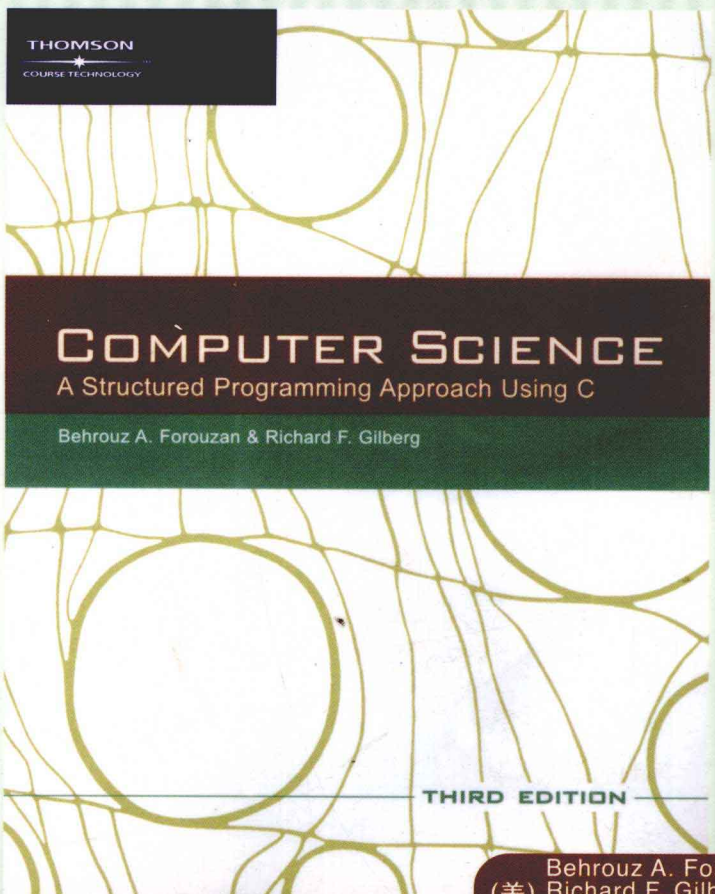
HZ BOOKS
经 典 原 版 书 库
THOMSON

计算机科学引论

基于C的结构化程序设计方法

（英文版 · 第3版）



Behrouz A. Forouzan
(美) Richard F. Gilberg 著
迪 安 那 大 学

机械工业出版社
China Machine Press

经 典 原 版 书 库

计算机科学引论

基于C的结构化程序设计方法

（英文版 · 第3版）

Computer Science

A Structured Programming Approach Using C

(Third Edition)

(美) Behrouz A. Forouzan
Richard F. Gilberg 著
迪 安 那 大 学

机 械 工 业 出 版 社
China Machine Press

Behrouz A. Forouzan and Richard F. Gilberg: Computer Science: A Structured Programming Approach Using C, Third Edition (ISBN 0-534-49132-4).

本书版权登记号：图字：01-2007-2429

图书在版编目（CIP）数据

计算机科学引论：基于C的结构化程序设计方法（英文版·第3版）/（美）佛罗赞（Forouzan, B. A.），（美）吉尔伯格（Gilberg, R. F.）著. -北京：机械工业出版社，2007.5

（经典原版书库）

书名原文：Computer Science: A Structured Programming Approach Using C, Third Edition

ISBN 978-7-111-21402-1

Ⅰ.计…　Ⅱ.①佛…　②吉…　Ⅲ.①计算机科学-英文　②C语言-程序设计-英文　Ⅳ.TP3

中国版本图书馆CIP数据核字（2007）第059389号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2007年5月第1版第1次印刷

145mm×210mm　·　37印张

定价：59.00元

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用

为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：（010）68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元　王　珊　冯博琴　史忠植　史美林
石教英　吕　建　孙玉芳　吴世忠　吴时霖
张立昂　李伟琴　李师贤　李建中　杨冬青
邵维忠　陆丽娜　陆鑫达　陈向群　周伯生
周克定　周傲英　孟小峰　岳丽华　范　明
郑国梁　施伯乐　钟玉琢　唐世渭　袁崇义
高传善　梅　宏　程　旭　程时端　谢希仁
裘宗燕　戴　葵

Preface to Third Edition

This text has two primary objectives: to teach the basic principles of programming as outlined in the ACM curriculum for a CS1 class and to teach the basic constructs of the C Language. Our text puts these objectives in the context of good software engineering concepts that we have developed through more than 30 years of experience in industry and academia.

