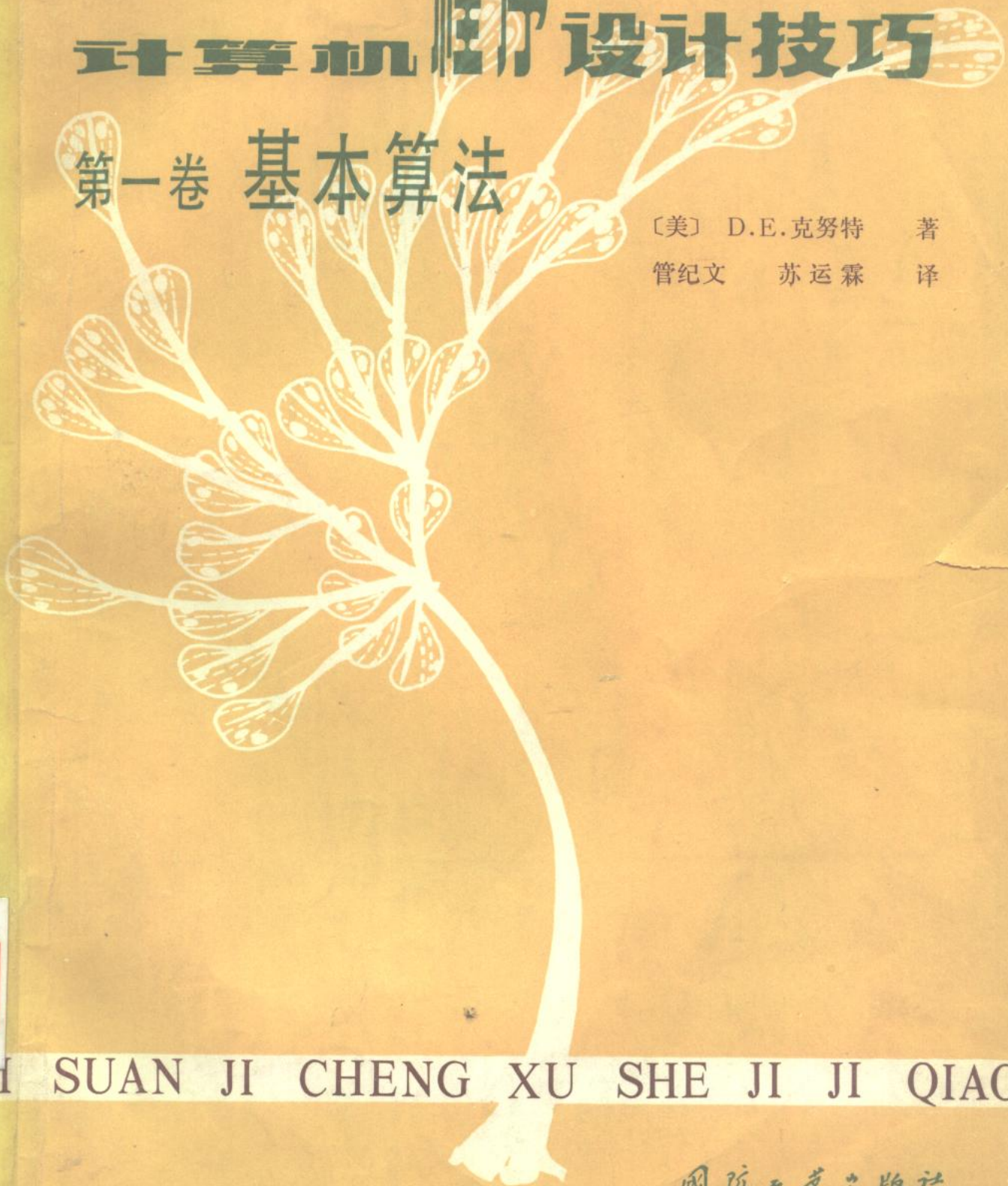
计算机程序设计技巧

第一卷 基本算法

〔美〕 D.E.克努特 著
管纪文 苏运霖 译



JI SUAN JI CHENG XU SHE JI JI QIAO

国防工业出版社

计算机程序设计技巧

(第一卷　基本算法)

〔美〕 D.E.克努特　著

管纪文　苏运霖　译

国防工业出版社

内容简介

《计算机程序设计技巧》是计算机科学方面的一部重要著作，受到了国际上有关方面的普遍重视。

本书共分七卷。主题是"算法分析"，即特定的计算机算法性质的理论。全书详尽地介绍了若干"非数值分析"领域中大量采用的各种优秀方法和技巧，并从数学上给予了严格的论证和比较分析。书中附有大量习题和答案，标明了难易程度及数学概念的使用。这部较完整的书，可供从事计算机科学、计算数学、计算技术诸方面的工作人员参考、研究和借鉴，也可作为高等院校有关专业的一部基本的教学参考书。

本卷是全书的第一卷——《基本算法》，分"基本概念"和"信息结构"两章。它是其他六卷的基础，是整部书的交叉点；介绍了全书其他部分所用的基本概念和数据结构，包含了各卷中需要的全部材料；可作为数据结构、离散数学和机器语言程序设计方面的教科书。

THE ART OF COMPUTER PROGRAMMING
Volume 1/Fundamental Algorithms
D.E.Knuth
1973 BY ADDISON-WESLEY PUBLISHING COMPANY, INC.

