任务型语码转换式双语教学系列教材

总主编 刘玉彬 副总主编 杜元虎 总主审 段晓东

计算机应用技术

COMPUTER APPLICATION TECHNOLOGY

主 编 李锡祚

大连理工大学出版社

任务型语码转换式双语教学

总主编 刘玉彬 副总主编 杜元虎 总主审 段晓东

计算机应用技术

COMPUTER APPLICATION TECHNOLOGY

主　编 李锡祚
副主编 王玲芬

大连理工大学出版社

图书在版编目(CIP)数据

计算机应用技术：汉英对照 / 李锡祚主编. — 大连：大连理工大学出版社，2014.8
任务型语码转换式双语教学系列教材
ISBN 978-7-5611-7554-5

Ⅰ. ①计… Ⅱ. ①李… Ⅲ. ①电子计算机－高等职业教育－教材－汉、英 Ⅳ. ①TP3

中国版本图书馆 CIP 数据核字(2013)第 004443 号

大连理工大学出版社出版
地址：大连市软件园路 80 号　邮政编码：116023
发行：0411-84708842　邮购：0411-84703636　传真：0411-84701466
E-mail：dutp@dutp.cn　URL：http://www.dutp.cn
大连金华光彩色印刷有限公司印刷　　大连理工大学出版社发行

幅面尺寸：183mm×233mm　印张：18.75　字数：607 千字
2014 年 8 月第 1 版　　2014 年 8 月第 1 次印刷

责任编辑：邵　婉　　责任校对：齐　悦
封面设计：波　朗

ISBN 978-7-5611-7554-5　　定　价：35.00 元

总序
PREFACE

2014年的初夏，我们为广大师生奉上这套“任务型语码转换式双语教学系列教材”。

“任务型语码转换式双语教学”是双语教学内涵建设的成果，主要由两大模块构成：课上，以不影响学科授课进度为前提，根据学生实际、专业特点、学年变化及社会需求等，适时适量地渗透英语专业语汇、语句、语段或语篇，“润物细无声”般地扩大学生专业语汇量，提高学生专业英语能力；课外，可向学生提供多种选择的“用中学”平台，如英语科技文献翻译、英语实验报告、英语学术论文、英语小论文、英语课程设计报告、模拟国际研讨会、英语辩论、工作室英语讨论会等，使学生的专业英语实践及应用达到一定频度和数量，激活英语与学科知识的相互渗透，培养学生用英语学习、科研、工作的能力及适应教育国际化和经济一体化的能力。

为保证“任务型语码转换式双语教学”有计划、系统、高效、科学地持续运行，减少教学的随意性和盲目性，方便师生的教与学，我们编写了这套“任务型语码转换式双语教学系列教材”。

本套教材的全部内容均采用汉英双语编写。

教材按专业组册，涵盖所有主干专业课和专业基础课，力求较为全面地反映各学科领域的知识体系。

分册教材编写以中文版课程教材为单位，即一门课为分册教材的一章，每章内容以中文版教材章节为序，每门课以一本中文教材为蓝本，兼顾其他同类教材内容，蓝本教材绝大部分是面向21世纪的国家规划教材。

教材的词汇短语部分，注意体现学科发展的新词、新语，同时考虑课程需求及专业特点，在不同程度灵活渗透了各章节的重要概念、定义，概述了体现章节内容主旨的语句及语段。分册教材还编写了体现各自专业特点的渗透内容，如例题及解题方法，课程的发生、发展及前沿简介，图示，实验原理，合同文本，案例分析，法条，计算机操作错误提示等。

部分教材补充了中文教材未能体现的先进理论、先进工艺、先进材料或先进方法的核心内容，弥补了某些中文教材内容相对滞后的不足；部分教材概述了各自专业常用研究方法、最新研究成果及学术发展的趋势动态；部分

