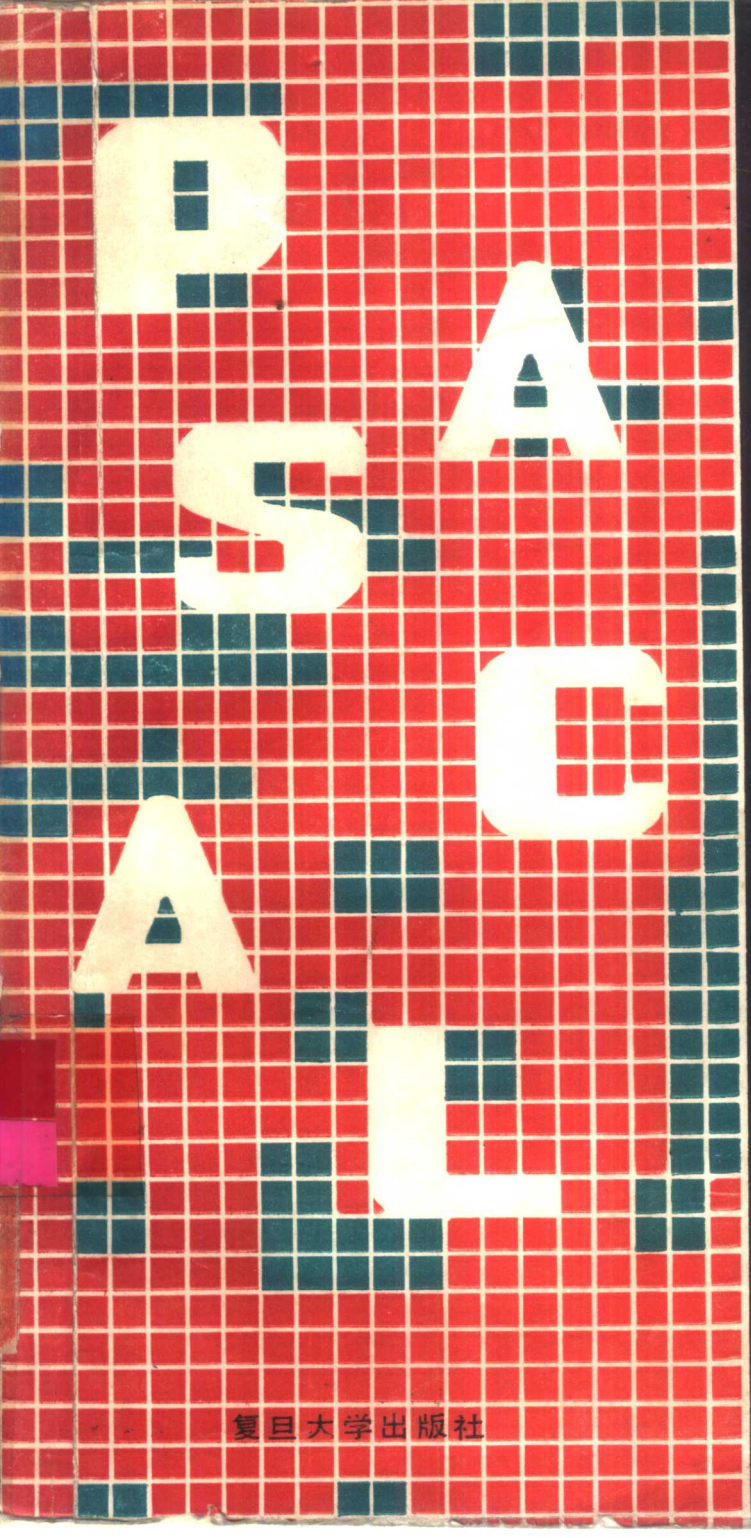

PASCAL程序设计

夏宽理编

复旦大学出版社

PASCAL程序设计

夏宽理编

复旦大学出版社

PASCAL 程序设计

复旦大学出版社出版
新华书店上海发行所发行
复旦大学印刷厂印刷

字数 377 千　开本 850×1168　1/32　印张 12.75
1986 年 10 月第一版　1986 年 10 月第一次印刷
印数：1——10,000

书号：13253·42　　定价：2.60 元

内 容 简 介

本书详细介绍了Pascal程序设计语言，并系统地讨论由算法到程序的逐步求精程序开发技术。

全书共十二章，第一章简单介绍程序设计的基本概念；第二章至第四章介绍Pascal语言的基本成分；第五章提出由算法到程序的逐步求精程序开发技术；第六章至第十一章深入介绍Pascal语言的高级成分；第十二章为程序设计例子，是前面各章知识特别是逐步求精和子程序开发技术的综合应用。

本书的特点是列举的例子和选用的习题有一定的难度，例子和习题提示采用由算法到程序的逐步求精和子程序的程序开发技术求解，注重程序开发能力的训练。

本书可作为高等院校程序设计课程的教材和教学参考书，也可作为软件水平考试补习班的教材和参考书。

前　言

计算机程序设计语言 Pascal 是由 N.Wirth 教授等于六十年代末提出的。Pascal 语言开发的目标是使程序设计教学更有系统，能用该语言方便地表达数值计算和非数值计算程序设计技术。Pascal 语言是较系统地体现了 E.W.Dijkstra 和 C.A.R.Hoare 等提出的结构程序设计思想的第一个程序语言。因 Pascal 语言简单且具有较强的表达能力，用它能写出结构良好的程序，在现代计算机上实现高效率和有较好的可移植性，Pascal 语言作为一种实用语言和理想的程序设计教学语言已被迅速推广使用。

近年来，国内已有介绍 Pascal 语言的著作出版，但多数类同于早期介绍 $ALGOL_{60}$ 程序设计的书籍，重点在介绍语言，较少讨论程序设计方法。作者受近年来有关结构程序设计和程序设计方法学等文献的启发，本书的重点在于讨论程序开发技术。本书系统地介绍从算法到程序的逐步求精程序开发技术，并用大量例子说明其应用。读者将会看到这种程序开发技术应用方便、自然。