*

计算机程序设计技巧
（第一卷　基本算法）
〔美〕D.E.克努特　著
管纪文　苏运霖　译

*

国防工业出版社　出版
新华书店北京发行所发行　　各地新华书店经售
国防工业出版社印刷厂印装

*

787×1092 1/16　印张36 3/4　849千字
1980年3月第一版　1980年3月第一次印刷　印数：00,001—40,200册
统一书号：15034·1873　　定价：3.75元

译 者 序

唐·欧·克努特是美国当代年青而杰出的计算机科学家。一九六〇年在俄亥俄州的克利夫兰的凯西理工学院取得数学学士和硕士学位，并于一九六三年在帕萨迪纳的加里福尼亚理工学院获得博士学位。从一九六三年到一九六八年，他在加里福尼亚理工学院任教，一九六八年又被加州斯坦福大学聘任为教授，当时他仅三十岁。

在计算机科学领域中，他的主攻方向是程序设计语言、算法分析及计算机科学史。在数学领域中，他主要研究组合分析和数论。并已发表论文一百八十余篇：其中专题研究九十余篇，其余是评述性或综述性论文。他还写过介绍算法分析技术的一本书，去年又写了一本数学短篇小品。这惊人的著作数量，既表明了他的卓越才能，也表明了他的勤奋刻苦。正是由于他的这些建树，引起了国际计算机科学界的注意，并赢得了很大声望。现在他是国际上十二种计算机科学和数学杂志的编委。他是美国计算机器协会（ACM）、美国数学会、美国电气及电子工程师学会（IEEE）、计算机学会、美洲数学学会、工程与应用数学学会、斐波那契学会等学术组织的成员，美国艺术和科学院院士，国家科学院院士。由于他的杰出成就，他于一九七一年获得美国计算机器协会颁发的格雷斯·霍珀奖，于一九七四年获得图林奖。

《计算机程序设计技巧》一书，是他的匠心之作。他正是由于这部书，而获得一九七四年的图林奖的。这部书共有七卷，现在还未全部完成。在计算机科学中，写出这样一部巨著，就其篇幅而言，堪称执牛耳者矣。它以妙趣横生的叙述和流畅通达的笔调，写出了范围广泛、精湛深刻的思想。《计算机程序设计技巧》讨论了构成计算机科学基础的两大主题：算法和语言。特别是就目前所见到的三卷而言，是以对各种算法的研究作为贯穿于全部这三卷的主线。无怪乎这部书出版之后，赢得了计算机科学界的一片称赞，且竞相开列此书作参考文献。它确实是计算机科学的一部经典著作。正是由于我们对于这部书的价值有上述认识，才决心下大气力把它译出，奉献给我国计算机科学界。期望它能对发展我国的计算机科学，赶超世界先进水平发挥作用。

现在，这部译稿终于开始和大家见面了。我们首先要衷心感谢以华主席为首的党中央。如果不是以华主席为首的党中央领导全国人民一举粉碎祸国殃民的“四人帮”，摧毁他们的文化专制主义，迎来我国科学的春天，我们这部译稿是决不可能问世的。关心、支持和鼓励这部书翻译工作的有十五所和科学院数学所及计算所的一些同志。在此，我们仅向这些单位和同志们致以深切的谢意。

我们虽然费了很大气力，力求忠实、准确地翻译原书，并一一勘正了原书中印刷或内容错误之处，但由于水平有限，译稿中难免还有缺点和错误，望读者在阅读中随时提出批评和指正。

一九七八年八月

前　言

为一台数字计算机编制程序的过程是饶有趣味的。因为它不但具有经济和科学价值，而且也是犹如赋诗或作曲那样的美学实践。这一部书的目的，就是要对读者进行作为熟练的程序员所必需的各种技巧的训练。本书是一部七卷巨著的头一卷。

本书并不打算作为计算机程序设计的入门介绍；而是假定读者已经有了一定的预备知识。所需要的预备知识实际上是非常简单的。但对于一个初学者说来，要想真正地理解数字计算机的概念，需要时间和实践。我们要求读者：

a）具有存储程序式数字计算机怎样进行工作的某些知识；但不必是电路方面的，而只须知道指令是如何保留在机器的存储器中，并依次被执行的。当然，对机器语言有预先的了解，也是有帮助的。

b）能把问题的解法表达成计算机能够“理解”的一种明显的形式（这些机器没有普通的感觉；它还不会“思考”。但它能准确地不折不扣地去做让它做的事情。当一个人初次试图使用一台计算机时，这是最难以掌握的概念）。

c）具有最基本的计算机技术的某些知识。例如循环（重复地执行一组指令），子程序的用法以及变址寄存器的用法等。

d）具有普通计算机专业术语方面的少量知识。例如，“存储器”、“寄存器”、“位”、“浮点”、“溢出”等。对于正文中未加定义的大多数词汇，在每卷末尾的索引中都给出了简明的定义。

这四方面的预备知识，也许可以概括成一个要求，即是，读者至少应该为一台计算机编写过和调试过，比如说四个程序。

我力图使这部书满足多方面的要求。首先，这是一部概括了若干重要领域已经获得的成果的参考书。同时，它也可以作为计算机和信息科学方面自学的或者大学课程的教科书。为了达到这两个目标，我在正文中附加了大量的习题，并对大多数习题提供了答案。我还尽力充实本书的内容，而避免作含糊空泛的评论。