教材还选择性地把编者的部分科研成果转化为教材内容，以期启发学生的创新思维，开阔学生的视野，丰富学生的知识结构，从教材角度支持学生参与科研活动。

本套教材大多数分册都编写了对“用中学”任务实施具有指导性的内容，应用性内容的设计及编写比例因专业而异。与专业紧密结合的应用性内容包括英语写作介绍，如英语实验报告写作，英语论文写作，英语论文摘要写作，英语产品、作品或项目的概要介绍写作等。应用性内容的编写旨在降低学生参与各种实践应用活动的难度，提高学生参与“用中学”活动的可实现性，帮助学生提高完成“用中学”任务的质量水平。

考虑学生英语写作和汉译英的方便，多数分册教材都编写了词汇与短语索引。

“任务型语码转换式双语教学系列教材”尚属尝试性首创，是多人辛勤耐心劳作的结果。尽管在编写过程中，我们一边使用一边修改，力求教材的实用性、知识性、先进性融为一体，希望教材能对学生专业语汇积累及专业资料阅读、英语写作、英汉互译能力的提高发挥作用；尽管编者在教材编写的同时也都在实践“任务型语码转换式双语教学”，但由于我们缺乏经验，学识水平和占有资料有限，加上为使学生尽早使用教材，编写时间仓促，在教材内容编写、译文处理、分类体系等方面存在缺点、疏忽和失误，恳请各方专家和广大师生对本套教材提出批评和建议，以期再版时更加完善。

在教材的编写过程中，大量中外出版物中的内容给了我们重要启示和权威性的参考帮助，在此，我们谨向有关资料的编著者致以诚挚的谢意！

编　者

2014 年 5 月

前言
FOREWORD

随着高等教育国际化的不断发展，“双语教学”作为一种新的教育教学方式，被认为是培养国际化人才的一个重要途径。对于计算机类专业，由于英语作为主要的专业技术描述语言，所以实施双语教学对学生日后进入全球化的信息技术领域进行国际交流，提高学生的国际竞争力，培养国际化计算机人才具有重要意义。针对我国计算机类专业双语教学的实际情况，为了使更多的学生都能在贯穿大学四年的各门课程教学和符合自己水平的不同层次的双语教学中受益，改变目前双语教育只能面向极少数学生的精英培养模式为普遍培养模式，我们构建了一个以人为本，符合认知规律，省时高效的语码转换式双语教学模式，即在课堂教学中，依托课程内容，采取语码转换式教学，不断渗透双语，并编写了《计算机科学与工程》(大连理工大学出版社，2008 年)。经过几年的努力，我们取得了一系列双语教学成果，并提出了可普遍实施、让学生大面积受益的任务型语码转换式双语教学模式。它主要包括两大模块：课上以不影响学科授课进度为前提，根据学生实际、专业特点、学年变化及社会需求等，适时适量地渗透专业英语词汇、语句、语段或语篇，“润物细无声”般地扩大学生专业词汇，提高学生专业英语能力；课外通过可向学生提供多种选择的“用中学”平台，如英语科技文献翻译、英语学术论文、英语小论文、英语课程设计报告、英语实验报告、英语辩论和英语竞赛等，使学生的专业英语实践达到一定频度和数量，激活英语与专业知识的相互渗透，培养学生用英语学习、科研、工作的能力及适应教育国际化、经济全球化的能力。这本《计算机应用技术》正是我校双语教学改革的重要成果之一，也是提高计算机类专业学生英语水平的重要手段之一。

本书以中英文对译形式汇集了计算机类专业主要课程和一些专业方向的简介、常用词汇、短语、基本概念、定义和语段等内容，这些课程和专业方向包括程序设计基础、面向对象程序设计、计算机组成原理、离散数学、数据结构与算法、操作系统、数据库概论、编译技术、软件工程、计算机网络、JAVA 技术、软件分析与技术、微型计算机接口技术、XML 技术、嵌入式系统、人工智能、计算机图形学、人机交互技术、企业级软件开发、软件质量保证与测试、网络安全、游戏编程专业方向、数字逻辑与数字系统、电子商务等。书中对基本概念和定义给出了详细的中英文说明，并在每章节以★标出重点词汇，以示必须掌握。

