内 容 简 介

本书分为数值算法和非数值算法两部分，介绍了数据结构、各种算法以及数学分析法与程序的概念、原理、相互关系和应用。数值算法部分包括多项式与线性代数方程组，矩阵与非线性方程，插值、逼近及其应用，数字信号处理，小波变换等内容。非数值算法部分包括线性表、栈、队列和串，树，图，排序、查找与文件操作，并行算法等内容。本书附录部分还介绍了电子商务系统中的加密算法、用于图像处理的并行计算机结构特征以及算法在数据压缩中的应用、COM 原理和 Web 服务的标准与组织。

本书可作为非计算机专业数据结构（及算法）等课程的本科生教材，也可作为相关专业的研究生或 MBA 人员的参考书。

图书在版编目(CIP)数据

数值算法与非数值算法/康晓东主编. —北京：电子工业出版社，2003.1
高等学校公共课计算机教材
ISBN 7-5053-8405-8

Ⅰ.数… Ⅱ.①康… Ⅲ.数值计算－计算方法－高等学校－教材 Ⅳ.O245

中国版本图书馆 CIP 数据核字(2002)第 105503 号

责任编辑：章海涛
印　　刷：北京李史山胶印厂
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×980　1/16　印张：21.5　字数：468 千字
版　　次：2003 年 1 月第 1 版　2003 年 1 月第 1 次印刷
印　　数：4 000 册　　定价：28.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：(010)68279077

目　录

数值算法部分

① “*”部分可根据专业特点和需要选学，全书同。

非数值算法部分

导　言

信息技术离不开计算机科学的支持。一方面，信息的表示是计算机科学的基础；另一方面，计算机科学的发展为科学计算及数据处理提供了高速和高精度的计算工具。计算机只能机械地执行人的指令，它本身不会主动地进行思维，也不能发挥任何创造性。因此，人们在用计算机解决问题之前，首先要进行程序设计。

目前，大多数计算机程序的主要目标或是完成运算，或是存储和检索信息。从存储空间和运行时间的实现角度来看，这些程序必须组织信息，以支持高效的信息处理过程。因此，研究和数据结构算法以有效地支持程序的实现就成为计算机科学的核心问题。

通常，程序设计主要包括两个方面：行为特性的设计和结构特性的设计。所谓行为特性的设计，通常是指将解决问题的过程的每个细节准确地加以定义，并且将全部的解题过程用某种工具完整地描述出来。这一过程也称为算法的设计。所谓结构特性的设计，是指为问题解决确定合适的数据结构。数据结构与算法之间有着密切的关系。特别是对于数据处理问题，算法的效率通常与数据在计算机内的表示形式有着直接的关系。

在实际工作中，技术人员需要不断处理问题、算法和计算机程序。这是3个不同的概念。

(1) 问题

从直觉上讲，问题无非是一个需要完成的任务，即对应一组输入有一组相应的输出。问题的定义中不能包含有关怎样解决问题的限制。只有在问题被准确定义并完被全理解后，才能研究问题的解决方法。问题的定义中应该包含对任何可行方案所需资源的限制。对于计算机要解决的任何一个问题，总有一些直接的或者间接的限制。例如，任何计算机程序只能使用主存储器和可用的磁盘空间，而且必须在合理的时间内完成运行。

从数学角度讲，可以把问题看做函数。函数是输入(即定义域)和输出(即值域)之间的一种映射关系。函数的输入可以是一个值或是一些信息，这些值组成的输入称为函数的参数。不同的输入可以产生不同的输出，但是对于给定的输入，函数得到的每次输出必须相同。

(2) 算法

算法是指解决问题的一种方法或者一个过程。如果将问题看做函数，那么算法就能把输入转化为输出。一个问题可以有多种算法，一个给定的算法解决一个特定的问题(如进行某种计算)。

