HZ BOOKS
华章科技
PEARSON
名家经典系列

由在IBM工作50余年的资深计算机专家撰写，Amazon全五星评价，算法领域最有影响力的著作之一

Google公司首席架构师、Jolt大奖得主Joshua Bloch和Emacs合作创始人、C语言畅销书作者Guy Steele倾情推荐

算法的艺术和数学的智慧在本书中得到了完美体现，书中总结了大量高效、优雅和奇妙的算法，并从数学角度剖析了其背后的原理

算法心得

高效算法的奥秘

（原书第2版）

（美）Henry S. Warren，Jr. 著
爱飞翔 译

Hacker's Delight
(Second Edition)

机械工业出版社
China Machine Press

名家经典系列

算法心得

高效算法的奥秘

（原书第2版）

（美）Henry S. Warren，Jr. 著
爱飞翔 译

Hacker's Delight

(Second Edition)

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

算法心得：高效算法的奥秘（原书第2版）/（美）沃伦（Warren，Jr.，H.S.）著；爱飞翔译．—北京：机械工业出版社，2014.1

（名家经典系列）

书名原文：Hacker's Delight，Second Edition

ISBN 978-7-111-45356-7

Ⅰ.算… Ⅱ.①沃… ②爱… Ⅲ.电子计算机—算法理论 Ⅳ.TP301.6

中国版本图书馆CIP数据核字（2014）第002804号

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2013-0212

Authorized translation from the English language edition, entitled *Hacker's Delight, Second Edition*, 9780321842688 by Henry S. Warren, Jr., published by Pearson Education, Inc., Copyright © 2013.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Chinese Simplified language edition published by Pearson Education Asia Ltd., and China Machine Press Copyright © 2014.

本书中文简体字版由Pearson Education（培生教育出版集团）授权机械工业出版社在中华人民共和国境内（不包括中国台湾地区和中国香港、澳门特别行政区）独家出版发行。未经出版者书面许可，不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

本书是算法领域最有影响力的著作之一，与大师高德纳所著的《计算机程序设计艺术》共同被誉为所有程序员都应该阅读的计算机著作。它由在IBM工作50余年的资深计算机专家撰写，Amazon全五星评价，Google公司首席架构师、Jolt大奖得主Hoshua Bloch和Emacs合作创始人、C语言畅销书作者Guy Steele倾情推荐。书中总结了大量高效、优雅和奇妙的算法，并从数学角度剖析了其背后的原理，算法的艺术和数学的智慧在本书中得到了最好的体现。

本书共18章。第1章是概述；第2章介绍了基础知识；第3～4章介绍了2的幂边界和算术边界；第5章讨论了位计数；第6～7章讲解了在字组中搜索位串和重排位元与字节；第8～10章分别介绍了乘法、整数除法和以除数为常量的整数除法；第11章讲解了初等函数；第12章介绍了以特殊值为底的数制；第13～15章分别讲解了格雷码、循环冗余校验和纠错码；第16～18章分别介绍了希尔伯特曲线、浮点数和素数公式。之后是各章习题的参考答案。附录分别介绍了计算机算术运算表、牛顿法和各种离散函数图像。

机械工业出版社藏書 图書馆

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：关 敏

北京市荣盛彩色印刷有限公司印刷

2014年3月第1版第1次印刷

186mm×240mm·27.25印张

标准书号：ISBN 978-7-111-45356-7

定 价：89.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991 88361066 投稿热线：（010）88379604

购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

译 者 序

写代码总会遇到难题，时而苦于乘法操作频繁溢出，时而苦于开方算法太过笨拙，于是，程序员之间口耳相传的那些代码秘籍，这些时候就该大显身手了。有些小程序，仅两三行代码即能解决平常数十行代码方能实现的功能；还有些小程序，只用0x24924925这般神奇的数字，即能成倍提升运算速度。读者若对此感兴趣，则本书定能令你开怀畅读。

作者从事计算机研发工作数十年，他将期间所得之大量技巧融于书中。本书不但讲授算法技巧，而且还会剖析背后的数学原理，令你在学会某个奇妙算法后，可举一反三，推出很多类似技巧，以运用于不同场合。

在研究这些高效而优雅的算法时，作者还会如数家珍地列出许多变体，并旁征博引地讲述可以解决同一问题的其他思路，铺陈完毕后，更会将各自优劣娓娓道来。实际应用中，经常需要权衡各算法之轻重，嵌入式开发、硬件编程、图形渲染、游戏智能等领域尤其如此，若是平素能像作者这样勤于总结、善于对比，那么在需要用到相关技巧时必能信手拈来，左右逢源。