Major Changes in Third Edition

This is a major change from the second edition. In addition to significant changes in the presentation and material covered, it also updates the language to the ISO/IEC 9899 (1999), or as it is more commonly known, C99.

While we have made major changes, we have held true to our primary objectives as outlined above. The text remains a comprehensive introduction to computer programming in a software engineering context.

The major changes are outlined in the following sections.

Standard C (1999)

The following topics, in alphabetical order, are explicitly discussed in the text.

- the Boolean type
- character set extensions (including Unicode and wide-characters)
- complex arithmetic
- extended math library
- *for* statement declarations
- integer type extensions
- line comments

Extended Chapter Material

In addition to the changes required by the inclusion of the new C99 extensions, several chapters have been revised or reorganized.

- A new section in Chapter 4, "Inter-Function Communication," provides a better understanding of how functions communicate.

- The concept of "incremental program development" is presented through an example that starts in Chapter 4 and is extended in Chapter 5. A second example in Chapter 8 completes the discussion.
- The discussion of text and binary files has been reworked to present the materials in a more logical, and less redundant, fashion.
- The presentation of structured types is changed to conform to C99 and common usage.
- Chapter 14, "Bitwise Operators," has been extended and includes more examples. In particular, we added several applications related to the Internet and network programming.
- Chapter 15, "Lists," has also been extended. It now includes an overview of the basic elements of data structures, including introductory discussions of stack, queues, lists, trees, and graphs.
- Additional flowcharts and structure charts are included throughout the text to provide better design considerations for some of the more complex examples.
- Appendix D, "Numbering Systems," provides an expanded presentation of the material.
- Appendix I, "Pointers to *void* and to Functions," has been expanded to include a discussion of using pointer to *void* types.
- Appendix K. "Program Development," is a new discussion of program development concepts.
- Appendix L, "Understanding Complex Declarations," contains the discussion on reading complex declarations, which has been removed from Chapter 10.

A C Language Perspective

While C is a complex and professional language, our classroom experience using the previous editions of this book has shown that beginning students can easily understand it. We believe that if the language is put into a perspective that allows the student to understand its design, C is not a difficult language.

There are two aspects of C that separate it from most other languages: expressions and pointers. The concept of expressions, as used in C, is unique. The first half of this text builds a firm understanding of expressions. We introduce pointers only to the extent necessary to cover passing them to functions in Chapter 4 and arrays in Chapter 8. Our experiences have shown that with a firm grasp of the expression concept, much of the mystery of C disappears.

In Chapters 9, we begin to develop the concept of pointers, which are further developed in Chapter 10. Chapter 11 provides a discussion of C's approach to strings. Finally, Chapter 14 provides an extensive discussion of bitwise operators, a subject that is needed to make the book complete.

The last chapter is a simple introduction to data structures. While not all courses will have time to cover this chapter, those that do will give the

students a head start into data structures. Motivated students will find that they can "get a leg up" on data structures over the term break.

The appendices at the end of the text comprise a rich set of references to subjects required in the complete C language. These will be of use to students who go on to take other courses in the C language.

Features of the Book

Several features of this book make it unique and easy for beginning students to understand.

Structure and Style

One of our basic tenets is that good habits are formed early. The corollary is that bad habits are hard to break. Therefore, we consistently emphasize the principles of structured programming and software engineering. Throughout each chapter, short code snippets and programs are used to demonstrate techniques.

