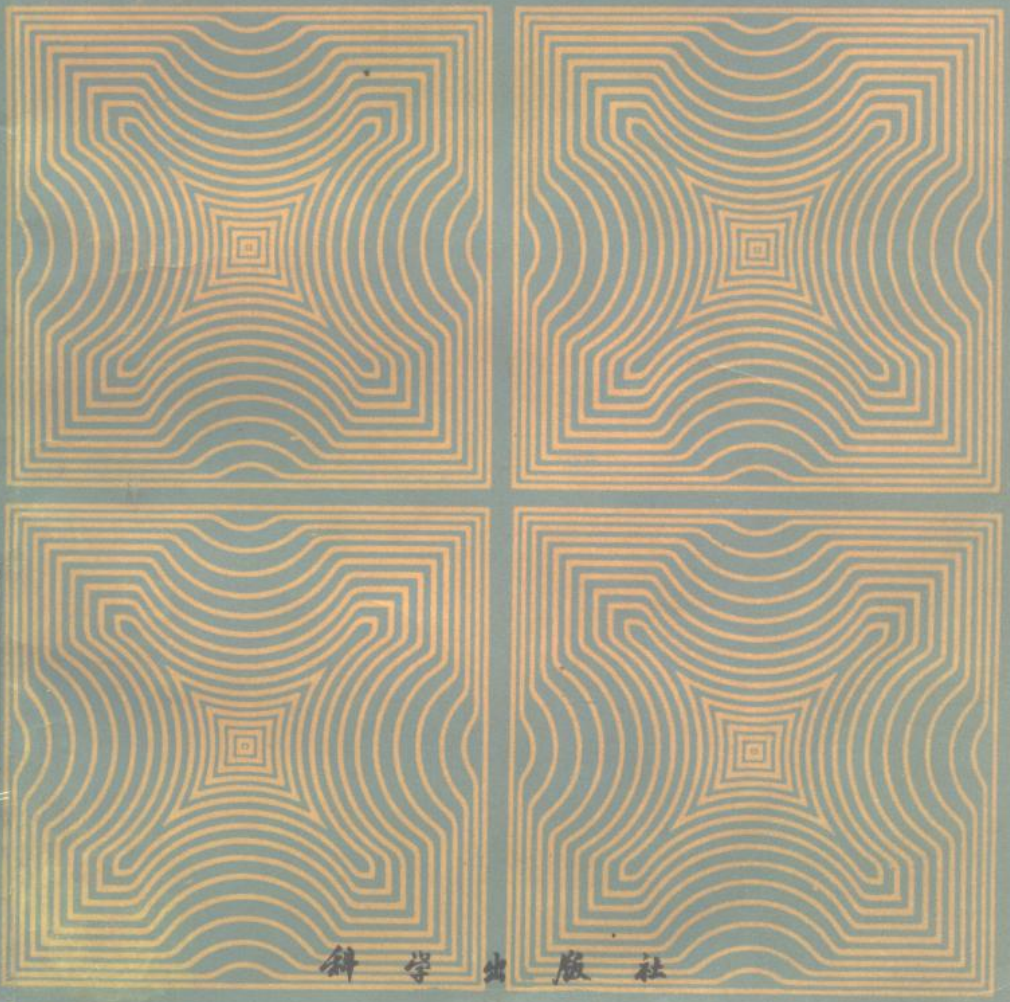
〔美〕W. A. 伍尔夫 M. 肖
P. N. 希尔芬格 L. 弗朗 著

计算机科学的基本结构



科学出版社

计算机科学的基本结构

〔美〕W. A. 伍尔夫　M. 肖
P. N. 希尔芬格　L. 弗朗　著

郑茂松　陈世鸿　梁　英　译

卢道全　校

科学出版社

1987

内 容 简 介

本书按照程序设计技术与基本数学概念相结合的原则，全面而又系统地论述了控制结构、数据结构及其相互作用，是一本计算机科学与软件工程的基础教材.