作者编写本书的目的是让读者熟练掌握一种结构程序设计语言和实用的程序开发技术，使读者在用高级语言开发程序的能力方面受到良好的训练。

本书的正文可分成三部分。第一章至第五章是程序设计的基础部分。这部分包括程序设计的基本概念，Pascal 语言的标准数据类型、输入输出和程序控制结构等设施，以及逐步求精程序设计方法。

第二部分为第六章至第十一章，介绍更多的数据类型和子程序。Pascal 语言具有较强的表达能力，主要是因为它有丰富的数据类型。用子程序开发程序也是一个极重要的程序设计技术，只有熟练使用子程序设计程序的人才能算是称职的程序员。

最后第十二章是程序设计例子。用已学的程序设计知识编制有一

定难度的程序，是从算法到程序逐步求精程序开发技术的更综合的应用。

各章后附有习题。有些习题（用＊注出）在附录中有提示。供读者参考。

本书原是作者为讲授 Pascal 程序设计而编的教材，经过多次试用，效果比较满意。初稿写成于 1979 年，在 1982 年进行了较大的修改，这次出版前又作了进一步的增删。限于作者水平，文中欠妥及谬误之处，敬请读者指正。

招兆铿老师审阅了本书，并提出了许多宝贵意见。在此深表谢意。

作者 1984 年 4 月

目　录

第一章　程序设计基本概念

1.1　计算机和程序

计算机从产生到现在只有近四十年的历史，然而它已广泛地应用于各行各业。如计算机用于企业管理，用计算机分析实验数据，用计算机模拟心理活动，用计算机辅助有人或无人的空间探索。因便宜的集成电路的发展，个人计算机变得更为实用。计算机已成为新的工业革命的主要标志。虽不能预见它对社会的最终影响，但是已清楚地表明在我们面前将有巨大的变化。

未使用过计算机的人，通常有一个错误的概念，认为计算机就是求解问题的机器。事实上，只有当为计算机编制了解决问题的程序，在计算机上运行这个程序时，计算机才能求解问题。如果没有程序控制，计算机是不会做任何有意义的事情的。在一些工业发达国家里，软件开发已是一个新兴行业。就在最近几年内，我国也将形成一支软件开发的产业军。