掌握一个问题多种解法的优点在于，可以用不同的解法对某个问题的几个特定变量更有效地组合，或者使对同一个问题的不同输入更有效。例如，有的排序算法适合于数目

较少的序列，有的算法适合于数目较多的序列，而有的算法则适合于可变长度的字符串。

(3) 程序

一个计算机程序被认为是对一个算法使用某种程序设计语言的具体实现。由于任何一种现代计算机程序设计语言都可以实现某个算法，所以可能有许多程序都是同一个算法的实现(当然，一些程序设计语言可能使得编程人员的工作更容易一些)。定义一个算法时，必须提供足够多的细节，以便必要时转化为程序。

算法必须是可终止的，不是所有的计算机程序都是算法。例如，操作系统是一个程序，而不是一个算法。但可以把操作系统的各种任务看成是一些单独的问题，每一个问题由一部分操作系统程序通过特定的算法来实现，得到输出结果后便终止。

问题是一个函数，或是输入和输出的一种联系。算法是一个能够解决问题的、有具体步骤的方法。算法步骤必须无二义性，算法必须正确且长度有限，必须对所有输入都能终止。程序在计算机程序设计语言中是算法的实现。

0.1 数据结构

直观地说，数据是描述客观事物的数字、字母和符号，是计算机程序使用和加工的“原料”。数据的基本单位是数据元素，性质相同的数据元素的集合叫做数据对象。数据对象中的元素彼此之间存在的相互关系叫做结构。

数据结构指的是数据之间的结构关系，它包括数据的逻辑结构和数据的物理结构。数据的逻辑结构是仅考虑数据元素之间的逻辑关系，它包括线性结构(如线性表、栈、队)和非线性结构(如树、二叉树和图)。数据的物理结构是指数据元素在计算机存储器中的表示，即存储结构，如向量、链表等。

一种逻辑结构通过映像便得到它相应的存储结构。同一种逻辑结构可以映像成不同的内部存储结构。反之，数据的存储结构一定要反映数据之间的逻辑关系。

0.1.1 数据的逻辑结构

数据的逻辑结构是独立于计算机的，对数据元素之间的逻辑关系的描述。从集合的观点看，它可以形式地用一个二元组 B = (D, R)表示。其中，D 是数据元素的集合，R 是 D 上关系的集合。可见，数据的逻辑结构有两个要素：数据元素和关系。

数据元素是独立的信息。它可以是一个单独的符号，如英文字母(A, B, …, Z)，也可以由若干个数据组成。

关系是指数据元素间的逻辑关系。它可以是线性的(对数据元素而言，只有一个前趋和一个后继)，也可以是非线性的(对数据元素而言，或者有一个前趋，多个后继；或者有多个前趋，多个后继)。

数据的逻辑结构按关系分为线性结构和非线性结构。线性结构包括线性表(典型的线性结构)、栈和队(特殊的线性表是具有特殊限制的线性结构，特殊限制是指数据运算只

能在表的一端或两端进行)、串(特殊的线性表,其特殊性表现在它的数据元素仅由一个字符组成)、数组(线性表的推广,它的数据元素是一个线性表)、广义表(线性表的推广,它的数据元素是一个线性表但不同构,即或者是单元素,或者是列表)。非线性结构包括树(具有多个分枝的层次结构)和二叉树(具有两个分枝的层次结构)、有向图(一种网状结构,边是顶点的有序对)和无向图(一种网状结构,边是顶点的无序对),如图 0-1 所示。