本书分四部分，共二十三章. 第一部分为第一至第六章，讨论基本控制结构；第二部分为第七至第十二章，讨论基本数据结构；第三部分为第十三至第十八章，讨论控制与数据的相互作用. 这三部分内容都是按数学模型、程序设计语言概念、表示、正确性、效率这样的顺序依次论述的. 第四部分为第十九至第二十三章，这一部分通过几个实例说明前三部分在实际程序设计中的应用.

本书可作为计算机软件专业大学生和研究生的教材，也可供程序设计人员参考.

W. A. Wulf, M. Shaw, P. N. Hilfinger, L. Flon
FUNDAMENTAL STRUCTURES OF COMPUTER SCIENCE
Addison-Wesley, 1981

计算机科学的基本结构

〔美〕W. A. 伍尔夫　M. 肖　P. N. 希尔芬格　L. 弗朗　著
郑茂松　陈世鸿　梁　英　译
卢道全　校
责任编辑　那莉莉

科学出版社出版
北京朝阳门内大街 137 号
中国科学院印刷厂印刷
新华书店北京发行所发行　各地新华书店经售
*
1987 年 2 月第 一 版　开本：850×1168　1/32
1987 年 2 月第一次印刷　印张：18 7/8
印数：0001—4,600　字数：491,000

统一书号：15031・774
本社书号：5075・15—8

定价：5.30 元

译者的话

本书是一本计算机科学与软件工程的基础教材．它是由美国著名计算机科学家 W．A 伍尔夫等人经过多年的科学研究和教学实践，在原讲稿基础上不断充实修改而成的．

本书按照程序设计技术与基本数学概念相结合的原则，全面系统地论述了控制结构、数据结构及其相互作用．本书在内容和结构上，新颖实用、深入浅出、注重理论在实际中的应用，它不仅适宜作为计算机科学和计算机软件专业的大学生或研究生的教材，而且对于计算机科学的研究人员和从事程序设计的各行业的科技人员，也是一本理想的参考书。

全书共分四部分。第一部分(第一章至第六章)讨论基本控制结构，包括有限状态自动机、框图、程序设计语言中的控制结构等控制模型、控制的表示、程序的形式描述与证明、计算效率的测定等．第二部分(第七章至第十二章)讨论基本数据结构，包括数据数学模型、数据类型、抽象数据结构、数据的表示及其正确性、的空间分析等．第三部分(第十三章至第十八章)讨论控制与数据的相互作用，包括计算模型与文法、标识符的解释、高级语言的运行时表示、递归、递归程序的证明、递归算法的分析等．前三部分是按数学模型、程序设计语言概念、表示、正确性、效率这样的顺序来论述的．第四部分(第十九章至第二十三章)是实例研究，通过几个实例说明前三部分内容在实际程序设计中的应用．前三部分每章末都附有文献概述，供读者进一步研究用．全书每章后都附有习题，这些习题大多是有实用价值的．

卢道全同志校阅了大部分译稿，吴鹤龄同志也审校了部分译稿，译者在此表示衷心的感谢．

限于译者水平，译文难免有错误和不妥之处，敬请读者批评指正。

序

本书是计算机科学方面的一本中等程度的教科书．它适用于至少有一至两学期的程序设计和解题实践并想要研究较高级的计算课题的学生，当然，最好还学过一学期的离散数学．

本书的内容和组织基于这样一个前提——**程序设计是一门工程学科**．同别的工程学科一样，程序设计与所构造的事物有关，尤其是与具有重大实际价值的**实在**事物有关；同时，也同别的工程学科一样，**好的**程序设计来源于科学的应用．掌握这门学科，对于一个好的程序员来说是必不不可少的．

本书也是有关"好的程序设计科学"方面的一本入门书，它包含了许多数学概念．传统上，我们仅把程序组织、数据结构、性能分析等软件概念当作程序设计技术进行教学，而在本书中，我们把程序设计思想同自动机、形式语言、数据类型这些基本数学概念联系起来描述这些程序设计技术，并通过例子来说明这些数学概念与实际构造的关系，以及如何把这些概念应用到实际构造中去．