从培养兴趣、锻炼思维、付诸实践三个角度观之，本书皆为精彩而思辨的智慧书。既可静心品读代码之诗意，又能细致体味数学之美感，何其乐哉！

作者乃业界翘楚，学识渊博而思维开阔，文中部分词句与日常用语及数学、计算机等领域一般用法不甚相同，故译文或加注释或添引号，以强调其特殊含义。

翻译过程中，得到机械工业出版社华章公司诸君勉励，于此深表谢意。

本书主要由爱飞翔翻译，舒亚林、张军、王鹏亦参与部分翻译工作。小弟乐意与各位朋友通过个人网站（www.agilemobidev.com）及电子邮件（eastarstormlee@gmail.com）探讨算法问题。由于时间仓促，水平有限，错误与疏漏在所难免，敬请读者不吝赐教。

爱飞翔

2014年2月

序（第 1 版序）

三十年前的一个暑假，我在麻省理工学院的 Project MAC 实验室㊀找到了首份工作。当时很喜欢操作那里的 DEC PDP-10 计算机㊁。这台计算机有丰富且易于操控的指令集，可以做位测试和位屏蔽，并操作位段㊂与整数，所以和其他计算机相比，拿它写汇编语言代码绝对要更顺手些。尽管已经停产多年，仍有一批狂热的粉丝还在玩 PDP-10，还有一些人拿家用电脑模拟 PDP-10 指令集，这样就可以执行以前的整个 PDP-10 操作系统及其附属软件了。这帮人甚至还在该平台上开发新的程序，比如说现在还有些网站用模拟的 PDP-10 平台来提供网页服务。（拜托，大家别笑，与那些爱开老爷车的人相比，这种怀旧方式也不算什么。）

1972 年夏天，还有一件事让我开心，那就是读到了一本全新的 MIT 研究备忘录：HAKMEM。这部秘籍收录了从电路到数论的各种技术细节㊃，而最令我着迷的内容还是其中各项精妙的编程小技巧。通常每一条编程技巧都会讲一个貌似符合编程规范但是看上去却十分怪异的操作。这些整数或位串操作（bit string，例如计算某个字㊄中值为 1 的

㊀ “数学与计算研究计划”（Project on Mathematics and Computation）一词的缩写，是 1963 年设立于麻省理工学院（Massachusetts Institute of Technology，MIT）的一个研究项目，为当前“MIT 计算机科学与人工智能实验室”（MIT Computer Science and Artificial Intelligence Laboratory）的前身。详情参见：https://en.wikipedia.org/wiki/Project_MAC#Project_MAC。——译者注

㊁ DEC 公司（Digital Equipment Corporation）于 20 世纪 60 至 80 年代推出的一系列大型计算机。详情参见：https://en.wikipedia.org/wiki/PDP-10。——译者注

㊂ field，在这里指 bit field，也叫“位域”，是一种数据结构，可将各个字段用若干个位元的形式紧密存储，以节省空间。详情参见：https://zh.wikipedia.org/wiki/位段。——译者注

㊃ “HAKMEM”是“hacks memo”的简称，为什么这样写呢？因为 PDP-10 平台的字（word）长度为 36 位，而每个字符占 6 个二进制位，这样，一个字就能容纳 6 个字符，所以很多 PDP-10 程序员用的名字都是 6 个字符，我们这帮人一看到 6 字符的缩写，立马就能还原出它的原意。所以，“HAKMEM”这个怪名字在那个时代是合理的，至少对程序员来说是这样。

㊄ word，在计算机领域里指由一定数量的二进制位（bit，比特）所构成的数据结构。为了与日常汉语中的“字”区别开，译文多以“字组”称之。详情参见：https://zh.wikipedia.org/wiki/字_（计算机）。此外，为了与十进制的“数位”（digit）和常用的“位置”等词相区分，译文将酌情以“位元”一词对应 bit。——译者注

此为试读，需要完整PDF请访问：www.ertongbook.com

二进制位共有多少个），本来用一个较长但是约定俗成的机器指令序列或是循环就能轻松实现，然而秘籍中却用了一种极为巧妙的办法来完成：它只用三四个甚至两个精心选择的指令就够了。这些指令的意思很隐晦，需要有人给你解释，或是自己仔细研究才能搞清楚。品尝这份编程大餐，与吃花生豆或夹心软糖那种小零食一样，也令人欲罢不能。这些技巧内涵丰富，充满了深邃的智慧，体现了程序的美感，甚至让人读出了诗意。