在深入讨论程序设计之前，首先讨论与程序设计有关的计算机的基本特征。

一台计算机组成的示意图见图 1.1。它由存贮器（内存）、处理机和若干外围设备组成。组成存贮器的基本单位是单元（或字节）。每个单元与一个编号（称为单元的地址）相关联，单元可存贮数据或程序。

处理机包含指挥操作的控制器和执行计算的运算器。程序员关心的是指示程序执行位置的指令计数器和为了临时寄存计算结果和数据等目的的寄存器。

计算机配有外围设备以实现外部信息输入计算机或计算机输出信息到外部，如键盘、显示器、卡片阅读机、行印机等，它们是人机交

换信息的最基本设备；大容量的外部信息存贮设备如磁盘、磁带，是计算机有限存贮器的扩充。

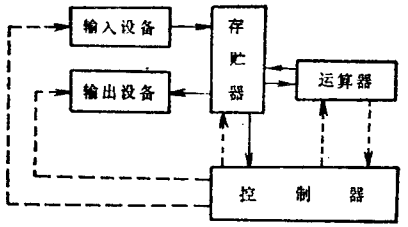

图 1.1 计算机结构示意图

计算机是一种能精确地遵照人们按某类计算过程所指定的规则而运行的自动装置。一个最重要的特征是它有若干基本指令。每条指令或指明对选定的操作对象执行指定的操作，或实现某种特定的控制。计算机之所以功能强及应用范围广，在于它能可靠地执行长长的指令序列。现代计算机一般每秒能执行几十万、几百万以至上亿条指令。计算机以如此高的速度反复按以下步骤执行指令：

（1）控制器从存贮器读取下一条指令；（2）指令被译成电子信号；（3）相应的信号送到运算器、存贮器或外围设备执行指令规定的操作；（4）回到第1步继续工作。

计算机的指令往往是非常简单的。例如实现求表达式

$$a*b+c/d$$

的值。设某架计算机有如下三种指令：

（1）从存贮器中读取操作数到寄存器；（2）对寄存器中的数加以运算，把结果存于某寄存器中；（3）把寄存器中内容存入存贮器。

现把实现上述计算的动作加以分解，并用该机器指令描述如下（设有两个寄存器 R_1 和 R_2）：

$R_1 \leftarrow a$ {从存贮单元 a 中读取数值存于寄存器 R_1}

$R_2 \leftarrow b$

$R_1 \leftarrow R_1 * R_2$ {寄存器 R_1 和 R_2 中的内容相乘，并把结果存于寄存器 R_1 中}

$z \leftarrow R_1$ {把 R_1 中内容存于存贮单元 z 中，z 表示用来存放中间结果的存贮单元}

$R_1 \leftarrow c$

$R_2 \leftarrow d$

$R_1 \leftarrow R_1/R_2$

$R_2 \leftarrow z$

$R_1 \leftarrow R_1 + R_2$

为了叙述方便，上面已把指令写成便于理解的形式。实际上，存于计算机中的指令都是机器代码形式。用计算机指令表示某类计算过程，往往有几千条甚至几万条这样的指令。称实现某类计算的指令序列为程序。

在此还要指出程序和执行程序的区别。程序在计算机上的运行就是计算机在执行程序。程序的运行也称进程。程序与进程的关系正如乐谱和演奏的关系，乐谱是音乐程序，执行音乐程序就是演奏。演奏的效果是美妙动听的音乐声。

指令、程序、进程有以下性质：（1）操作指令需指定操作对象。程序就是要施操作于某些对象。进程则实现指定的操作。计算机程序的操作对象称为数据。（2）为了便于人们理解和计算机识别，程序伴有指明数据属性的说明。（3）程序是个静态实体。它有一个明确的目标。执行程序的进程是动态的，执行的结果就体现程序目标的实现。（4）组成程序的指令序列中，有些顺序执行，某些指令组可能被重复执行多次。程序也可能含有判定的指令，进程执行该指令时，根据判定标准作出正确的判断。