尽管本书中的许多材料曾按传统的方式作为大学高年级和研究生的课程进行过教学(以较深的程度)，但是我们认为，在开始阶段介绍这些材料是必要的和适宜的．就象学习计算和物理是大多数工程学科的必备课程一样，这里给出的材料为软件工程和高级计算机科学课程奠定了一个适当的基础．这些材料的组织，见第 xviii 页的课题组织表以及该表前面的说明．

本书的用法

最初我们曾把本书设计为计算机科学的一门选修课，自 1974 年以来，把它的多种版本作为大学二年级一个学期的课程讲授．

要在一学期内教完本书的全部内容，时间是不够用的．但在一学期内，我们一般可以教完第一、二部分的大部分内容和第三部分关于理论的内容，还可讲完第四部分中的几个含有实例的章节．作业大多是有实用价值的程序设计习题．虽然在一学期内要讲这么多的内容，但是我们看到，学生对这些内容的学习没有多大困难．

虽然我们深信，本书中的材料为计算机科学的选修课程最好能早点给出，但本书也有许多其它用途．例如，

- 作为两学期课程的基础．本书所包括的材料用作一学年的课程是足够的．然而，在这门课程中，我们不仅要教完这些材料，而且要强调高级程序设计技术和更高级的数据结构表示．
- 与软件工程课程结合．本书中的大部分材料都是严格论述软件工程的形式前提的．可以把形式模型、功能描述、验证、性能分析等章节作为三年级或四年级软件工程的半学期课程．
- 并入理论计算机科学课程中．本书中的许多材料都是以离散数学和形式逻辑知识为基础的，因此构成了关于形式数学概念的有意义的应用．
- 作为有实践经验的程序员自学训练的材料．当今大多数有实践经验的程序员都是在只积累计算技巧的情况下受教育的．现在已不再是这种情况了．为了不低于目前的大学毕业生的水平，有经验的专业人员应当掌握这里给出的内容，特别是较为形式的内容．

本书中所用的语言基本上是 PASCAL．然而，我们的教材适用于一般的程序设计，而不是任何特定的语言，因此，我们在给出的一些材料中常与 PASCAL 有所不同．附录 C 总结了主要的不同点．在含有实例的章节中程序都用 PASCAL 编译程序执行过．这个编译程序最初是由西德汉堡大学在 DEC PDP-10 机上开发的．但是，我们又竭力避免在我们的实例中含有那个编译程序所提供的特有的语句．

计算机科学在过去的十到十五年中已经取得了巨大的发展．它已不再是一门简单的技术或一种特别的程序设计技巧的积累．现在，它已成为一个真正的科学和数学的结合体，未来的程序员必须懂得并使用这个结合体．本书很难说是这个课题的最后的总结，但它对初学者来说是个良好的开端，对有经验的专业人员来说是一本参考资料集．

W. A. 伍尔夫　　M. 肖

P.N. 希尔芬格　　L. 弗朗

宾夕法尼亚州，匹兹堡1980年4月

前　言

程序设计是一种工程活动

程序设计是一种工程活动．因此，同其它好的工程一样，好的程序设计来源于把科学精心地应用到实际问题中去．然而，为了了解本书特有的内容和组织的理论基础，我们首先需要研究程序设计与其它工程相似及不同的方法．

当然，计算机只是一种工具，但它是一种“通用”工具．因为它不仅仅是为解决某一个问题而设计的，而是专门用程序（计算机执行的一系列指令）来解决一个特定问题的．程序设计是编写程序的活动，实际上，它是构造一种专用工具的活动．正是这种对“构造事物”的强调引出了我们的前提．

然而，软件工程与其它各种工程之间有着重要的差别．大多数工程师必须研究他们所用材料的物理限制，例如，重量和长度的限制．在这些限制范围内完成一个可行的设计常常是工程师所面临的中心问题．虽然计算机本身也有些限制，例如存贮量和处理机速度的限制，但这些限制与程序规模的大小没有多大关系．程序设计中所遇到的限制却大多与我们自身，即人的能力有关：