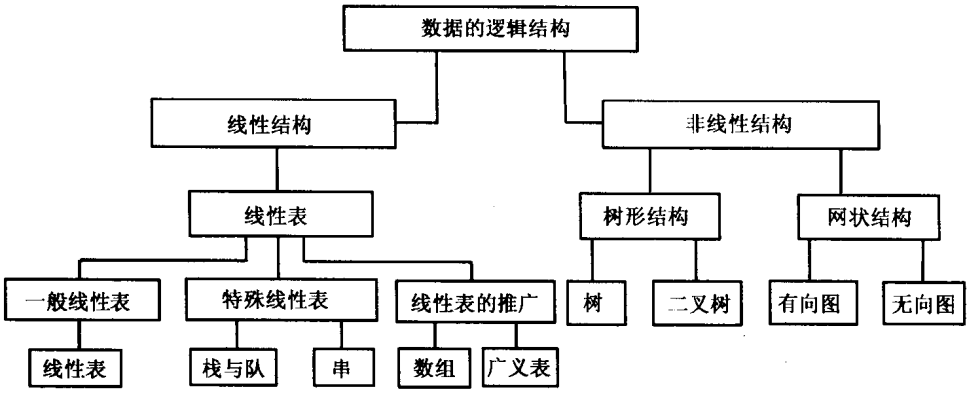

图 0-1　几种逻辑结构

0.1.2　数据的物理结构

数据的物理结构是指数据的逻辑结构在计算机中的映像,即存储表示。映像包括数据元素的映像和关系的映像。数据元素的映像是结点,即用一个结点表示一个数据元素。结点是数据结构讨论的基本单位。关系的映像有两种:顺序映像和非顺序映像。

数据的物理结构(即存储结构),按关系的映像分为顺序存储结构和非顺序存储结构。顺序存储结构是逻辑上相邻的数据元素存储在物理位置上相邻的存储单元里,元素的关系由存储单元的邻接关系来体现。非顺序存储结构是数据元素可以在计算机内任意位置上存放(它不要求逻辑上相邻的元素在物理位置上也相邻),它们的逻辑关系用指针来链接。所以,非顺序存储结构又叫链式存储结构。链式存储结构将数据元素存放的存储单元为两个部分,分别用来存放数据和指针,称之为数据域和指针域。

顺序存储结构主要包括向量(典型的顺序存储结构,一组相邻的连续单元)和数组。链式存储结构是链表,按指针域的个数分为单链表、双向链表和多重链表。

指针,即地址。程序中的变量名、下标地址都是指针。指针是数据结构中关键的概念,是许多数据结构得以实现的基础。链式存储结构就是用指针来实现逻辑结构与存储结构的映像。指针的灵活应用将有助于许多优雅算法的产生。

无论是什么样的存储结构,实际上计算机系统仅为其提供一个最基本的存储方式,即一维数组。也就是说,任何计算机系统的主存可以看做是一个一维数组。尽管大多数高级语言的编译程序还提供了二维数组、三维数组,但实际存储仍是一组连续单元。

较高维数的数组,同样按一维数组存储,通过建立的各种下标计算公式进行访问。所

以,任何一种存储结构都有两种状态:一种是逻辑状态(从用户的角度看),一种是物理状态(从计算机的角度看)。

值得指出的是,选择某个结构和选择某个结构的表示是不同的。前者是为解决某个问题,在对问题理解的基础上,选择一个合适的逻辑结构表示出数据的逻辑关系;后者是对这个逻辑结构为适应求解(即运算的需要),选择一个恰当的存储表示。前者的选择是面向问题,后者的选择是面向机器。而"面向问题"的数据的逻辑结构向"面向机器"的数据的存储结构转换的问题,正是数据结构所要研究的。

0.2 算法

对于一个问题,如果可以通过一个计算机程序,在有限的存储空间内运行有限长的时间而得到正确的结果,则称该问题是算法可解的。但算法不等于程序,也不等于计算方法。程序可以作为算法的一种描述,但程序通常还需考虑许多与方法和分析无关的细节问题,因为在编写程序时要受到计算机系统运行环境的限制。通常,程序设计的质量不可能优于算法的设计。

0.2.1 算法的特征

一般来说,算法中的各种运算总是要施加到各个运算对象上,而这些运算对象又可能具有某种初始状态,这是算法执行的起点或依据。因此,一个算法执行结果总是与输入的初始数据有关,不同的输入将会有不同的结果输出。如果输入不够或输入错误,则算法本身也就无法执行或执行有错。可见,只有当算法拥有足够的信息时,该算法才是有效的[①]。