我心里当时就在想："这种技巧不会只有这么几条吧？"我把这些年来在各种场合发现的编程心得总结了一下，果然数量非常多。所以我又想，"是不是该有一本专著来谈谈这个问题呢？"

看到 Hank Warren 先生的原稿时，我实在太高兴了：他将这些编程小技巧系统地收集起来，按主题分类，并清晰地讲解其意义。虽说有些技巧是用机器指令描述的，但是这本书可不是只写给汇编语言程序员看的。本书主题是探讨计算机中整数与位串的基本结构关系，以及如何才能更为高效地操作它们。汇编语言中能用的技巧，放在 C 或 Java 语言里照样适用。

很多算法和数据结构的书都在讲排序和搜索这些复杂的技术，告诉你如何维护哈希表和二叉树，怎样操作记录和指针等，然而它们都没有讲到二进制位及位元数组这种微型数据结构的功用。单用二进制加减法和位操作就能实现很多强大的功能。由于进位链（carry chain）机制，改变一个位元的值有可能会影响到它左方的所有二进制位。某些利用此机制的二进制加法技巧不为人熟知，然而它们却是操作数据的利器。

的确该有一本书来总结这些技巧了。你手中捧着的这本就是，而且写得非常好。要想优化编译器或编写高效代码，那非读它不可。也许这些技巧不是每天都能用到，但在遇到困难时，还是会派上用场的。有时我们要遍历一个字中的位元，有时要执行某些不太好实现的整数操作，有时为了让运行速度加倍，还要优化内循环[⊖]中那些棘手的整数或位元运算——如果真碰到上述情况，那就得求助此书啦。当然了，有时也不必想那么多，直接翻开来读，就必能体会到个中乐趣。

Guy L. Steele，Jr.

2002 年 4 月于马萨诸塞州伯灵顿

⊖ inner loop，指两重循环中内部的那一层，通常优化程序时应首先考虑减少内层循环体的执行时间。详情参见：https://en.wikipedia.org/wiki/Inner_loop。——译者注

前　言

程序员创造力第一定律：软件维护成本与程序员创造力的平方成正比。

——Robert D. Bliss，1992 年

本书收集了笔者多年来总结的一些编程小技巧，其中大部分都必须运行在以二进制补码来表示整数的电脑[㊀]上。尽管本书假设寄存器长度是 32 位，但在寄存器长度不是 32 位的电脑中，这些技巧基本上仍能适用。

本书不打算讲高深的排序算法或是编译器的优化技术等大话题，而是着重来谈涉及单个计算机字组或指令的小技巧，比如怎样统计字组中值为 1 的位元数。此类技巧通常需要混用算术与逻辑指令。

我们还需假定整数溢出[㊁]中断已经屏蔽，这样的话，就算整数运算溢出了，也不会出问题。C、Fortran、Java 等程序都是如此，但使用 Pascal 与 Ada 语言的程序员一定要注意这一点！

书中以通俗的语言来讲述各技巧，只有算法含义不明显时才会给出证明，有时甚至略去证明。算法中将会出现计算机算术指令、“地板”函数[㊂]、算术与逻辑操作混搭等内容，它们要证明起来通常比较困难，而且也不易表述。

笔者把很多算法都写成了程序代码，在电脑上执行并验证，以减少笔误及疏忽。用实际存在的编程语言来描述算法，就有这个好处；然而即便如此，每一种计算机语言也还是各有其缺陷。我使用很多人能看懂的 C 代码来描述高级语言部分，因为它可以直接

㊀ 在本书语境中，电脑、计算机（computer）、机器（machine）、处理器（processor）等词常常都指代中央处理器（CPU），故译文在不致混淆的情况下，将其视为互相通用的概念。——译者注

㊁ integer overflow，指算术运算时由于结果过大或过小而超出存储范围的情形。详情参见：https://en.wikipedia.org/wiki/Integer_overflow。——译者注

㊂ floor function，又叫向下取整函数或最低值函数，返回不大于自变量的最大整数，也常称为“高斯函数”。详情参见：https://zh.wikipedia.org/wiki/地板函数。——译者注