我这部书的对象，不是对于计算机仅仅出于一时的兴趣的那些人们。但这决不是说，这部书仅仅是为计算机的专业人员写的。事实上，本书的主要目标之一，就是要使工作在其它领域的许多人能有更多的机会接触到这些程序设计技术，这样，当使用计算机时，就不必要去查找分散在各种技术杂志上的所有必要的情报资料。

这部书的主题，可以叫做“非数值分析”。尽管计算机传统地同数值问题的求解——诸如方程式根的计算，数值内插和数值积分等等有关。但在这里，除顺便说及外，我们不讨论这些课题。数值的计算机程序设计是一个非常有趣和迅速发展的领域，而且已经出版许多书了。然而，近些年来，对于利用计算机来解决本质上是非数值的问题——例如，分类、翻译语言、解高等代数和组合分析方面的数学问题、定理证明“软件”（为编制其它程序提供方便的程序）的研制，以及对日常生活中的各种过程的模拟等——也已经进行了大量有

趣的工作。在这样一些问题当中，仅仅是由于巧合才出现数值。这里用的是计算机进行判断的能力，而不是它进行算术运算的能力。在非数值问题中，我们使用了一些加法和减法，很少需要乘法和除法。值得注意的是，即使主要是从事数值计算机程序设计的人，他也能从非数值技术的研究中获益，因为这些技术在数值程序中同样存在。

非数值分析方面新近的研究成果，分散于各种技术杂志中。而且在它们成文时，多少还处于杂乱无章的状态。这里所采取的途径，是研究最为基本的那些技术，它们可被广泛用于许多类型的程序设计场合。我力图把这些整理成或多或少的“理论”，并使读者能了解到这门知识领域现今的最新发展情况，并给出这些基本的技术在设计软件程序方面的应用。

当然，对于这个研究领域说来，“非数值分析”这一名称纯粹是反面的名称。如果有一个用来表征这个课题的正面的描述性术语，就要好得多。对于我所考虑的内容说来，“信息加工”是一个太广泛的称呼，而“程序设计技术”又嫌太窄。因此，我希望以“算法分析”来作为包含于这部书中的课题材料的适当的名称。如同这部书中更详细地说明的那样，这一名称隐含有“关于具体的计算机算法之性质的理论”的意思。

一般说来，要让某个成果始终保持其经济价值，是很困难的。因此，自然而然的是，预期这里所叙述的许多技术，最终将被更好的一些技术所代替。当然，对于我来说，预见“未来两年的发展水平”是不可能的，而且上边提到的最新发展情况肯定将发生变化。在这方面，我有一种交集的感情。因为我当然希望这部书将促进对计算机程序设计艺术的进一步的研究，但同时也希望这种促进作用不致过大，以致很快就使本书本身变得过时陈腐了。

实际上，这里所介绍的大多数算法，已经为许许多多人使用了五年之久以至更长时间。因此，就某种意义而言，这些方法已经是成熟了：已经被人们很好地理解了，大体上已经具备了最好的形式。所以，把它们集中到一部教科书当中，让学生们进行学习，已不再是为时过早的事情了。

整个这七卷书，以“计算机程序设计技巧”为题，其总目录如下：

第六卷　语言理论

第11章　数理语言学

第七卷　编译程序

第12章　程序设计语言的翻译

我从 1962 年开始，就动笔以这些章次来写一本书。但是我很快就发现，更为重要的是要以一定的深度来讨论这些课题，而不是轻率地一带而过。我写出的文稿的篇幅使我意识到，每章本身就包含了足够大学一学期课程的内容，所以，把这部书分作几卷出版，而不把它弄成一两册大部头的巨著是明智的（在一卷书中仅有一章或两章，这似乎是少见的。但我已经决定，为了便于交错参考，仍保留这种编排。我打算不久就出版第一卷到第五卷的一个缩写本，专门用作大学计算机课程的更通用的教科书。它的内容将是这部书的“子集”，但省略了这部书中那些更专门的材料，在这版本中，我打算仍使用同样的章次）。

本卷可以看作整部书的“交叉点”。意思是说，它包含了所有其它各卷中使用的全部材料。第 2 卷到第 7 卷，可以彼此独立地加以阅读；对于第 5 卷和第 7 卷之间联系较密切的某些部分或许有所例外。第 1 卷不仅是用来与第 2 卷到第 7 卷相联系的一部参考书，也可以用作数据结构方面的大学课程或自学的教科书（以第 2 章的材料为重点），或者作为离散数学方面的教科书（以 1.1，1.2，1.3.3 和 2.3.4 等节的材料为重点），或者作为机器语言程序设计方面的教科书（以 1.3 和 1.4 节的材料为重点）。

在编写这十二章时，我所采取的观点与当代许多关于计算机程序设计的书的观点是不同的。我并不试图向学生讲授如何使用别人编制的子程序，而宁可致力于向他们讲述：怎样才能编制出更好的子程序来？

针对这部书的教学内容，还有几句话要说一说。我已经把材料组织成这样，使得具有高中代数知识的人就可以阅读它，只要他一带而过地读读有更多数学内容的部分就行。但是对数学有所爱好的读者将学习到有关“离散数学”的许多有趣的数学技巧。为兼顾这两方面的需要，我对每个习题都评定了“分数”，并把那些主要是数学的习题特地标了出来。同时，在大多数章节中，把主要的数学结果编排在它们的证明之前来叙述。证明或者留作习题（答案可在一个独立的部分中找到），或者在一节的末尾给出。