Complete programs, generally found in the last section of the chapter, are written in a well-document consistent style. As programs are analyzed, style and standards are further explained. While we acknowledge that there are many good styles, our experience has shown that if students are exposed to one good style and adopt it, they will be better able to adapt to other good styles.

Principle Before Practice

Whenever possible, we develop the principle of a subject before we introduce the language implementation. For example, in Chapter 5 we first introduce the concept of logical data and selection and then we introduce the *if . . . else* and *switch* statements. This approach gives the student an understanding of selection before introducing the nuances of the language.

Visual Approach

A brief scan of the book will demonstrate that our approach is visual. There are more than 400 figures, plus more than 70 tables and 180 program examples. While this amount of material tends to create a large book, the visual approach makes it easy for students to follow the material.

Examples

While the programming examples vary in complexity, each uses a consistent style. Our experience working with productional programs that live for 10 to 20 years convinced us that readable and understandable programs are easier to work with than programs written in a terse, cryptic manner. For that reason,

and to emphasize the structure of the language, we label the sections in a function with comments. We consistently follow a style that places only one declaration, definition, or statement on a line.

The source code for these programs is available on the Thomson Course Technology Web site (see "Web Site Support Materials" later in this Preface). Students are able to run these programs and to explore related topics through modification and experimentation.

Coding Techniques

Throughout the text we include coding techniques that make programs more readable and often more efficient. For example, in the analylsis of Program 5-4 you will find the following discussion:

> Where do we check for *greater than*? The answer is that we default the *greater than* condition to the outer *else*. . . . When coding a two-way selection statement, try to code the most probable condition first; with nested selection statements, code the most probable first and the least probable last.

These techniques are drawn from our extensive industry and classroom experience.

Software Engineering

A discussion of software engineering principles concludes each chapter. Our intent is not to replace a separate course in software engineering. Rather, we firmly believe that by incorporating basic software engineering principles early in their studies, students will be better prepared for a formal treatment of the subject. Even more important, by writing well-engineered programs from the beginning, students will not be forced to unlearn and relearn. They will better understand software discussions in their subsequent classes.

While the software engineering sections are found at the end of each chapter, they are most successfully taught by introducing them as the chapter unfolds. Then, a short review at the end of the chapter summarizes the principles that have been demonstrated during the lectures.

These sections are visually distinguishable from the rest of the chapter. They have been set apart for several reasons. First, they are in reality a small book within a book. While these sections contain important material, the book stands on its own without them. You may, therefore, decide to cover the software engineering sections as formal lecture topics or informally while the chapter material is being covered. You may decide to assign them to the student as additional reading, or, you may decide to exclude them entirely from the course.

In general, software engineering sections directly or indirectly pertain to the chapter material. Where they don't, they discuss general software engineering subjects, such as cohesion, coupling, and quality.

Pedagogical Material

End chapter material meets two pedagogical objectives: first, it helps the students to review or summarize what they have learned, and second, it tests the students' mastery of the chapter material.

Review Material

- **Tips and Common Programming Errors** points out helpful hints and possible problem areas.
- **Key Terms** provides a list of the important terms introduced in the chapter.
- **Summary** contains a concise overview of the key points for students to understand in the chapter.

Practice Sets

- **Review Questions** contain several multiple choice questions that are similar to the questions found in the examination database. The answers to the odd-numbered questions are included in the solutions on the Course Technology web site.
- **Exercises** are short questions covering the material in the chapter. The answers to the odd-numbered exercises are also included in the solutions on the Course Technology web site.
- **Problems** are short coding problems, generally intended to be run on a computer. They can usually be developed in two to three hours. Once again, odd-numbered solutions are found on the web site.
- **Projects** are longer, major assignments that may take the average student six to nine hours to develop.

Appendices

The appendices are intended to provide quick reference material, such as the Unicode Character Set, or to provide a review of material, such as numbering systems, usually covered in a general computer class.

Web Site Support Materials

Included on the Thomson Course Technology Web site (http://www.course.com) are two sets of materials, one for the professor and one for the student.