在一个计算机系统中有效的程序集组成该系统的软件，而称计算机本身为硬件。采用这两个名词，主要是强调程序与计算机一样的重要。名字的直觉含义形成对照：硬件是可见的，实在的东西。计算机系统的硬件不易改变；软件通常处于不断改变的状态中。

软件最重要的部分之一是操作系统。它包含常驻存贮器的控制程序集合。操作系统完成许多准备和运行一个用户程序所需要的例行任务。如在多用户系统中，确定下一个运行的用户，准备它的输入，把用户程序装入存贮器，为用户程序的执行分配计算机时间，帮助用户程序完成输入输出操作，等等。

1.2 程序设计语言

直至五十年代后期，人们都用计算机指令序列来描述某类计算过程。用这种方法编制的程序的特点是：（1）程序员面向具体机器编程序，必须认真考虑机器细节。一种机器上的程序不能移植到不同种类的机器上执行。（2）这样编制的程序很难交流。为了充分发挥机器性能，程序所含的指令与具体的机器密切相关。这样，欲理解同行们设计的程序是非常困难的，甚至常常难于解释自己编制的程序。所以要发现早期程序的错误、修正错误以及改动程序等都很困难。从考虑算法、编制程序直至把程序调试正确，需化费很长的时间。

以上的缺点导致了称为高级程序设计语言的发展。实际上，程序是实现某个目标的一种手段。程序员凭它与执行者计算机通讯。通讯需要一种语言。虽然自然语言（例如汉语）也常常用于描述非形式的指令，但大多数行业都有它们自己的语言。如作曲家、编织师都设计了完全独特的语言用来传达他们的指令。一台计算机为了能与人通讯而定义的指令集就是它的语言。称计算机的指令集为机器语言。用机器语言编制的程序为机器语言程序。人们不能简单地采用自然语言（如汉语）写程序。这是因为程序要详细、精确地描述计算动作序列。而应用自然语言不能达到这个要求。事实上，汉语的伟大是她的应用范围广阔，而她的细微差异和意义含糊的可能性，以及隐喻，在此不得不认为是缺点。当讲字句的本义、精确性和完整性时，自然语言恰与机器语言极其相反。我们需要选择这样的一种程序设计语言，它有自然语言易读和应用的普遍性，又有机器语言的直率和精确。相对于机器语言称为高级语言。Pascal 语言就是一种高级程序设计语言。

然而一个用 Pascal 那样高级语言写的程序计算机不能立即执行。必须把 Pascal 程序翻译成等价的机器语言程序，才能使它被执行。这里涉及三个程序：用户用 Pascal 语言写的程序正文，称为源程序；与源程序等价的机器语言程序，称为目的程序；以及把源程序翻译成目

的程序的编译程序。

在计算机上运行一个 Pascal 程序，分成两个不同阶段：

(1) 翻译

运行程序 = 编译程序

输入数据 = 源程序

输出数据 = 目的程序

(2) 执行

运行程序 = 目的程序

输入数据 = 问题的输入数据

输出数据 = 计算结果

当人们用某种高级语言写程序时，常常把编译程序和计算机看作一台能直接执行源程序的单一机器。为了表示与实际的计算机相区别，把一个编译程序和计算机的结合体称为虚机器。一台计算机配上了一个 Pascal 编译程序，就构成了一台 Pascal 虚机器，简称 Pascal 机。本书中所有可运行的程序就是指在 Pascal 机上可运行的程序。

用高级语言写的程序也常常有错误。主要的程序设计错误有三类：(1) 在翻译过程中发现的错误。这类错误由编译程序报告。报告通常有错误性质的编号，发现错误的源程序位置。然而，编译程序可能被错误引入歧途，而使一个错误被延迟发现，在正确的程序处发现了另一种错误。(2) 在执行过程中发现的错误。这类错误常使正确的计算不能继续进行。如求平方根时发现自变量的值为负等。(3) 在翻译和执行过程中不能查出的错误。仔细的用户可能发现程序的计算结果不合理。该类错误往往是程序员把计算公式写错。如 a+b 写成 a−b 等。