对于那些主要是对程序设计而不是对有关的数学内容感兴趣的读者，只要感到数学确实已经成为难题了，可停止阅读有关数学内容方面的大多数的节次。另一方面，一个面向数学的读者将发现，这里收集了大量的有关数字方面的有趣的材料。有关计算机程序设计的许多已发表的数学著作，毛病很多，因而本书的目的之一，是向读者讲授与这一课题相适应的数学方法。由于我自认为是一个数学家，尽我所能地来保持数学的完整性，自然是我的义务。

对于这部书的大多数部分说来，有初等微积分的知识就够用了。因为所需的大多数其它的理论，书中已加以提供。不过，也有个别地方，出于必要，引用了复变函数论、概率论、数论等较深的定理。

尽管一般都认为计算机是属于“应用数学”的领域，但也有些象我这样的“纯粹数学家”，已经发现在计算机和抽象数学之间，有着许多值得注意的联系。从这一立场出发，可把本书的一些部分看作是一个纯粹数学家关于计算机的“观点”。

对于一个外行人说来，电子计算机已经成了数学在当今世界上之重要性的象征，而且有少量的数学家，现在正专心地去熟悉机器。造成这一意外的状况的一个原因是，计算机似乎已经使某些事情变得太“舒适”了：人们不再动用笔和纸去做很多事情，也不去发现许多辅助工作的数学简化。某些数学家有时不满于计算机的侵扰，这并非因为他们害怕自动化会抢去他们的饭碗，而是因为他们担心，也许将不大必要来搞某些发明了。另一方面，在数值分析、数论和统计学等等方面，计算机和数学之间，有着明显的联系。

我希望说明，计算机和数学之间的联系，比之于上述传统关系所意味的，要远为深入和密切。用一组基本的指令来编制一个计算机程序，非常类似于从一组公理来构造一个数学证明。其次，纯数学的问题，历史上往往是从另一领域中的实际问题提出来的，而由于计算机的出现已经产生了许多这样的问题。本书中所研究的本质上属于这种类型的问题有：（a）研究具体算法的随机性质：确定这些算法如何按预期的目标得以实现；（b）构造最优的算法，例如对于分类或多项式的求值；以及（c）语言理论。除了数学工具在程序设计问题上的有趣应用之外，计算机对于探索数学的猜想方面，例如，在组合分析和代数中，也有有趣的应用；而且，在许多情况下，计算机和经典数学之间，有着相当大的相互作用。数学的机械化尝试也是很重要的。因为它使我们更深入地理解我们认为已经知道的概念（深入到已经针对一台计算机来说明这些概念）。我相信，本书中所列举的计算机和纯粹数学之间的联系的重要性将变得越来越大。

在编写这部书时，我所要作的最困难的决定，是表达各种技术的方式。流程图和算法的非形式的逐步描述方式的优点是众所周知的；关于这方面的讨论，见ACM Communications，Vol 6（1963年9月），第555～563页上的论文《计算机画的流程图》（Computer-Drawn Flowchavts）。但为了确定任何计算机算法，一种形式的精确的语言也是必要的。而且我还需要就下面这样一个问题作出决择，即：是使用诸如ALGOL或FORTRAN这样一种代数语言呢？还是使用一种面向机器的语言？我决定使用一种面向机器的语言，对此也许当今的许多计算机专家不同意，但我确信这是一个正确的选择。其理由如下：

a）就这里所讨论的非数值问题同数值问题比较而言，代数语言更适合于后者；尽管程序设计语言正在逐步地改进，但是今天的语言仍然不能适应诸如共行程序、输入输出缓冲、生成随机数、多精度算术等课题，以及到处出现的有关组装数据、组合检索和递归等许多问题。

b）程序员编写程序所使用的语言，对他有着巨大的影响；这里压倒的趋势是，宁愿在所用语言当中有最简单的结构，而不求对机器说来最好的结构。通过以面向机器的语言来写程序，就使程序员趋向于使用更为有效得多的方法。这更接近于实际。

c）除少数例外情况外，我们需要的程序，大都是比较短的。所以如果有了一台合适的计算机，了解这些程序并不困难。

d）一个比较正经地对计算机感兴趣的人，应当很好地学会机器语言。因为它是一台计算机的基本部分。

e）作为第1，9，10和12章中所叙述的软件程序的输出，无论如何总是需要某种机器语言的。

诚然，以高级程序设计语言来写程序，被公认为容易的，而且检查程序也容易得多。因

此有一大类问题，使用代数语言（来编程序）要更好些，虽则对应于这一代数语言程序的实际机器语言程序，通常距它可能的最好形式，还差得很远。然而，本书中许多我们所感兴趣的问题，对于程序员的技巧说来至关重要。例如，作为软件例行程序的程序，在一个计算机装置中，每天都要使用那么多次，因而作出额外的努力来编写这些程序，是值得的。因为这些程序仅仅需要编写一次就够了。

作出了使用一个面向机器语言的决定之后，应当使用哪一个语言呢？我可以选择一个特殊机器X的语言。但由此，那些不具有机器X的人们，将把本书想象成仅仅是为与机器X有关的人们写的。机器X总还有同本书的内容全然无关的许多特性，但却仍须在本书中加以交代，而且，两年之内，机器X的厂家将生产出机器$X+1$或$10X$来，因而机器X将不为任何人所感兴趣了（当然，如果我虚构出一台假想的计算机，它可能现在就已经不为任何人感兴趣了）。