算法实际上是一种抽象的解题方法,它具有动态性。算法应具有以下 5 个特征:

- 确切性:算法的每个规则、每个操作步骤都有确切的含义,即无二义性。
- 可执行性:算法的每一个操作,计算机是可执行的。
- 有穷性:算法必须在有限步骤后计算出结果。
- 结果性:一个算法必须取得一定的结果。
- 一般性:算法是求得某一类问题的解,而不只是一个特殊的、具体问题的解。

算法的执行总是与特定的计算工具有关,而每一种计算工具的有效数字的位数总是有限的。因此,在实际的数值计算过程中,所有参与运算的数值通常是近似的。

数值型算法还应具有以下 5 个特点。

(1) 理论上的精确运算与实际运算之间存在着差异。算法与计算公式是有差别的,数学上的一些运算规则、恒等变换公式在算法中不一定适用。

---

① 所以说,算法是一组严谨地定义运算顺序的规则,并且每个规则都是有效且是明确的,此顺序将在有限的次数下终止。

(2) 理论上的解题方案与实际能用性之间存在着差异。有些数学上的计算公式实际上不能真正用于计算。这些公式本身是正确的,但参与公式中各种运算的数值,由于受计算工具的有效数字的限制,都是近似的。也就是说,它们都是有误差的,而误差在计算过程中将会产生各种影响。

(3) 精确解法与近似解法往往没有本质区别。所谓近似解法,一般是指迭代解法。前面已经提到,在算法执行过程中,参与各种运算的数值通常都有误差,而这些误差通过运算后最终会影响结果。因此,精确解法所得的结果未必准确,而近似解法所得的结果未必不准确,两者没有本质的区别。

(4) 大多数问题不能简单地用某个标准算法来解决。在工程中,对于同一类问题中的某个具体问题,其算法也往往是千差万别的,不可能用一个算法去解决所有的问题,要根据不同问题的特点来设计不同的算法。

(5) 小规模问题往往存在快速算法。在数值计算领域中,有些小规模问题的计算过程已经被人们所熟悉并经常使用,对于这些计算过程通常还可以减少运算次数,从而在大量调用这些计算过程时可以大大减少计算工作量;有时还可以利用这些过程来构造解决大规模问题的快速算法。

例如,计算 $A = ac + ad + bc + bd$,按照通常的方法需要做4次乘法和3次加法。但如果将上式写成下列等价形式:

$$A = (a + b)(c + d)$$

则只需做1次乘法和2次加法。显然,在需要对不同数据 a, b, c, d 来计算 A 的时候,用后一种等价形式来代替原来的算式是具有实际意义的。

又如,两个复数相乘通常是这样来运算的:

$$e + \mathrm{j}f = (a + \mathrm{j}b)(c + \mathrm{j}d) = (ac - bd) + \mathrm{j}(bc + ad)$$

可见,需要4次实数乘法才能完成1次复数乘法。但实际上,两个复数的乘积结果中,e 和 f 可以表示成

$$e = ac - bd = (a - b)d + a(c - d)$$
$$f = bc + ad = (a - b)d + b(c + d)$$

若令

$$P = (a - b)d, Q = a(c - d), R = b(c + d)$$

则

$$\begin{cases} e = P + Q \\ f = P + R \end{cases}$$

可见,为了计算两个复数乘积中的实部 e 和虚部 f,只需做3次乘法就够了。当然,这是以增加加减法次数为代价的,但这也是合算的,因为做1次乘法所花的时间一般要比做1次加减法的时间多得多。

0.2.2 算法的描述与评价

1. 算法的描述

算法描述语言是表示算法的一种工具，它只面向用户，不能直接用于计算机，但很容易转换为计算机上可执行的程序。为了方便，有的算法也可能直接用自然语言或流程图的形式来描述。在用算法描述语言描述一个算法时，如果被描述的算法可以用过程来实现，则一开始总是先要对输入与输出参数进行必要的说明，并且第一行总是为：