把整数与位串操作写在一起，而且很多 C 语言编译器都能产生高质量的目标代码㊀。

书中偶尔也会用到机器语言，它们都以“三地址格式”㊁出现，这样读起来更方便。其中的汇编语言是笔者参照当下的 RISC 指令集㊂虚构出来的。

大家应该尽量编写没有分支的代码（branch-free code），因为分支代码会拖慢很多电脑的指令获取速度，并阻止 CPU 并发执行。此外，它还会妨碍一些编译器优化措施㊃，如指令调度㊄、重复子表达式消除㊅、寄存器分配㊆等。也就是说，编译器优化大段的简单代码要比优化很多小代码块的效果更好。

此外，编码时还应该多用数值较小的常量㊇，多与 0 比较（少与非 0 值比较），尽量写出易于并发执行的指令序列㊈。如果改写书里很多代码，让它们从内存中查表，其结果会更精确，然而笔者通常不提这种做法。因为与算术指令相比，从内存中加载数据反而更耗时。虽说查表法的确很实用，但是一般来说都比较乏味。当然还有些特例不属于上述情况。

最后要说的是，本书书名㊉中的“hacker”用的是本意，也就是指痴迷于计算机的人。这种人喜欢拿电脑开发点新玩意儿，或是用新潮而有创意的办法来实现原有的功能。

㊀ object code，又叫“目的码”，是编译器处理源代码后的产物，一般由机器码或近于机器语言的代码组成。存放此类代码的文件叫目标文件（object code），也叫二进制文件。详情参见：https://zh.wikipedia.org/wiki/目标代码。——译者注

㊁ three-address format，即 three address code，缩写为 TAC 或 3AC，又叫“三位址码”、“三地址码”。它把通常的算式“运算结果＝操作数 1 操作符 操作数 2”写成“操作符 操作数 1，操作数 2，运算结果”的形式，因其中牵涉 3 个变量（即操作数 1、操作数 2、运算结果）而得名。例如 $z=x+y$ 用 TAC 格式来写就是 add x，y，z。详情参见：https://en.wikipedia.org/wiki/Three_address_code。——译者注

㊂ RISC，Reduced Instruction Set Computing 的缩写，是一种设计 CPU 的模式，该设计思路精简了指令数目及定址方式，使 CPU 的制作更为容易，也有助于提升其并行能力与执行效率。使用此种指令集的 CPU 目前主要应用于智能手机和平板电脑等移动设备上，与之相对的是 x86 等处理器所用的“复杂指令集”（CISC，C 表示 Complex）。详情参见：https://zh.wikipedia.org/wiki/精简指令集。——译者注

㊃ 常见的编译器优化技法请参考：https://en.wikipedia.org/wiki/Category:Compiler_optimizations。——译者注

㊄ instruction scheduling，是通过指令流水线（instruction pipeline）技术提升指令并发执行效果的优化手法。详情参见：https://en.wikipedia.org/wiki/Instruction_scheduling。——译者注

㊅ commoning，即 common subexpression elimination（CSE），是通过消减重复的表达式来提升执行速度的优化技术。详情参见：https://en.wikipedia.org/wiki/Common_subexpression_elimination。——译者注

㊆ register allocation，该技术让众多变量共用一个 CPU 寄存器，以优化执行速度。详情参见：https://en.wikipedia.org/wiki/Register_allocation。——译者注

㊇ immediate value，也叫“立即值”。——译者注

㊈ instruction-level parallism，缩写为 ILP，意为指令级并行度，用于度量指令序列的并发执行程度。例如某段指令在非并发方式下需要 3 个时间单位来执行，而并发执行只需 2 个时间单位，那么其 ILP 就是 3/2。详情参见：https://en.wikipedia.org/wiki/Instruction-level_parallelism。书中还用该词表示 CPU 并发执行指令的能力。——译者注

㊉ 本书英文书名为 Hacker's Delight。——译者注

他们都挺会用电脑，但却很可能不是一名专职电脑程序员或设计师，这些电脑极客[㊀]开发出来的东西，有些有用，有些只是玩玩而已。说到此类纯属玩乐的戏作，让我想起很多资深编程票友都写过一段小程序，它可以把自己照原样打印出来[㊁]。这才是我使用“hacker”一词的初衷，在书里可别想找到教你怎样入侵他人电脑的把戏哦。