为了摆脱这个困境，我试图设计一台叫做MIX的“理想的”计算机。它具有非常简单的操作规则（比如说，仅仅需要学习一个小时就可以掌握了），而且它非常类似于现存的各台计算机。因此，MIX程序就可以为大多数实际的机器所接受，或者可以在大多数机器上加以模拟。

一个学生没有任何理由要害怕学习多个计算机的特征；事实上，应该预料到，一生当中要遇到许多不同的机器语言；而且一旦掌握了一种机器语言，其它的就易于融汇贯通了。所以对于一个凭空想出的机器，所剩下的唯一缺点是，它不能执行对它写出的任何程序（为了解决这个问题，建议教师要有一个可以利用来运行学生习题的MIX模拟程序。这样一个模拟程序有一个优点，就是可以容易地插入自动评分程序；但它有明显的缺点，就是要花上几天的功夫来准备这样一个程序。为了简化这个任务，第1章包含了以MIX机器本身的语言写成的一个MIX模拟程序，对于类似的机器，可以很容易地修改这个程序）。

幸而计算机科学这个领域还很年轻，使我得以进行相当彻底的研究。我已经尽我所能地精读了有关本书所讨论的课题的直到目前为止所有已发表的文献，而且事实上我还读了大量的未发表的文献。当然我不能吹嘘已经全然无漏。为了精确地编写出在每章中所讨论的重要思想的历史，我曾经写过大量的信。然而，在任何这样大量的工作当中，即使已经进行过大量的准确性校验，仍然难免会产生一些疏忽和转手的错误。特别是，我要向在本书关于历史的那些叙述中，可能被无意地疏忽的任何人表示歉意。谬误之处，恳请读者提出，以便将来再版时尽快地订正。

我试图就每个课题提供一份现有好文章的带注释的参考文献，而且试图选用精确的，同当前的用法相一致的术语。在引证文献时，期刊的名称都采用标准的缩写给出，而最普遍引证的一些杂志，则采用下列的缩写：

CACM＝Communications of the Association for Computing Machinery（美国计算机器协会通讯）

JACM＝Journal of the Association for Computing Machinery（美国计算机器协会杂志）

Comp. J.＝The Computer Journal(British Computer Society)(计算机杂志——英国计算机协会)

Math. Comp. =Mathematic of Computation（计算的数学）

AMM=American Mathematical Monthly（美国数学月刊）

例如，“CACM 6 (1963)，555～563”表示在本前言中前面那段所提到的那篇文章。

本书还把许多技术内容收集在习题中。当包含于一道深奥的习题中的思想不属于我自己的时候，我尽力把这一功劳归于这一思想的创始者。参考文献通常在有关的段的附文中给出，或者在习题的答案中给出。但在许多情况下，习题是以未发表的材料为基础的。对于这些材料，就不能给出进一步的参考文献了。