我们编写的程序的主要限制常常是由不理解这个设计所引起的，而不是由计算机的物理性能引起的．

使用本书的大多数学生也许编写过 100 行的程序，并对这个工作的复杂性有所感受．现在考虑一个 5000 行的程序，这样的程序是中等规模的，但相对于某个实在的程序还是很小的．5000 行程序只是 100 行程序的 50 倍，因此，如果 100 行程序要花费一周时间来完成，读者可能会猜测 5000 行程序大约需花费 50 周的时间，即大约一年的时间．

遗憾的是，通常要花费更长的时间．编写一个程序所需要的时间与它的复杂性有关，而与它的规模只有间接的关系．问题在于，我们需要产生的不是50个彼此无关的100行程序，而是50个相互关联的100行程序．如果不对这些相互关联性适当地加以控制，就会引起更加复杂的情况，产生5000行程序所需的时间将从若干周增长到若干年．

正是在复杂性方面的这种非线性增长，更准确地说，是人在处理复杂性方面所受的限制，构成了软件工程的核心问题．软件工程师使用的最重要的工具和技术是控制复杂性的工具和技术．本书的内容是这些工具和技术的科学基础，这是不足为奇的．

复杂性和抽象

程序不是人类必须处理的唯一的或最复杂的系统．例如，国民经济就更为复杂，即使一个简单物体中分子的运动也比最复杂的程序复杂得多．然而，人不能完全了解国民经济或分子结构的全部细节．为了处理复杂的情况，使用一个强有力的方法——抽象．我们情愿忽略复杂系统的细节，而建立只反映某些重要的宏观行为性质的模型．例如，牛顿的运动定律是实在事物的一个物理模型．这个模型是从物体的元粒子的复杂运动中抽象出来的，并描述了它们整体行为的总性质．

几乎在所有的场合下物体中分子的精确运动都是无关紧要的．唯一相关的信息是通过以某种方式概括个体运动来表示的．例如，我们说物体的速度(实际上是它的所有分子的平均速度)和温度(分子运动动能的一种度量)，等等．当我们把许多细节概括为某一性质时(例如，速度或温度)，我们就是从这些细节中进行抽象．

在某些场合下，这些总的抽象可能是不够的．例如，晶体的结构依赖于其分子的性质．用于描述结晶性质的抽象必须比用于描述碰巧是晶体的物体温度的抽象要详细．如果我们想要了解化学

反应，就必须考虑一个更细的模型，即物质的原子结构．请注意下述很重要的问题．在每个这样的情况下，我们只不过使用另外的**模型**，另外的实在事物的**抽象**．每个模型恰好包含有足够的细节来解释所研究的现象．为了分析一个物体的运动或温度，我们完全可以忽略其分子结构．例如，为了分析其简单的化学性质，我们可以忽略亚原子质点的波状性能．只有当我们去研究原子核的反应时，才需要一个更具体的粒状模型，即量子力学．

那么，很显然，潜在于现代科学中的一个深刻的哲学假设是：通过理解一组模型，可以理解实在事物的复杂性，这些模型有的描述宏观行为而忽略了细节；有的则逐步地解释不断加细的微观行为．虽然这个简化的假设是否完全正确尚需讨论，但我们有限的智能**迫使**我们这样做．没有这种假设，我们就不能处理我们周围的复杂事物．

让我们回到程序设计上来．在这里，模型和抽象的概念也起了主要作用，但情况也有些不同．对于化学家和物理学家来说，他们关心的是实在事物，他们提出模型来描述和预言实在事物．对于他们来说，一个模型成功的程度能通过实验和把结果同从模型推导出的预言进行比较来测试．另一方面，程序员构造人造事物．他也可以使用抽象来控制复杂性，但原则上，他有能力使这些模型精确．这就是说，他能使程序在所有的情况下都能按模型所预言的那样准确地执行．因此，从某种意义上说，模型和抽象的概念在程序设计中甚至比在物理学中更为有用．可是，实际上，用在程序功能描述中的抽象描述，在某种程度上几乎总是不完全的，这就象在物理学中一样．例如，我们很少指定程序的精确运行时间．此外，正如化学和物理的物理系统说明模型的层次那样，每个都比后者更详细．程序设计也可应用类似的分层结构．一整个程序通常太复杂，以致不易理解，因此我们将用一个抽象模型来描述它．而所定义的模型又必须用相当复杂的程序来实现．一般来说，我们必须用更具体的模型来解释每个子程序．因此，在一个大型程序中，应当期望找到许多抽象的层次．只有这样，我们才有希望避免

复杂性剧增.