为了方便读者，书中词汇、短语、概念和定义按照相应的中文教材章节中出现的次序进行编排，并在书后提供了中文索引，以便快速查阅。

本书共汇编了近6000个词条，包括词汇、短语、概念和定义。这些词条直接选自英文文献、最新原版教材及专业手册等，有些词汇或术语的翻译，不是机械地按英文直译，而是按专业术语或习惯给出译文，保证了本书具有较好的适用性。在附录中，还增加了VC++开发环境常见错误信息及解决方案、Oracle数据库常见错误信息、Java错误信息及解决方案、英文摘要范例和英文实验报告范例等，这使本书不仅适合于理论教学还适合于实践教学。因此，本书对于读者循序渐进地掌握计算机类专业英语词汇以及专业英语的表达方式，阅读计算机类专业文献，开展国际学术交流具有一定的裨益。

本书由李锡祚担任主编，并负责全书统稿。参加编写的人员有王晓强（第一章）、宋海玉（第二章）、李玲华（第三、十三、二十三章和附录五）、姜楠（第四、二十一章）、李锡祚（第五、九章和附录四）、王玲芬（第六章和索引）、何丽君（第七章和附录二）、王朋杰（第八、十七和二十章）、云健（第十章）、赵晶莹（第十一、十二、十九章和附录三）、郭海（第十四章）、魏巍（第十五章）、刘爽（第十六章）、包书哲（第十八章）、李威（第二十二章和附录一）、赵丹丹（第二十四章）。

在编写本教材的过程中，参考了大量国内外有价值的参考文献，这些资料在参考文献中已经列出，在此特向这些书的作者致以崇高的敬意。

由于编者水平有限，本书所收录的计算机应用技术方面的词汇、短语、概念和定义还很不全面，而且错误和不足之处在所难免，深望读者批评指正。

编 者

2014年7月

使用说明

1. 正文中★的含义：表示该词条在本篇章中为重要词条，要求学生必须掌握。
2. 查阅方法：本书可从两方面进行查阅。一种是按照课程的章节顺序进行查阅。另一种是按索引法，即按照词条的拼音顺序查找词条的出处，再查阅正文。

目录 CONTENTS

目录 CONTENTS

第一章　程序设计基础

Chapter 1　Programming Fundamentals

The course takes the C language as the carrier, and the teaching content includes the basic concepts of programming, the basic knowledge of algorithm design, and the basic syntax of C language. Students can initially establish the way of thinking to solve simple problems by computer, master the required basic technology of programming, and be able to design, coding, and debug the source programs after learning this course. And also, the course is the basis of the follow-up courses.

本课程以C语言为载体，教学内容包括：程序设计的基本概念、算法设计的基本知识、C语言的基本语法等。学生学习本课程后，能够初步建立应用计算机解决简单问题的思维方式，掌握程序设计所需的基本技术，能够设计、编写、调试解决简单问题的计算机程序，为后续课程奠定程序设计的基础。

第一节　程序设计概论

Section 1　Introduction to Programming

程序设计　programming
计算机语言　computer language
机器语言　machine language
汇编语言　assembly language

1 算法　algorithm: An algorithm is a method or procedure for carrying out a task.
算法是完成一项任务所采用的方法或步骤。

2 数据结构　data structure: In computer science, data structure is a way of storing and organizing data in a computer, and it makes programming efficiently.
在计算机科学中，数据结构是一种在计算机上存储和组织数据的方式，它可以使程序设计更加有效。

第二节　算法设计基础

Section 2　Basics of Algorithm Design

算法描述　algorithm description
伪代码　pseudo-code
自顶向下　top-down

1 流程图　flow chart: A flow chart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. A programmer can solve a given problem step-by-step by using the flow chart.
流程图是一种代表算法或过程的图形，它通过各种不同类型的图形展示解决问题的各个步骤。各个图形之间的顺序通过带箭头的直线连接。程序员通过流程图可以一步一步地解决一个指定的问题。

2 结构化程序设计　structured programming: Structured programming is a method of program design that emphasizes a modular, top-down approach.
结构化程序设计是一种程序设计方法，它强调模块化、自顶向下的设计思想。

3 可移植性　portability: Portability means that a program written for one computer system can be compiled and run on another system with little or no modification.
可移植性意味着为一个计算机系统写的程序，只作很小的修改或不作修改就可以在另一个系统上编译和运行。