唐·欧·克努特（D. E. Knuth）1967 年 10 月

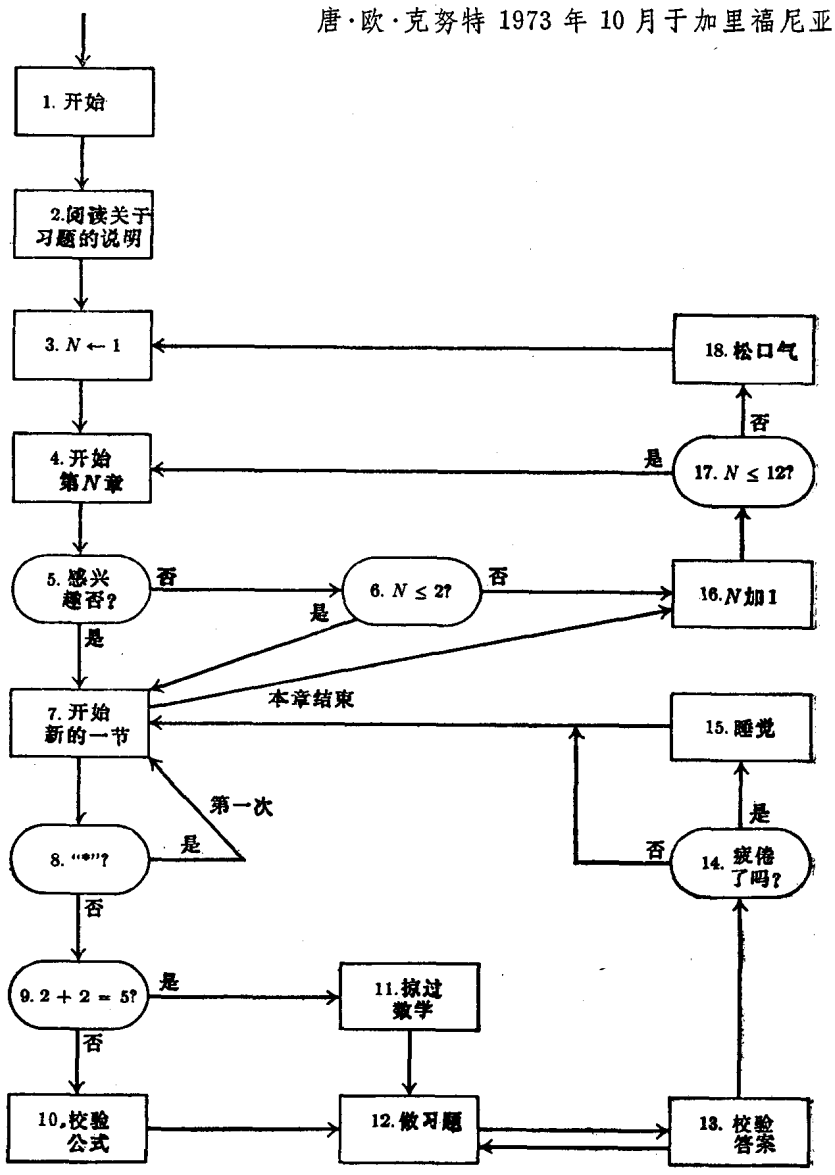
于加里福尼亚帕萨迪纳

第二版前言

本书第一卷的头一版受到了出乎意料之外的欣赏和欢迎，使我极为高兴。在这第二版里，我重新审阅了全部手稿，作了大量的改进，并暗暗地加上了某些新材料，但仍保留了原来的编页。不经心的读者很难看出第一版和第二版有任何区别。但是，事实上有百分之九十以上的篇幅都已以某种方式作了改进。

第 2.3.1～2.3.3 节作了实质性的修改。在那里，我果断地对树形的遍历次序的术语作了改动。幸而还没有任何人采用在第一版中对于这些次序引入的拙劣的名称。其余的许多改动可以在历史和参考文献部分找到，我已经把这一部分内容一直延伸到当前为止。

唐·欧·克努特 1973 年 10 月于加里福尼亚斯坦福



阅读本书的流程图

阅读这部书的步骤

1. 开始，先阅读这一步骤，除非你已经读过了。继续忠实地遵循这些步骤(这一步骤的叙述以及与之相应的框图将贯穿全书予以使用)。

2. 阅读关于习题的说明。

3. 置N等于 1。

4. 开始阅读第N章。不要读本章开头部分的语录。

5. 你对这章的课题感兴趣吗？若感兴趣，进行第 7 步；若不感兴趣，进行第 6 步。

6. $N \leqslant 2$？若否，进行第 16 步；若是，随便地浏览这一章（第 1 章和第 2 章包含有重要的介绍性材料，并且也是基本的程序设计技术的综述。你应当至少浏览关于记号和关于 MIX 的部分)。

7. 开始阅读本章的下一节。如果你已读完本章，就进行第 16 步。

8. 这一节带有“*”吗？若是，则在头一次阅读时，你可略去这一节(这里包含有较为专门的课题，这些课题是有趣的，但并非必要的)；进行第 7 步。

9. 你对数学有爱好吗？如果你对数学完全不懂，进行第 11 步；否则进行第 10 步。

10. 校验这一节中所作的数学推导（并把错误之处告知作者)。进行第 12 步。

11. 如果这一节充满了数学计算，你最好略去这些推导不读。然而，你应当熟悉这一节的基本结果；这些基本结果通常在开始处附近叙述，或者在困难部分的末尾用斜体字叙述。

12. 按照（你在第 2 步读到的）关于习题的说明中给出的提示作这节中推荐的习题。

13. 在你已经自认为满意地把题目做完后，用本书后面相应的答案部分中印出的答案来校验你的答案（如果对于这个问题有答案的话)。也读一读你没有时间做的习题的答案。注意：在大多数情况下，在做第 $n+1$ 道习题之前，先读第 n 道习题的答案是合理的，所以第 12 步和第 13 步通常是同时做的。

14. 你是否疲倦了？若否，返回第 7 步。

15. 去睡觉。然后，醒来，并返回第 7 步。

16. N增 1。如果$N=3, 5, 7, 9, 11$ 或 12，开始这部书的下一卷。

17. 若N小于或等于 12，返回第 4 步。

18. 现在试试来说服你的朋友购买第 1 卷，并开始阅读它。而且，返回第 3 步。

关于习题的说明

这部书的习题既适合于自学，又可用作课堂作业。任何人只想通过单纯的阅读，而不把所学到的知识应用于特定问题并强迫自己对已经阅读过的东西反复进行思考，就想学习一门学问，如果说不是不可能的，那也是困难的。其次，凡人们自己亲身探索学到体会到的东西学得最好。因此习题就成为这一工作的主要部分。我力求使这些习题尽可能地有益于上述目的，并尽量选择解决有趣味的问题。

在许多书里，容易的习题随机地混杂在极其困难的习题当中。这有时是不合适的，因为读者在认真解决一个问题之前，首先要想到需要用多少时间来解这个问题（否则他也可能跳过所有的问题）。这种情况的一个典型的例子是理查德·贝尔曼（Richard Bellman）所著的《动态规划》（Dynamic Programming）一书。这是一本重要的开创性的书，其中在若干章的末尾，在“习题和研究题”的标题下，把一组问题收集在一起，而且把极为浅显的问题混杂在深奥的还未解决的问题当中。据说曾经有人问起贝尔曼博士怎样把习题同研究题区分开，他回答说：“如果你能解决，它就是一个习题；否则它就是一个研究题。”

把研究题和非常容易的习题同时包括在这一类型的书里究竟是好是坏可进行充分的辩论；为使读者免于可能陷入确定是习题还是研究题的困境，我特地提供了分数以指出困难的程度。这些分数的一般意义如下：