Professor Materials

The professor's materials include the solutions to all review questions, exercises, and problems. Because the projects are more complex assignments for which there is no standard answer, we do not provide solutions for them.

In addition to the end-material solutions, there are three other sets of support materials: (1) Copies of all programs in the text. The programs have been extensively tested on at least two different computers with different operating systems. All of them conform to the C99 Standard.

(2) PowerPoint materials covering objectives, figures, programs, important points, and chapter summaries. The PowerPoint materials can be used as is or edited to suit individual class needs. This flexibility allows professors to argument the text materials with their own. It also allows them to rearrange the materials to suit their individual needs and style.

(3) New with the third edition is the availability of ExamView®. This objective-based test generator lets the professor create paper, LAN, or Web-based tests from testbanks designed specifically for this text. Using the QuickTest Wizard, they can easily and quickly create true/false and multiple-choice tests. It is also possible to add questions that cover supplemental material provided by the professor.

Student Materials

The student materials include the solutions to the odd-numbered review questions, exercises, and problems. They also have access to the copies of the programs used in the text.

Acknowledgments

No book of this scope can be developed without the support of many people.

Reviewers

To anyone who has not been through the process, the value of peer reviews cannot be appreciated enough. Writing a text rapidly becomes a myopic process. The important guidance of reviewers who can stand back and review the text as a whole cannot be measured. We would especially like to acknowledge the contributions of the reviewers of all three editions.

Stephen Allen, *Utah State University*
Mary Astone, *Troy State University*
Ali Behforooz, *Towson State University*
George Berry, *Wentworth Institute of Technology*
Ernest Carey, *Utah Valley State College*
Ping Chu Chu, *Fayetteville State University*

Constance Conner, *City College of San Francisco*
John S. DaPonte, *Southern Connecticut State University*
Maurice L. Eggen, *Trinity University*
Peter Gabrovsky, *CSU Northridge*
Robert Gann, *Hartwick College*
Henry Gordon, *Kutztown University*
Rick Graziani, *Cabrillo College*
Barbara Guillott, *Louisiana State University*
Jerzy Jaromczyk, *University of Kentucky*
John Kinio, *Humber College*
Roberta Klibaner, *College of Staten Island*
Joseph A. Konstan, *University of Minnesota*
Krishna Kulkarni, *Rust College*
John Lowther, *Michigan Technological University*
Mike Michaelson, *Palomar College*
Kara Nance, *University of Alaska—Fairbanks*
Ali Nikzad, *Huston-Tillotson College*
Jo Ann Parikh, *Southern Connecticut State University*
Mark Parker, *Shoreline Community College*
Savitha Pinnepalli, *Louisiana State University*
Oskar Reiksts, *Kutztown University*
Jim Roberts, *Carnegie Mellon University*
Ali Salenia, *South Dakota State University*
Larry Sells, *Oklahoma City University*
Shashi Shekhar, *University of Minnesota*
Robert Signorile, *Boston College*
Brenda Sonderegger, *Montana State University*
Deborah Sturm, *College of Staten Island*
Venkat Subramanian, *University of Houston*
John B. Tappen, *University of Southern Colorado*
Marc Thomasm, *California State University, Bakersfield*
John Trono, *St. Michael's College*
K.C. Wong, *Fayetteville State University*

Course Technology Staff

Our thanks to our editors, Alyssa Pratt, Senior Product Manager, and Mary Franz, Senior Acquisitions Editor, for helping us produce the book. We are also indebted to the Quality Assurance staff who diligently double-checked each chapter and program. Our thanks to Burt LaFountain, Serge Palladino, and Chris Scriver.

We would also like to thank the staff at GEX Publishing Services, especially Sandra Mitchell, who ably guided the book through production.