第三节　C语言基础

Section 3　Basics of C Language

字符集　character set
编辑　edit
编译　compile
链接　link
运行　run
逻辑错误　logic error
运行时错误　run-time error
语法错误　syntax error

编译器 compiler
编辑器 editor
连接器 linker
源程序 source code
目标代码 object code
注释 comment / remark
换行符 newline character (\n)
调试 debug
变量声明 variable declaration
变量定义 variable definition
初始化 initialization
数据类型 data type
操作数 operand
字面常量,直接常量 literal constant
符号常量 symbolic constant
小写 lowercase
大写 uppercase
区分大小写 case-sensitive
下划线 underscore
分号 semicolon
整型 integer
无符号 unsigned
有符号 signed
浮点型 floating point
补码 complement
字符 character
字符串 string
赋值运算符 assignment operator
算术运算符 mathematical operator
算术表达式 mathematical expressions
类型转换 type conversion
自动类型转换 auto type conversion
强制类型转换 cast
自增运算符 increment operator
自减运算符 decrement operator
赋值表达式 assignment expression
逗号运算符 comma operator
逗号表达式 comma expression
运算符优先级 operator precedence
单目运算符 unary operator
双目运算符 binary operator
三目运算符 ternary operator
关系表达式 relational expression
if 语句 if statement
逻辑表达式 logical expression
条件运算符 conditional operator
转义符 escape character(\)
格式控制字符串 format string

1 关键词 keyword: In the programming language, a keyword is a word or identifier that has a particular meaning.
在程序设计语言中,关键词是一个具有特殊意义的词或标识符。

2 变量 variable: A variable is a named data storage location in our computer's memory. By using a variable's name in our program, we are, in effect, referring to the data stored there. And it has a value that can change during program execution.
变量是一个在我们计算机内存中被命名了的数据存储位置,通过在程序中使用变量名,就可以指向这个存储位置,并且它有一个可以在程序执行期间改变的值。

3 常量 constant: Like a variable, a constant is a data storage location used by our program. Unlike a variable, the value stored in a constant can't be changed during program execution.
与变量相同的是,常量也是一个在我们程序中被使用的数据存储位置。与变量不同的是,存储在常量中的值在程序的执行期间是不可改变的。

4 表达式 expression: In C, an expression is anything that evaluates to a numeric value. C expression consists of simple expressions and complex expressions. Simple expression consists of a single item: a simple variable, literal constant or symbolic constant; Complex expressions consist of simpler expressions connected by operators.
在 C 语言中,表达式可以求出数值。它有简单表达式和复合表达式两种。其中,简单表达式由一些简单的项组成,包括:简单变量、字符常量或符号常量等;而复合表达式由简单表达式和运算符连接而成。

5 运算符 operator: An operator is a symbol that instructs C to perform some operations, or actions, on one or more operands.
运算符是一个符号,它可以使 C 语言在一个或多个操作数上执行一些操作或行为。

6 关系运算符　relational operator: Relational operators always perform comparisons between their operands (for example, greater than).
关系运算符通常在它们的操作数间进行比较(比如:大于)。

7 逻辑运算符　logical operator: Logical operators operate on true/false expressions. C uses 0 and 1 to represent false and true respectively, and that any nonzero value is interpreted as being true.
逻辑运算符用于操作真或假表达式,在C语言中,使用1和0来分别代表真和假。另外,任何非零值也都被解释成真。

第四节　基本控制结构

Section 4　Basic Control Structures

顺序结构　sequence structure
选择结构　selection structure
循环结构　loop structure
循环体　loop body
嵌套　nesting
复合语句　compound statement
空语句　null statement
迭代　iteration

1 语句　statement: A statement is a complete direction instructing the computer to carry out some task. In C, statements are usually written one per line, although some statements span multiple lines, and always end with a semicolon.
语句是使计算机执行一些任务的一条完整指令。在C语言中,尽管有些语句需要跨行,但通常情况下,语句写在一行之内,并且以一个分号结束。