分数	解　　释
00	一个极其容易的习题，如果你理解了书上的材料，就能立即回答它，而且几乎总是可以只用心算，不必动手就能解出。
10	一个简单的问题，它使你思考一下刚刚阅读过的材料，但这并不意味着是困难的。至多在一分钟内就能够做出它。为了得到解答，可能要用到笔和纸。
20	一个普通的题目，它能测验你对于课文内容基本的理解程度，但大约化上十五到二十分钟的时间就可能得到完全的答案。
30	一个中等难度或复杂的问题，可能需要两个小时以上的时间才能找出满意的解答。
40	确实是一个困难的或冗长的问题，在学校里，它也许适合于作为学期设计。预期一个学生要在相当长的一段时间里才能解决这个问题，解这类问题虽费力，但是很有意义。
50	一个研究题，它（就作者在写书时所知）是尚未满意地解决的问题。如果读者发现了这个问题的答案，请把它写出来发表；而且，本书的作者将乐于尽快地听到关于这个解答的消息（假定它是正确的）！

通过在这些分数的“基准”标度中进行插值，其它分数的意义就自明了。例如，分数17将表示这是一个比普通的题目要简单的题目。带有分数50的问题，如果随后被某个读者解决了，在本书再版时可能就带有分数45。

作者试图认真地评定准确的分数。但提出问题的人要确切地知道这个问题对于另外一个人的难度如何，确实是有困难的。而且某个人对于某种类型的问题较之其他人可能更适应些。希望分数能成为表示困难程度的一个好的估计，但它只应当作一般的准绳，而不应当作绝对的指示。

本书是为经过不同程度的数学训练和具有不同程度的数学素养的读者写的；而且有些习题是仅仅想供给有更多的数学基础的读者利用的。因此，如果习题所涉及的数学概念或出题的动机大大超过了仅关心程序设计算法者的需要，则在分数前写上M。如果习题的解答需要有微积分或者在本书中未提供的其它高等数学的知识，习题就标以字母“HM”。标示“HM”的习题并不一定困难。

有些习题以一个黑箭头“▶”打头，这表示这个问题是特别有益的和特别推荐的。当然，并不期望读者或学生来做所有的习题，所以才把那些也许是最有价值的标示出来。这并不意味着贬低其它习题的价值，每一读者至少应当试图解决所有分数为10以下的那些习题。箭头还可以帮助你决定哪些具有较高分数的问题应该先做。

大多数习题的答案已在答案部分中给出。请明智地使用它们。在你未独自地作出真正的努力来解决问题之前，不要先翻阅答案，除非你没有时间来做那个问题。在你得到自己的解答或对问题进行了郑重的尝试后，你可能感到答案是有益的和有帮助的。给出的答案通常是十分简短的，而且假定你自己事先已经进行了认真的尝试，因而已将细节略去。

有时解答给出的信息比想要的少；但通常都要多给一些。你的答案很可能比我们这里给出的要更好些，或者你可能在发表的解答中发现某些错误；在这样的情况下，作者希望尽快地知道有关答案的详细情节。在本书再版时，作者将把改进了的答案，连同解答者的名字一起在适当的地方给出来。

当做完一个习题时，一般地如未明文禁止，你可以利用前面习题的答案。我在指定分数时也已经考虑到这一点了。于是，也有可能，习题$n+1$的分数比习题n低，尽管习题$n+1$包含习题n的结果作为特殊情况。

代码一览表

▶	推荐的
M	面向数学的
HM	需要“高等数学”
00	即刻的
10	简单的（一分钟）
20	普通的（一刻钟）
30	中等难度的
40	学期设计
50	研究题

习题

▶1.〔00〕 分数“$M20$”意味着什么？

2.〔10〕 一本教科书中的习题对读者有什么价值？

3.〔14〕 证明$13^3=2197$。推广你的答案。〔这是一类繁重计算的问题的例子，作者将力图避免之〕。

4.〔$M50$〕 证明当n是一个整数，$n>2$时，方程$x^n+y^n=z^n$无正整数x，y，z的解。

目　录

第1章　基本概念

第2章　信息结构

第1章 基本概念

1.1 算 法

对于所有的计算机程序设计说来，算法的概念总是基本的，所以我们应当首先细心地分析这一概念。

“算法”(Algorithm)一词本身就是十分有趣的。初看起来，这词好像是某人打算要写的“对数”(Logarithm)一词但却把头四个字母写得前后颠倒了。这个词直到1957年之前在《韦氏新世界词典》(Webster's New World Dictionary)中还未出现，我们只能找到带有它的古代含意的较老形式的“算术”(Algorism)，指的是用阿拉伯数字进行算术运算的过程。在中世纪时，珠算家用算盘进行计算，而算术家用算术进行计算。中世纪之后，对这个词的起源已经拿不准了，早期的语言学家试图推断它的来历，认为它是从把algiros〔费力的〕+arithmos〔数字〕组合起来派生而成的，但另一些人则不同意这种说法，认为这个词是从“喀斯迪耳国王Algor”派生而来的。最后，数学史学者发现了算术“algorism”一词的真实起源：它来源于有名的《波斯教科书》(Persian textbook)的作者的名字阿布·贾法·穆哈默德·伊本·穆萨·阿科瓦里茨米(Abu Ja'far Mohammed ibn Mûsâ al-Khowârizmî)(约公元825年)——从字面上看，是“Ja, far的父亲，Mohammed, Moses的儿子，Khowârizm的土人”。Khowârizm是今天苏联基发(ХИВА)市的小城镇。Al-Khowârizm写了著名的书《复原和化简的规则》(Kitab al jabr w'al-muqabala)；另一个词，“algebra”(代数)，是从他的书的标题引出来的，尽管这本书实际上根本不是讲代数的。