抽象和结构

在明确了抽象和模型的必要性以后，我们需要考虑如何构造好的抽象或模型.在物理学中一个称为"奥克姆剃刀"的原理,很早就被作为在竞争的模型间进行选择的标准.实际上,奥克姆剃刀的原理所论述的是"挑选能充分描述一种现象的最简单的模型". 这种对简单性的强调是重要的. 记住，采用模型总的理由是要适应我们人类在处理复杂性方面所受到的限制. 判断好的程序设计抽象的标准也是简单性. 我们应当选择简单(容易理解)而又足以描述所需程序行为的抽象.

经典的玻尔原子是一个简单模型的极好例子. 它只有两部分: 原子核及其沿轨道运行的电子. 这两部分又可用它们的重量和电荷来刻划，而它们之间的关系可用控制轨道运动的简单定律来描述.

在程序设计领域里，我们处理的部分是数据和程序. 它们之间的关系诸如下述: 如何控制从程序的一部分到另一部分的流程，以及程序的每个部分对它所使用的变量做出何种假设. 作为科学家、工程师和程序员,我们致力于使各个部分及其之间的关系尽可能简单,尽可能好.

好的程序设计方法

构造一个好的程序是没有绝对规则可循的，而只有好的工程设计都必须具备的一般性质:它必须很好地完成它所要求的功能，而且必须是合算的. 然而，有一组通常导致一个好的工程程序的一般准则. 这些准则经常称为(1)结构程序设计学,(2)程序设计方法学,或(3)软件工程学. 本书中的许多材料在于为这些学科提供坚实的基础. 这是本书的主要观点.

编写(和理解)大型程序的困难，可能表现在问题语言与程序设计语言之间概念的差距上．例如，一个编目控制问题谈论的是部件号码、再订购地点、销售额、供给量、仓库容量，等等．在我们用于写最终程序的程序设计语言中没有提供这些概念，而我们谈论的是整型和实型变量、数组和记录、子程序和循环．程序设计的任务是用外部(程序的用户)模拟问题概念的方法来使用程序设计语言．因为在这两者之间有很大差距，所以这种模拟可能要使用大量的程序代码．

同许多其它领域一样，对程序设计进行清楚明了的陈述有助于程序设计困难的解决．如果这个困难是问题语言与程序设计语言之间概念方面的差距，那么我们必须缩小这个差距．我们不能指望设计一个对所有问题都适合的程序设计语言，但是我们可以概括出一个具有相同作用的程序设计方法．

1．分析问题，写出构成一个解答的精确描述．

2．规定适合于表示问题解答的抽象，并且用它们写一个抽象程序．

3．分析提出的抽象程序；特别是，

■验证这个抽象程序满足问题的功能描述．

■验证这个抽象程序的空间和时间耗费对于它的假想用户来说是可接受的．

4．实现比第2步更低层次的抽象．

为了实现第4步，我们可以再引用上面的四步过程．于是，这种方法引入了几个抽象的层次，每一层次都是良构的．

当然，这是理想化的陈述，因为实际上这些步骤并不总是按这种次序进行的，它们之间不可避免地要有些重复．然而，我们努力使抽象程序尽可能地清楚和简单，而这种抽象程序是用具有简单关系的少数简单部件构造出来的．由于这种抽象程序是用与所说明的问题相同的术语写的，因此程序的规模大小应与任务的固有复杂性有关，而与现代程序设计语言和计算机的花样无关．一般说来，这种程序将出奇地小，并当给出抽象概念的描述时，抽象程

序正确性的形式化的数学证明是可行的．如果这个抽象程序不能呈现出好的结构，一般则是由于它包括得太多的缘故，那么就给程序员发出一个明显的信号，指示他返回并重新审查用于表示该程序的抽象．