2 循环　loop: A loop is a sequence of statements which is specified once but which may be carried out several times in succession. The code "inside" the loop (the body of the loop) is obeyed a specified number of times, or once for each of a collection of items, or until some conditions are met, or indefinitely.
一个循环是一个语句序列,指定一次,但可以连续执行几次。循环内部的代码(循环体)服从一个指定的次数,或每一次的项目集合,或直到某些条件得到满足,或无限期。

3 嵌套循环　nested loop: The term nested loop refers to a loop that is contained within another loop.
术语嵌套循环是指一个循环包含在另一个循环之中。

第五节　数组、结构体、联合和枚举

Section 5　Arrays, Structures, Unions and Enum

抽象数据类型　Abstract Data Type(ADT)
数组元素　element of an array
数组初始化　array initialization
二维数组　two-dimensional array
多维数组　multidimensional array
成员运算符　structure member operator(.)
成员选择运算符　member selection operator(->)
枚举　enumeration
位运算符　bitwise operator
与运算符　AND operator(&)
或运算符　inclusive OR operator(|)
异或运算符　exclusive OR operator(^)
左移位运算符　left shift operator
右移位运算符　right shift operator
位段　bit field

1 位　bit: A bit is the smallest unit of data storage, and it can have only one of two values: 0 or 1.
位是数据存储的最小单位,它仅仅能在0和1之间取一个值。

2 数组　array: An array is a collection of data storage locations, each having the same data type and the same name. Each storage location in an array is called an array element and is distinguished from each other by a subscript.
数组是一个数据存储单位的集合,它们有相同的名字和数据类型。每一个在数组中的数据存储单位被称为数组元素,它们相互之间通过下标进行区别。

3 下标　subscript (or index): The subscript (or index) is a number following the array name,

enclosed in brackets.

下标(或称索引)是一个数字,紧跟在数组名后,并使用方括号括起来。

4 一维数组 single-dimensional array: A single-dimensional array has only a single subscript.

仅仅有一个下标的数组称为一维数组。

5 结构体 structure: A structure is a collection of one or more variables grouped under a single name for easy manipulation. The variables in a structure, unlike those in an array, can be of different variable types. A structure can contain any of C's data types, including arrays and other structures. Each variable within a structure is called a member of the structure.

结构体可以被看做是一个或多个变量的集合,它们拥有同一个名字,从而更容易操作。结构体中的变量与数组中的变量不同,它们可以有不同的数据类型。结构体可以包含C语言中所有的数据类型,包括数组和其他结构体。在结构体中的每一个变量都被称为结构体的一个成员。

6 共用体 union: Unions are similar to structures. A union is declared and used in the same ways that a structure is. A union differs from a structure in that only one of its members can be used at a time. The reason for this is simple. All the members of a union occupy the same area of memory.

共用体和结构体很相似,它们以同样的方式被声明和使用。共用体和结构体的区别之处仅仅在于在某一时刻,共用体仅仅有一个成员可以被使用。原因很简单,结构体中的所有成员占有内存中的同一块区域。

7 枚举 enum: In programming, the enumerated type (enum) is a data type consisting of a set of named values called elements or members. The enumerator names are usually identifiers that behave as constants in the language.

在程序设计中,枚举类型是一种由被命名了的值组成的集合,这些值通常被称为元素或成员。在程序设计语言中,枚举值通常被当成常量来使用。

第六节 函数

Section 6 Functions

模块化 modularization
函数定义 function definition
函数原型 function prototype
函数声明 function declaration
无参函数 no argument function
函数头、函数名 function header
函数体 function body
形参表列 parameter list
空函数 null function
传递值 pass argument
返回值 return value
被调用函数 called function
调用者 caller
全局变量 global variable
存储类型 storage type
作用域 scope
生存期 lifetime
寄存器变量 register variable
自动变量 automatic variable
静态变量 static variable
外部变量 external variable
外部函数 external function
库函数 library function

1 函数 function: Functions are independent sections of code that perform specific tasks. When our program needs a task performed, it calls the function that performs that task.