逐渐地，“algorism”的形式和意义就变得面目全非了。如牛津英语词典所说明的，这个词是由于同arithmetic(算术)相混淆而形成的错误的造作。由algorism又变成algorithm。只要了解人们已经忘记了这个词原来派生的实况，就不难理解这种变化。一本早期的德文数学词典《大全数学辞典》(Vollständiges Mathematisches Lexicon莱比锡，1747)，给出了Algorithmus(算法)一词的如下的定义：“在这个名称之下，组合了四种类型的算术计算的概念，即是，加法，减法，乘法和除法”。拉丁短语algorithmus infinitesimalis(无限小算法)，在当时就用来表示“莱布尼兹(Leibnitz)所发明的以无限小量进行计算的方法”。

1950年左右，algorithm一词经常地同欧几里得算法“Euclid's algorithm”联系在一起。这算法就是在欧几里得的《几何原本》(Euclid's Elements，第VII卷，命题i和ii)中所阐述的求两个数的最大公因子的过程。在这里给出欧几里得算法来，也将是有益的：

算法E(欧几里得算法)。给定两个正整数m和n，求它们的最大公因子，即能够同时整除m和n的最大的正整数。

E1. 〔求余数〕以n除m并令r为所得余数(我们将有$0 \leqslant r < n$)。

E2. 〔余数为0？〕若$r = 0$，算法结束；n即为答案。

E 3.〔互换〕 置$m \leftarrow n$，$n \leftarrow r$，并返回步骤 E1. ▮

当然，欧几里得并非就是以这样的方式来给出他的算法的。上面的格式充分演示了在给出本书中所有的算法时贯穿全书的风格。

我们对所讨论的每一个算法，都给出了一个标识的字母（例如上边算法中的E），并且用这个字母后面接上数字（例如E 1，E 2 等等）来标识这个算法的步骤。书中各章分为编了号的节，在一节之内的算法仅用字母来标记；但当在其它节中引用这些算法时，则附以相应的节号。例如，我们现在是在第 1.1 节；在这一节中，欧几里得算法叫做算法 E，而在后边的节引用它时，就记作算法 1.1 E。

一个算法的每一步（例如上边的步骤 E 1）以一个方括号中的一个短句开始，它尽可能简短地概括这一步的主要内容。这一短句通常也出现在一个与这个算法相对应的框图（例如图 1）中。借助于框图，读者将有可能更直观地看出这个算法的流程。

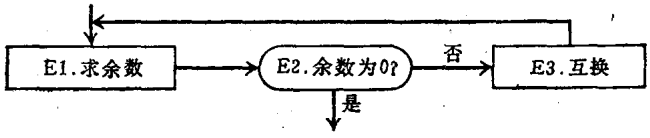


图 1　算法 E 的框图

在概括性的短句之后，是有待执行的某个动作或有待作出的某个判断的文字符号的叙述。偶而也有带圆括号的注释（例如，步骤 E 1 的第二句），它是关于这一步的说明性信息，通常指出变量的某些特征或这一步的当前目标，等等；带圆括号的注释并不影响属于这一算法的动作，而只是为了便于帮助读者理解。

步骤 E 3 中的箭头←是最重要的替代运算（有时叫做赋值或代换）。“$m \leftarrow n$”意思是变量 m 的值代之以变量 n 的当前值。当算法 E 开始时，m 和 n 的值是原来给出的数；但当它结束时，一般说来，这些变量将有不同的值。箭号用来把替代运算同相等关系加以区别：我们将不说“置 $m = n$”，但也许我们将要问“是否 $m = n$？”“=”号标记一个可被测试的条件，“←”号标记一个可被执行的动作。n 增 1 的运算通过“$n \leftarrow n + 1$”来标记（读作以 $n + 1$ 代替 n）。一般说来，“变量←式子”意味着用出现于式子内的所有变量的当前值来计算式子，并将结果代替箭头左边的变量的原先值。不熟悉计算机工作的人们有时有这样的倾向，即用“$n \rightarrow n + 1$”来标记 n 增 1 的运算，说是“n 变成 $n + 1$”，这只能导致混乱，因为它同标准的约定正相冲突，因此我们应加以避免。

注意，步骤 E 3 中的动作的次序是重要的。“置 $m \leftarrow n$，$n \leftarrow r$”完全不同于“置 $n \leftarrow r$，$m \leftarrow n$”。因为后者意味着，在用 n 的值来置 m 之前，n 的原先值已经失去了。因此，后一次序等价于“置 $n \leftarrow r$，$m \leftarrow r$”。当若干个变量全都置成等于相同的量时，我们使用多重的箭头；于是，“$n \leftarrow r$，$m \leftarrow r$”可以写成“$n \leftarrow m \leftarrow r$”。两个变量的值相互交换，我们可以写成“交换 $m \longleftrightarrow n$”；这一动作也可通过使用一个新变量 t，并写“置 $t \leftarrow m$，$m \leftarrow n$，$n \leftarrow t$”来确定。