people's code or trying to clarify your own.(　　)

(5) Everyone needs to know what a <>or binary tree is because people will generally expect that any computer scientist will understand how they work. (　　)

Ⅲ. *Answer to Have a try*

1.

	Systems language	*Scripting language*	*Functional language*
Example	C, C++	Perl, Ruby	Scheme, ML, Haskell
Advantages	1. Make you understand memory allocation; 2. Make you understand more about how the system is designed; 3. Make you be able to communicate with other programmers more easily	Help you quickly create interesting programs and test ideas.	Improve your understanding of programming languages and broaden your horizons about the possibilities.

2. 3 √, 4 √

Lesson C Translation Tips

Ⅰ. *Technical Terms*（专业术语）

在计算机科学文献中经常会出现大量的计算机专业术语，这些术语是承载科技要领的语言符号，在计算机技术领域占有重要的地位，更是计算机英语翻译的难点。

许多生活中常见的英语词汇在计算机英语中被赋予专业的意思，准确把握这些专业术语的涵义是理解整个篇章的关键和基础。如：Memory，记忆，在计算机英语中则指存储器，内存。Boot，穿靴子，在计算机中指登录。Java，爪哇岛（属印度尼西亚，在计算机语言中指程序设计语言及平台）。bug 意为小虫子，比喻隐藏在程序中的小错误等等。

本章出现了很多具有上述特征的计算机专业术语，以下将列出其中部分用语。请在翻译成中文后强化记忆，并试着从文章中找到更多的专业词汇，在后面章节的学习中也要保持这一学习习惯：在文章中搜索专业术语，然后记忆，记忆，再记忆！

序号	英文形式	中文形式
1	Artificial Intelligence	
2	modularity	
3	interpreter	
4	sorting	
5	expression	
6	statement	
7	script	
8	base	
9	functional language	
10	binary tree	

Ⅱ. *Answer*

序号	英文形式	中文形式
1	Artificial Intelligence	人工智能
2	modularity	模块化
3	interpreter	解释程序，解释器
4	sorting	排序
5	expression	表达式
6	statement	语句
7	script	脚本
8	base	基数
9	functional language	函数式语言
10	binary tree	二叉树

Unit 2 Computer Hardware

Lesson A Hardware[1]

Ⅰ. *Getting started*

To enjoy using a computer, it is necessary to know about the organization of computers and how components of computer works. It is worth for you spending sometime reading this brief introduction to computer hardware.

Ⅱ. Text

The hardware are the parts of computer itself including the **Central Processing Unit**[2] (CPU) and related **microchips**[3] and **micro-circuitry**[4], keyboards, monitors, case and drives (hard, CD, DVD, floppy, tape, etc...). Other extra parts called peripheral components or devices include mouse, printers, modems, scanners, digital cameras and cards (sound, colour, video) etc... Together they are often **referred to as**[5] a **personal computer**[6].

Central Processing Unit — Though the term relates to a specific chip or the processor a CPU's **performance**[7] is determined by the rest of the computer's circuitry and chips.

Keyboard — The keyboard is used to **type**[8] information into the computer. There are many different **keyboard layouts**[9] and sizes with the most common for Latin based languages being the **QWERTY**[10] layout (named for the first 6 keys). The standard keyboard has 101 keys. **Ergonomically**[11] designed keyboards (Fig.2-1) are designed to

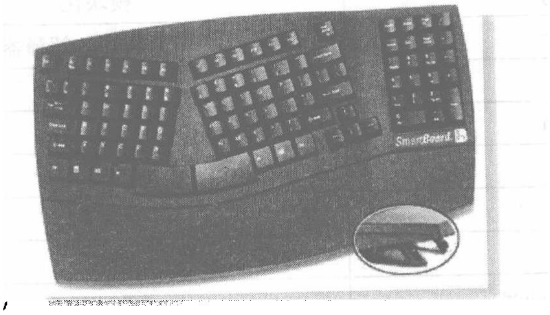


Fig.2-1 *A keyboard with an ergonomic design*

⌘ 批注：

1	硬件
2	中央处理器
3	微芯片
4	微电路
5	被称为
6	个人计算机，简称 PC 机
7	性能
8	输入
9	键盘布局
10	标准的传统键盘
11	人体功率学的

make typing easier. Hand held devices have various and different keyboard configurations and **touch screens**[12].

12 触摸屏

Some of the keys have a special use. There are referred to as command keys. The 3 most common are the **Control or CTRL, Alternate or Alt and the Shift keys**[13]. Each key on a standard keyboard has one or two **characters**[14]. Press the key to get the lower character and hold Shift to get the upper.

13 控制键、切换键、换挡键
14 字符

Disk Drives[15] — All disks need a drive to get information off — or read — and put information on the disk — or write. Each drive is designed for a specific type of disk whether it is a CD, DVD, hard disk or floppy. Often the term ‘disk’ and ‘drive’ are used to describe the same thing but it helps to understand that the disk is the storage device which contains computer files — or software — and the drive is the **mechanism**[16] that runs the disk.

15 磁盘驱动器
16 机械装置

Digital **flash drives**[17] work slightly differently as they use **memory cards**[18] to store information so there are no **moving parts**[19]. Digital cameras also use Flash memory cards to store information, in this case photographs. Hand held devices use digital drives and many also use memory cards.

17 存储卡、U 盘，或闪存
18 存储卡；记忆卡；存储器插件板
19 运动机件

Mouse — Most modern computers today are run using a **mouse controlled pointer**[20]（Fig.2-3）. Generally if the mouse has two buttons the left one is used to select objects and text and the right one is used to access menus.

20 鼠标控制的指针

Fig.2-2

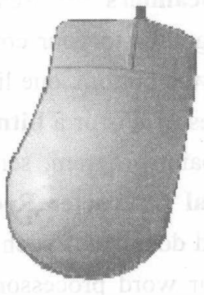


Fig.2-3

One type of mouse has a round ball under the bottom of the mouse that rolls and turns two wheels which control the direction of the pointer on the screen. Another type of mouse uses an optical system to **track**[21] the movement of the mouse. **Laptop computers**[22] use **touch pads**[23], buttons and other devices to control the pointer.

21 寻路而行、跟踪
22 膝上计算机
23 触摸盘、触摸板