PROCEDURE 过程名(输入/输出参数表)

在算法正文中，必要时，对算法中的每一行用自然数依次编上行号，以便在叙述中对算法进行说明。

常用的算法描述语言中的各个语句说明如下。

(1) 符号与表达式

符号是以字母开头的字母和数字的有限串，用以表示变量名、数组名等，必要时也用来表示语句标号。语句标号后应跟随一个冒号，然后是语句。

算术运算符通常用 +，−，*，/和↑，关系运算符用 =，≠，<，>，≤和≥来表示，逻辑运算符用 AND(与)，OR(或)和 NOT(非)来表示。

(2) 赋值语句

赋值语句的形式为

$a \leftarrow e$

其中，a 表示变量名或数组元素，e 表示算述表达式或逻辑表达式。

(3) 控制转移语句

无条件转移语句用如下形式：

Goto 标号

条件控制语句有以下两种形式：

If C Then S

或

If C Then S1

Else S2

(4) 循环语句

循环语句有两种形式：While 语句和 For 语句。

在算法描述中，还可能要用到其他一些语句。例如，Exit 语句通常用退出于某个循环，使控制转到包含 Exit 语句的最内层的 While 语句，或转到 For 循环后面的一个语句；Return 语句用于结束算法的执行；Input(或 Read)和 Ouput(或 Write)语句用于输入和输出。

算法中的注释可以用一对方括号括起来。几个短语句可以写在一行上，但彼此之间

用分号隔开。复合语句总是用一对花括号括起来作为语句组。

设计一个算法以后，要对它进行描述，以便交流、阅读。算法描述包括对以下两个内容的表现：

- 算法的逻辑结构
- 算法的实现(施加在数据结构上一系列操作的表述)

如果是大型复杂问题的算法，则算法描述还将分为两部分：第一部分是算法结构的描述，通常用算法结构图描述各项处理功能及它们之间的逻辑关系(各子算法间的接口)；第二部分是子算法描述。

算法描述的语言是设计语言(一种示意性语言)。这种语言只描述算法的设计思想，即仅考虑便于交流，而不考虑计算机是否有相应的翻译系统。当前描述的语言有两大类：一是图示法，即流程图；二是语言描述法，如 Pascal，Pidgin Algol 及自然语言等①。

2. 算法的评价

从根本上讲，给定一个计算机系统后，一个程序就拥有两个资源：时间和空间。它们的使用受到三个因素——硬件、算法和数据——的影响。硬件很大程度上随所使用的硬件或软件配置而定，数据量大小随问题规模而定。因此，利用好程序的两个资源，算法将起很大的作用。算法首先要求是正确，还应是高效率的，即占用内存空间少，需要运行的时间短。

评价算法优劣的是算法分析②。算法分析的内容包括：

- 误差与运算误差分析
- 算法稳定性分析
- 算法的复杂度与最优性
- 算法的自适应性

数据结构与算法之间的本质联系表现在失去一方，另一方将没有任何意义。数据结构是为了研究数据运算而存在的；算法是为了实现数据运算，即实现数据的逻辑关系的变化，或是在这个结构上得到一个新的信息而存在的。进一步讲，对于数据结构而言，若不了解施于数据上的算法，就无法决定实施算法的数据结构；对于算法而言，若不了解基础的数据结构，就无法确定施加在数据结构上的操作(即算法)。

数据结构与算法的本质联系是：

---

① 在程序设计语言的发展史上，20 世纪 60 年代末是承上启下的重要时期。这个时期有三种重要的语言问世，一是由 2001 年图灵奖得主 Ole-Johan Dahl 和 Kriisten Nygaard 所创建 Simula 67；二是由 IFIP 组织欧美一批顶尖计算机科学家共同设计的 Algol 68，以及由 IBM 公司为和 360 系列机配套而联合两大计算机用户组织 SHARE 和 GUIDE 共同开发的 PL/I。这三个语言各有特色，均有所创新，都对后来的程序设计语言产生了重大影响。但客观地说，Simula 67 的面向对象概念影响是最巨大而深远的。它本身虽由于比较难学、难用而未广泛流行，但在它的影响下所产生的面向对象技术却迅速传播开来。