除了程序有错之外，数据也可能有错。如准备输入数据 5，而粗心地打入了 4 等。数据的错误可能以程序错误的第 2、第 3 类表现出来。所以，为了得到一个正确的结果，程序和数据必须都正确无误才行。

通常，一个好的编译程序，同时也提供调试功能。在调试情况下，编译程序在翻译时插入许多错误诊断指令。执行这样的目标程序能为

用户提供改变值的变量和改变后的值，以及程序执行的控制轨迹等信息。

任何一种程序语言包括两方面的内容：它的语法（句法）和意义（语义）。为了正确地使用程序语言，就应对这两方面都有正确的认识。Pascal 语言的语义用自然语言描述，而它的句法则用句法图表示。如 Pascal 语言中的标识符。标识符的语义解释为“标识符用来命名程序和程序中的对象。标识符是以字母开头，后接任意个字母或数字的字符序列。”标识符的句法图见图 1.2。这个句法图可理解为沿着箭头，从入口到出口，能在该图中通过的最长字符序列，即为标识符。Pascal 编译程序在编译过程的某处期望源程序有一个标识符时，就用该句法来识别。表 1.1 给出若干源程序正文被标识符句法识别的结果。当由标识符句法来识别源程序正文时，首先要求第一个字符为字母，否则出错。以后连续的字母或数字字符都被标识符句法所接受，直至遇到其他字符时才结束标识符识别。

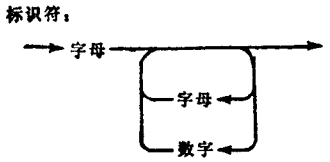

图 1.2　标识符句法图

表1.1 标识符句法识别结果表

源程序正文	识别结果	终止字符
AB05 •	AB05	'•'
1AB5	出　错	'1'
A - B	A	'_'
- A	出　错	'_'
AB　DB	AB	' '{空白符}

1.3 算　法

为了用计算机解决某类问题，必须为该问题编制一个程序。然而在编制程序之前，需要知道问题的求解方法。接着才能按求解方法用机器语言或高级语言编制程序。非形式地，把求解问题的方法称为“算法”。算法是计算机科学中的一个重要概念。一个算法指明一个

计算过程。算法由操作步骤序列组成。它有明确的开始点。算法执行从这开始点开始，直至遇到它的终点才结束。要求组成算法的每个操作步骤是明确的，通常还要求算法的执行结果是确定的；执行算法所化费的时间必须是有限的。一个算法应完成原先赋予它的目标，即还要求算法是有效的。对实现同一要求的几个算法的优劣的评价标准有多种。主要以实现算法所需要的空间少、执行算法所化费的时间短、算法思想容易理解、实现简单等标准来衡量。

为了能使人理解一个算法，允许哪些基本成份（术语）来描述算法，是一个重要问题。很显然，基本成份的选取是和人与计算机（虚机器）的水平有关的。应选择这样一些操作指令作为描述算法的基本成份。使得读算法的人能立即理解该算法，执行算法的虚机器能高效地完成计算。为了使基本成份完备，往往包含一些很基本的操作指令。而为了理解方便及容易构造算法，就要求引入控制结构能方便地用基本成份来描述各种复杂计算。

下面是本书采用的描述算法的基本操作指令集：（1）各种复杂的算术、逻辑表达式；（2）给变量置值，写成“变量：=表达式”；

（3）读外部数据给变量，写成“read（变量$_1$，…，变量$_n$）”；

输出变量的值和文字，写成“write（……）”；

另外，还提供以下控制结构。其中左边是书写形式，右边是它的框图表示。

（1）**顺序控制**：顺序执行它的操作指令。

BEGIN{开始}

操作$_1$；

⋮

操作$_n$

END

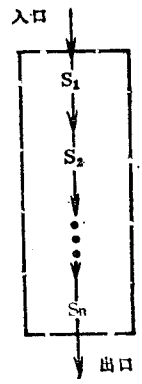

图 1.3　顺序控制→