函数是一些执行特定任务的独立的代码块。当我们的程序需要去完成一个指定的任务时,程序就调用某个特定的函数去完成这项任务。

2 实参 argument: An argument is an actual value passed to the function by the calling program. Each time a function is called, it can be different arguments to be passed.

实参是当函数被程序调用的时候被传递给函数的实际的值。每一次函数被调用的时候,传递给它的实参值都可能是不同的。

3 形参 parameter: A parameter is an entry in a function header; it serves as a "placeholder" for an

argument. A function's parameters are fixed; they do not change during program execution.
形参是在函数头中的一个入口，它为实参保留一个“占位符”。在一个函数中，形参是固定的，在程序的执行期间它们不发生变化。

3 局部变量　local variable: Variables declared in a function are called local variables. The term local means that the variables are private to that particular function and are distinct from other variables of the same name declared elsewhere in the program.
声明在一个函数内的变量被称为局部变量。属于局部意味着，变量对于某个特定的函数来说是私有的，并且与程序中其他的同名变量相区别。

5 递归　recursion: The term recursion refers to a situation in which a function calls itself.
递归是指一个函数自己调用自己的情况。

第七节　编译预处理

Section 7　Compiler Preprocessing

预处理命令　preprocessing directive
宏　macro
宏展开　expand a macro
标识符　identifier
标准路径　standard directory
当前路径　current directory
条件编译　conditional compilation
重定义　redefinition

1 预处理器　preprocessor: C provides certain language facilities by means of a preprocessor, which is conceptionally a separate first step in compilation. The two most frequently used features are # include, to include the contents of a file during compilation, and # define, to replace a token by an arbitrary sequence of characters.
C语言通过预处理器完成一些特定的功能，从概念上来说，预处理器是编译过程中单独执行的第一个步骤。两个经常被使用的预处理器是#include(用来在编译期间将指定的文件内容包含进当前文件)和#define(用一个任意字符序列替换一个标记)。

2 宏替换预处理命令　macro substitution preprocessor directive(# define): In C language, the # define preprocessor directive has two uses: creating symbolic constants and creating macros.
在C语言中，#define预处理命令有两个用途：创建符号常量和创建宏。

3 文件包含预处理命令　file inclusion preprocessor directive(# include): We use the # include preprocessor directive to include header files in our program. When it encounters an # include directive, the preprocessor reads the specified file and inserts it at the location of the directive.
我们使用#include预处理命令来把所需的头文件加入我们的程序中。当程序遇到#include预处理命令时，处理器就会读入特定的文件，并将它插入到预处理命令指定的位置。

第八节　指针

Section 8　Pointers

地址运算符　address operator(&)
间接引用运算符　indirection operator(*)
函数指针　function pointer
地址　address
指向　point to
直接存取　direct access
间接存取　indirect access
链表　linked list
节点　node
插入　insert
删除　delete
动态内存分配　dynamic memory allocation

1 指针　pointer: A pointer is the address of a variable.
一个变量的地址称为该变量的指针。

2 指针变量　pointer variable: A pointer variable is a variable that contains the address of another variable.
指针变量是一个存储其他变量地址的变量。

第九节 文件
Section 9 Files

二进制 binary
磁盘 disk
文本流 text stream
美国信息交换标准码 American Standard Code for Information Interchange(ASCII)
文件名 filename
缓冲区 buffer

1 文件 file: A file in a computer system is a stream of bits stored as a single unit, typically in a file system on disk or magnetic tape.
通常情况下,文件在计算机系统中是一个按位流存储的单位,它由文件系统管理并存储在磁盘或磁带上。

2 文件指针 file pointer: File pointer points to a structure that contains information about the file, such as the location of a buffer, the current character position in the buffer, whether the file is being read or written, and whether errors or the end of the file have occurred. Users don't need to know the details, because the definitions obtained from <stdio.h> include a structure declaration called FILE.
文件指针指向一个包含文件信息的结构,这些信息包括:缓冲区的位置、缓冲区中当前字符的位置、文件的读或写状态、是否出错或是否已经达到文件尾等等。用户不必关注这些细节,因为<stdio.h>中已经定义了一个包含这些信息的结构 FILE。