② 算法分析的标准是算法的时间复杂度和空间复杂度。

算法 + 数据结构 = 程序

计算机程序是人们利用计算机解决某一个问题时，向计算机提供的、用一种程序设计语言书写并能被计算机执行的一组规则和步骤。程序具有一定的功能，能回答“做什么”，且还有供计算机执行的一系列操作，用于回答“怎样做”。程序能被计算机运行，运行的对象是数据，运行的目的是加工、处理数据，以得到问题的解答。

程序包括两个内容：数据（按一定方式存储的数据）和操作（处理数据的过程）。因此，程序的研制实际上是数据结构与算法的研究。在程序的研制中，对数据结构来说，要研究两个问题：一是求解问题的数据的逻辑结构的研究，二是数据的逻辑结构向物理结构转换（即映像）的研究。

同样，在程序的研制中，对算法来说，也要研究两个问题：一是算法结构的研究，对于大问题更是如此，要分解为若干个子问题，研究子问题间的接口关系；二是数据运算，即施加在数据结构上的研究。数据结构与算法的研究共同基于同一个观点，即降低计算复杂度。在这方面，两者的关系是：一个好的数据结构将决定一个好的算法，一个好的算法必定建立在一个好的数据结构上。

同时，在数据结构中稍复杂一些的算法设计中可能用到多种技术和方法，如算法设计的构思方法，动态变量及链表，流程图及其变换方法，算法编码，递归技术，与特定问题相关的技术等。某一环节运用不好，都可能会影响到算法的设计。

一个用数据结构与算法结合研究非数值问题的研究模式如下：

- 方法：解决问题的方法。
- 实施方法的数据结构（一种静态的数据结构）：通过对问题环境的理解，确定面向问题的数据的逻辑结构，再将它转换成面向过程的数据的存储结构。
- 算法结构：对于大型复杂的问题将它划分为各个子问题，即子算法，并描述各个子算法间的接口关系。
- 工作的数据结构：根据运算需要进而设计的各种动态的数据结构，如索引表、栈、队等。
- 子算法设计：为了完成子功能，施加在数据结构上的一系列操作。

0.2.3 并行算法

在数据密集的计算中，越来越多的应用对计算机的性能提出了更高的要求。正是这些应用构成了研究未来并行计算机所需要的技术基础。

过去，单节点计算机性能的提高是靠制造速度更快的器件和采用越来越多的隐式并行操作来实现的，如大容量内存储器和新型总线。未来计算机器件的速度的提高将主要取决于有效开发并行性（包括隐式和显式）的能力。同时，高网络带宽的需要和可伸缩设计的商业限制，也使人们联想到一种分级互联网络，而对称多处理器设计中的最新进展，

使其成为分级网络中节点的理想选择。机器的并行要求和分级结构[①] 为重新考虑算法的设计和分析提供了依据。

最终，计算在这一级上表达成利用这种结构的每个节点紧耦合处理器的、纯粹的共享存储器的并行算法。

所谓并行算法，是指适合于在各种并行计算机上求解问题的算法。它是一些可同时执行进程的集合，这些进程互相作用和协调处理从而达到对给定问题的求解。可以从不同的角度将并行算法分为：数值并行算法和非数值并行算法，或者同步并行算法、异步并行算法和分布式并行算法，或者 SIMD 并行算法、MIMD 并行算法和 VLSI 并行算法等。

数值并行算法：其计算是基于代数运算（+，-，×，÷）的一类计算问题（如矩阵运算、多项式求值、线性方程组求解等）的求解算法。

非数值并行算法：基于关系运算（如排序、选择、搜索等）问题的求解算法。

同步并行算法（Synchronized Parallel Algorithms）：某些进程必须等待别的进程的一类并行算法。因为一个进程的执行依赖于输入数据和系统中断，所以全部进程均须同步，在一个给定的时钟，以等待最慢的进程结束。运行在 SIMD 机器模型上的并行算法属于同步并行算法，其处理器之间的同步由硬件实现。