只有当抽象程序写完之后，我们才转到如何实现它所使用的抽象概念的问题上．最终每个抽象概念在目标程序设计语言中都有一个连接的模拟，当它对所有的抽象程序来说都正确时，就完成程序．然而，这种情况常常是不容易达到的，我们需要找出实现这些抽象概念的某种方法．但是注意，已经出现了两个重要的事情：

- 第一，我们已经把原问题分成一些较小的、精确定义的子问题，而且每个子问题可以在相对分离的情况下继续工作．
- 第二，每个子问题类似于原问题，它们都可以用同样的方法去解决．特别是，子程序仍然可能在概念上与目标程序设计语言有很大的差距，以致于不允许一个**可理解的**直接实现．然而，有些抽象概念对于子问题是自然而又具体的，它们足以表示问题的解答．因此，我们能写出另一个抽象程序，规定它所使用的抽象，验证它，实现它的抽象概念，等等．

本书的组织与内容

本书分为四个主要部分：第一部分讨论**控制结构**，第二部分讨论**数据结构**，第三部分讨论这两者的相互作用，第四部分的每章都有一个实例，用以说明这些材料在现实程序设计构造中的应用．

在前三部分中，我们设置了同样的结构：

- 首先，引入适于深刻理解主要课题的数学工具和模型．
- 其次，讨论出现在程序设计语言中的主要课题．
- 再其次，讨论**表示**，这个概念通常与新程序员无关，但它却是程序设计和分析中最重要的概念之一．
- 最后是关于程序正确性与效率的推理方法：关于程序做什么（功能描述和确认），以及如何做好它（性能分析）．

在这个前言后面，是说明各课题之间关系的一个表格．本书中的每部分都列为表中的一行，相应的章作为列．表中的项目是各章中说明的课题．在前三部分中，任何一行或任何一列的课题间都有密切的联系．我们的材料是按下列顺序排列的：首先讨论控制方面的所有课题，然后讨论数据方面的所有课题，等等．然而，除了我们需要较早引入的实际问题外，在给出关于程序设计语言的讨论之前，我们均讨论控制、数据及其相互作用的数学模型，即，我们把对主要结构的强调放在列的课题上，而不是行的课题上．

虽然我们希望按章节顺序来读本书，但读者应当记住呈现在表中的关系．例如，控制与数据的表示实际上不是无关的，它们是以重要的而有时又是微妙的方式相互作用的．当读到数据的表示那一章时，通过回顾前面关于表示控制的问题，读者会从本书中学到更多的东西．

课 题 组 织 表

	基本数学模型	基本的程序设计语言概念	扩充的程序设计语言概念	表 示	关于正确性的推理	关于效率的推理
第一部分：控制	1 ■有限状态模型：有限状态机，正规表达式 ■模型的等价性 ■模型的局限性	2 ■框图 ■简单的控制结构 ■与有限状态模型的等价性	3 ■扩充的控制结构：选择，重复，程式	4 ■基本结构的表示 ■扩充结构的表示	5 ■功能描述 ■最弱前置条件	6 ■阶算法 ■实验代价测量 ■代价前置条件
第二部分：数据	7 ■存贮器的模型 ■数据类型 ■描述技术	8 ■纯量类型，包括引用 ■构造类型：向量，记录 ■其它问题	9 ■抽象构造类型：栈，排队，集合，图 ⋮	10 ■表示技术 ■第九章中类型的其它形式的表示	11 ■抽象数据类型的验证 ■指针的语义	12 ■静态空间分析 ■动态空间分析
第三部分：控制与数据的相互作用	13 ■非受限状态模型：下推自动机，图灵机，产生式文法 ■模型的可计算性与局限性	14 ■动态数据类型 ■递归 ■递归程序设计技术	15 ■作用域 ■作用期 ■结合，参数机构	16 ■程序的运行时表示：活动记录，静态存贮分配、栈式存贮分配、堆式存贮分配之间的比较，层次显示表	17 ■递归程序的归纳证明	18 ■递归关系的解答
第四部分：实例研究	19 词法分析程序	20 集合的快速实现	21 在集合上的重复	22 符号求导	23 表达式的符号简化	

目　　录

第二部分 基本数据结构