致谢

首先要感谢 Bruce Shriver 与 Dennis Allison 两位先生鼓励我写作本书，然后还要对 IBM 诸位同仁致意，参考文献中也提到了其中一些同事的名字。尤其感谢 Martin E. Hopkins 先生，他可谓 IBM 的“活编译器”(Mr. Compiler)。Hopkins 先生在编码工作中一丝不苟，认真优化每一个循环——这种敬业精神深深感染了我。仰赖 Addison-Wesley 各位评审，本书内容有了极大改观。他们的姓名笔者大多不了解，我唯一记住的是大名鼎鼎的 Guy L. Steele, Jr.。在 50 页书评中，Guy 补充了一些我当时没有谈到的新问题，如位元的打乱与复原，“绵羊与山羊分离操作”(分羊法，sheep and goats operation)，等等，而且他提出的一些算法也比我原来用的更高明。Guy 是个非常仔细的人，比方说，我曾把十六进制数 AAAAAAAA 的质因子分解式错写为 2×3×17×257×65 537，而他发现其中的 3 应该是 5。此外，他还提了一些旨在改善写作风格的建议，并且从不回避书里的细节问题。读者若发现文中有疑似他人捉刀之处[㊂]，那恐怕得归功于 Guy 了。

H. S. Warren, Jr.

2012 年 6 月于纽约州约克镇

本书英文版相关资料请查阅
www. HackersDelight. org

㊀ 本书语境中的“hacker”可理解成“电脑极客”、“程序玩家”、“编程票友”、“编程达人”、“技术发烧友”等意思，以便与容易引发歧义的“骇客”、“黑客”等通俗叫法区分开。本来 hacker 指计算机技术狂热爱好者，而 cracker 指破坏计算机安全的人，但后者的词义逐渐侵蚀了前者，导致 cracker、hacker 及“骇客、黑客、怪客、刽客”等译名全都成了“计算机安全破坏者”，令 hacker 一词丧失了原意。详情参见：https://zh. wikipedia. org/wiki/骇客。——译者注

㊁ 比方说，下面这段用 C 语言写的程序代码：
main () {char * p="main () {char * p=%c%s%c; (void) printf (p, 34, p, 34, 10);}%c"; (void) printf (p, 34, p, 34, 10);}

㊂ 原文为 parallel prefix，意为“并行前置式算法”、“平行前缀算法”，是一套分组归并形式的速算流程。详情参见：https://www. wikipedia. org/wiki/Prefix _ sum。作者在此处以之比喻他与 Guy 四手联弹的佳话。——译者注

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章
概述

1.1 记法

本书既不是普通的数学算式教程，也不是单纯的电脑程序算法手册，它讲的是“计算机算术”（computer arithmetic），参与运算的数是长度固定的位串与位元数组[㊀]。计算机算术中的表达式与普通的数学表达式近似，不同之处在于其中的变量指代的是 CPU 寄存器中的内容，而且计算机算术表达式的值是一串不具备特定符号性的位元。在此类表达式中，操作符可能会以不同方式解读其操作数，例如“比较操作符”（comparison operator）有时会将其操作数当成有正负号的二进制整数，有时却把它视为无符号的二进制整数。为免混淆，本书所列计算机算式以不同的符号来区分上述两种情况。

计算机算式与数学算式的主要差别在于：无论是加减法还是乘法，计算机算式的结果总是要跟 2 的 n 次方取模，n 是指当前机器的字长[㊁]。此外，计算机算式的运算种类也远多于数学算式：除了基本的四则运算外，还有逻辑与、异或、比较、左移，等等。

如未特别指明，则默认字长为 32 位，且带符号的整数均以“2 补码”[㊂]形式表示。

㊀ bit vector，即 bit array，也写作 bitmap、bitset、bit string 等，是由若干位元排列而成的数组，又叫“位向量”、“位矢量”等，详情参见：https://en.wikipedia.org/wiki/Bit_array。译文在不致混淆时，均以“位元数组”称之。——译者注

㊁ word size，就是字组中所含的位元个数，也称 word length、word width。在不致混淆的情况下，“字长”、“字宽”均指这一概念。——译者注