异步并行算法（Asynchronized Parallel Algorithms）：各进程之间不必相互等待的一类并行算法。此时，进程之间的通信是通过动态地读取共享存储器的全局变量来实现的。运行在 MMD-SM 机器模型上的并行算法即属于异步并行算法，进程之间的同步通常由软件（程序）实现。

分布式并行算法（Distributed Parallel Algorithms）：由通信链路连接的多节点（计算机）并行协同完成某个计算任务的一类并行算法。运行在 MIMD-CL 机器和虚拟共享存储器的 MIMD-DVSM 机器模型上的算法均属于分布式并行算法。

VLSI 并行算法：在 VLSI 计算模型上发展的一类算法。

对于并行算法，除了要研究算法所需要的运行时间之外，还需要研究算法所需的处理器数目，以及研究两者在最坏的情形下与问题规模的变化关系。

0.3　程序设计

程序设计语言是用来编写程序的工具。

目前，世界上经常使用的计算机语言虽然只有十几种，但已经设计和实现的却有上千种之多。这些计算机语言可以分成两大类：一类是因不同的计算机主机而异的汇编语言，另一类是通用的程序设计语言。前者称为低级语言，后者称为高级语言。

汇编语言依赖于具体的机器，可移植性差，不易推广。高级语言接近于自然语言和数

① 分级结构的分级特性是：编程工具应允许分级的编程模型。这样一来，编程人员就可以把精力放在开发特定的软件上，而不必顾虑整个分级的结构细节。

学语言,在各种机器上都可以通用。高级语言不论在算法描述的能力上,还是在编写和调试程序的效率上,都远比低级语言优越。

1. 程序的编译

就目前的冯·诺依曼模式的计算机而言,其硬件只懂自己的指令系统,即只能直接执行用相应机器语言编写的代码程序,而不能直接执行用高级语言或汇编语言编写的程序。因此,要在计算机上实现除机器语言之外的任一程序设计语言,首先应使此种语言为计算机所"理解"。解决这一问题的方法有两种:一种是对程序进行翻译,另一种是对程序进行解释。

所谓翻译,是指在计算机中放置一个能由计算机直接执行的翻译程序,它以某一种程序语言(源语言)所编写的程序(源程序)作为翻译或加工的对象,当计算机执行翻译程序时,就将它翻译为与之等价的另一种语言(目标语言)的程序(目标程序)。"源"和"目标"这两个术语总是相对于特定的翻译程序和翻译过程而言的。如果一个翻译程序的源语言是某种高级语言,其目标语言是相应于某一计算机的汇编语言或机器语言,则称这种翻译程序为编译程序。汇编程序也是一种翻译程序,它的源语言和目标语言分别是相应的汇编语言和机器语言。

可见,按翻译方式在计算机上执行用高级语言编写的程序,一般需经过两个阶段:第一阶段为编译阶段,其任务是由编译程序将源程序编译为目标程序,若目标程序不是机器代码,而是汇编语言程序,则还需汇编程序再行汇编为机器代码程序;第二阶段称为运行阶段,其任务是在目标计算机上执行编译阶段所得到的目标程序。在执行目标程序时,一般还应有一些子程序配合进行工作,常见的数据格式转换子程序、标准函数计算子程序、浮点解释子程序、数组动态存储分配程序、下标变量地址计算机子程序等都属此类。这些子程序组成一个子程序库,称为运行系统。运行子程序库中的各个子程序,大都按模块化的结构来编制。显然,库中的子程序愈丰富,各子程序的功能愈强,编译程序本身就愈明确紧凑。