Behrouz A. Forouzan
Richard F. Gilberg

Operator	Description	Example	*	Assoc	Pr

	Identifiers
	Constants
	Parenthetical Expressions
		`amount`
		`3.14159`
		`(a + b)`
			N	N/A	16

`[ ]`	Array Index	`ary[i]`	N		
`f(...)`	Function Call	`doIt(x, y)`	Y		
`.`	Direct Member Selection	`str.mem`	N	Left-Right	16
`->`	Indirect Member Selection	`ptr->mem`	N		
`++ --`	Postfix Increment • Decrement	`a++`	Y		
`++ --`	Prefix Increment • Decrement	`++a`	Y		
`sizeof`	Size in Bytes	`sizeof(int)`	N		
`~`	Ones Complement	`~a`	N		
`!`	Not	`!a`	N	Right-Left	15
`+ -`	Plus • Minus	`+a`	N		
`&`	Address	`&a`	N		
`*`	Dereference / Indirection	`*ptr`	N		
`( )`	Type Cast	`(int)ptr`	N	Right-Left	14
`* / %`	Multiply • Divide • Modulus	`a * b`	N	Left-Right	13
`+ -`	Addition • Subtraction	`a + b`	N	Left-Right	12
`<< >>`	Bit Shift Left • Bit Shift Right	`a << 3`	N	Left-Right	11
`< <= > >=`	Comparison	`a < 5`	N	Left-Right	10
`== !=`	Equal • Not Equal	`a == b`	N	Left-Right	9
`&`	Bitwise And	`a & b`	N	Left-Right	8
`^`	Bitwise Exclusive Or	`a ^ b`	N	Left-Right	7
`|`	Bitwise Or	`a | b`	N	Left-Right	6
`&&`	Logical And	`a && b`	N	Left-Right	5
`||`	Logical Or	`a || b`	N	Left-Right	4
`? :`	Conditional	`a ? x : y`	N	Right-Left	3

`= += -=`
`*= /= %=`
`>>= <<=`
`&= ^= |=`
	Assignment	`a = 5`
		`a %= b`
		`a &= c`
		`a |= d`
			Y	Right-Left	2

`,`	Comma	`a, b, c`	N	Left-Right	1

* Side Effects (Yes / No)

Precedence Table

Argument Type	Size Specifier	Code
integral	hh (char), h (short), none (int), l (long), ll (long long)	i
integer	h (short), none (int), l (long). ll (long long)	d
unsigned int	hh (char), h (short), none (int), l (long), ll (long long)	u
character octal	hh (unsigned char)	o
integer hexadecimal	h (short), none (int), l (long), ll (long long)	x
real	none (double), l (double), L (double)	f
real (scientific)	none (double), l (double), L (double)	e
real (scientific)	none (double), l (double), L (double)	g
real (hexadecimal)	none (double), l (double), L (double)	a
character	none (char), l (wchar_t))	c
string	none (char string), l (wchar_t string)	s
pointer		p
integer (for count)	none (int), hh (char), h (short), l (long), ll (long long)	n
set	none (char), l (wchar_t)	[

Sizes, and Conversion Code for *scanf* Family

Argument Type	Flag	Size Specifier	code
integer	-, +, 0, space	hh (char), h (short), none (int), l (long), ll (long long)	d, i
unsigned integer	-, +, 0, space	hh (char), h (short), none (int), l (long), ll (long long)	u
integer (octal)	-, +, 0, #, space	hh (char), h (short), none (int), l (long), ll (long long)	o
integer (hex)	-, +, 0, #, space	hh (char), h (short), none (int), l (long), ll (long long)	x, X
real	-, +, 0, #, space	none (double), l (double), L (double)	f
real (scientific)	-, +, 0, #, space	none (double), l (double), L (double)	e, E
real (scientific)	-, +, 0, #, space	none (double), l (double), L (double)	g, G
real (hexadecimal)	-, +, 0, #, space	none (double), l (double), L (double)	a, A
character	-	none (char), l (w-char)	c
string	-	none (char string), l (w-char string)	s
pointer			p
integer (for count)		none (int), h (short), l (long)	n
to print %			%

Flags, Sizes, and Conversion Codes for *printf* Family

Contents