㊂ two's complement，也叫“2 的补码”、“二补数”，是一种用二进制表示带符号数字的方式。整数和零的补码表示法与其二进制写法相同，只是左方要补足 0，而要表示负数，则需先将其绝对值按位取反，也就是求绝对值的一补数（也称反码），然后再加 1。例如用 8 位二进制表示数字时，5 可以表示为 0000 0101，而要表示 −5，则需先求其绝对值 5，写成二进制形式 0000 0101，然后对其按位取反得到 1111 1010，再加 1。最终的结果 1111 1011就是 −5 的补码表示。详情参见：https://zh.wikipedia.org/wiki/补码。带符号整数与无符号整数的英文分别为 signed integer 与 unsigned integer，其中“带符号”与“无符号”是计算机处理二进制数的两种不同方式。前者会根据最高有效位来判定数字的正负，0 为非负，1 为负，而后者则一律将其视为非负数。以上面的 1111 1011 来说，若视为带符号整数，则其值为 −5（因最高位为 1，故是负数。先将其按位取反得 0000 0100，再加 1 得 0000 0101，即十进制的 5，再添上负号得 −5），若视为无符号整数，则其值为 251（最高位的 1 不再表示负数，而当做 128 来算）。详情参见：https://zh.wikipedia.org/wiki/有符号数处理。——译者注

计算机算式与数学算式的书写方式相同，只是其中指代CPU寄存器中内容的变量会以粗体标出。为了和位元数组运算的通则一致，我们把电脑中的字组也视为由一串位元构成的数组。若某常量表示CPU寄存器中的值，那么它也会以粗体字出现。（这种情况在位元数组运算中找不到对应物，如果要在位元数组算式里书写常量，只能逐个列出其中的每个位元。）然而常量如果作为shift等指令的操作数，则不加粗。

对于“+”这样的操作符，若其操作数为粗体，则表明执行的是计算机加法，也就是“向量加法”，否则意味着执行的是纯数字加法。如果操作数未加粗，则其对应的操作符就是纯数学运算中的含义。要是我们想拿原来做计算机运算的粗体 $\boldsymbol{x}$ 值做数学运算的话，那就会把它写成不加粗的 x，其符号性应该能够从上下文中推出。假如 $\boldsymbol{x}=$ **0x8000 0000**，$\boldsymbol{y}=$**0x8000 0000**，那么在做带符号的整数运算时，$\boldsymbol{x}=\boldsymbol{y}=-2^{31}$，$x+y=-2^{32}$，而 $\boldsymbol{x}+\boldsymbol{y}=\boldsymbol{0}$[㊀]。此处的 **0x8000 0000** 是用十六进制表示的位串，它最左边的位元是“1”，后面跟着31个“0”。

位元的序号从右侧算起，最右方的位元（也就是最低有效位）叫做0号位元。术语“位”、“半字节”、“字节”、“半字组”、“字组”、“双字”[㊁]所对应的位元数量分别是1、4、8、16、32、64。

简短的代码段用的都是计算机算式，并以左箭头表示赋值操作，偶尔还会用if语句。在这种情况下，它只是以一种与电脑平台无关的方式编写汇编语言代码罢了。

过长或过于复杂的计算机算式则用C语言来写，其代码遵循ISO 1999标准[㊂]。

完整描述C语言不是本书该做的事，不过书中用到的大部分C语言基本表达式[H & S]都总结到表1.1中了，用过程序语言编程但是不熟悉C语言的读者应该看看。表中也列出了笔者在计算机算式中用到的对应操作符，它们按照优先级从高到低的顺序排列（高者先算），在优先级这一列中，L表示左结合，比如乘号就是左结合的运算符：a·b·c=(a·b)·c，而R则表示右结合[㊃]。本书计算机算式里的运算符，其优先级与结合性同C语言一致。

表1.1 C语言与计算机算术表达式对照表

优先级	C	计算机算式	含义
	0x...	**0x...，0b...**	十六进制常数，二进制常数

㊀ 不加粗的 $x+y$ 是普通的数学运算，故结果为 -2^{32}，而粗体的 $\boldsymbol{x}+\boldsymbol{y}$ 则是按照计算机算术规则做二进制加法，其结果为1 00000000 00000000 00000000 00000000，也就是1后面32个零。宽度为32的字组无法容纳这个33位元的数，故而最高有效位的“1”会被舍去，留下32个0，因此最终的运算结果就是0。——译者注

㊁ 对应英文写法分别为：bit、nibble、byte、halfword、word、doubleword。——译者注

㊂ 该标准的正式名称是ISO/IEC 9899：1999，俗称C99，详情参见：https://zh.wikipedia.org/wiki/C语言#C99。C语言的最新标准是2011年发布的ISO/IEC 9899：2011，俗称C11。——译者注

㊃ 左结合与右结合的英文分别是left-associative与right-associative，结合性指的是遇到两个相邻的同优先级运算符时，要从左先算还是从右先算。详情参见：https://en.wikipedia.org/wiki/Operator_associativity。——译者注

此为试读，需要完整PDF请访问：www.ertongbook.com