编译程序与运行系统合称为编译系统。

源程序的编译(或汇编)和目标程序的执行不一定在同一种计算机上完成。当源程序由另一种计算机进行编译(或汇编)时,将此种编译(或汇编)称为交叉编译(或汇编)。

图 0-2 简单地表示了按编译方式执行一个高级语言程序的主要步骤。

用高级语言编写的程序也可以通过解释程序来执行。解释程序也以源程序作为它的输入,它与编译的主要区别是:在解释程序的执行过程中,不产生目标程序,而是解释执行源程序本身。这种边翻译边执行的方式工作效率很低,但由于解释程序的结构比编译程序简单,且占用内存较少,在执行过程中也易于在源程序一级上对程序进行修改,因此一些规模较小的语言(如 BASIC)也常采用此种方式。然而就目前的情况来看,纯粹的解释程序并不多见,通常的做法是把编译和解释结合起来。有的先将源程序作为某种易于进行解释执行的中间语言形式,然后再对此中间语言程序进行解释执行;有的甚至在进行上

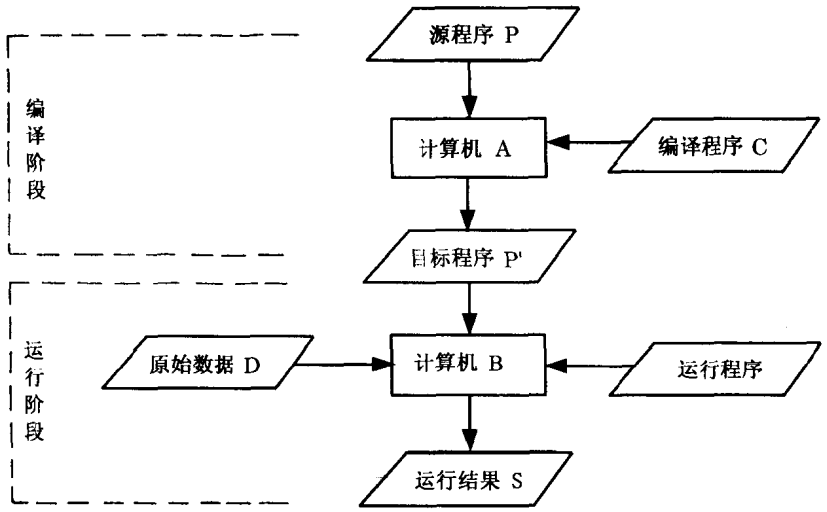

图 0-2　计算机执行高级语言程序的步骤

述翻译时，在采取上述措施后，解释程序执行效率不高的缺陷将有可能得到弥补。

2. 程序的应用与发展

高级语言还可以从不同角度再分类。

从应用范围来看，高级语言有通用语言和专用语言之分。目标非单一的语言称为通用语言，如用于数值计算方面的 FORTRAN 语言。目标单一的语言称为专用语言，如数控语言 APT、系统模拟语言 MIMIC 等。

从使用方式来看，高级语言有交互式和非交互式之分：具有反映人—机交互作用的语言，如 BASIC 语言；而 FORTRAN，Pascal 等语言则属非交互式语言。

从编程方法来看，高级语言有过程式语言和非过程式语言之分。例如，FORTRAN 语言、Pascal 语言、C 语言等均属过程式语言。因为在程序设计时，必须用语言的语句、函数、命令等一步一步描述解决问题的处理过程。而有的语言，如人工智能语言 PROLOG，在编程时，只要说明和定义输入、输出及要求做的工作即可，因此，这种语言被称为非过程式语言。

几种常用的语言及其主要的应用场合分别是：

- FORTRAN(FORmula TRANslation，公式翻译语言)主要用于科学计算。
- C 语言是较靠近机器而又不被机器所束缚的一种程序设计语言，多用于系统软件的开发及公用程序设计。
- Pascal 是一种结构化程序设计语言，主要用于构造数据结构的场合。
- PROLOG(PROgramming in LOGic)是一种处理逻辑问题的语言，也是一种人工智能语言，常用于专家知识的描述和模拟。
- 面向对象的 Java 语言现常用于网络环境。