高等学校教材

PASCAL

语言程序设计

钟伯刚 主编

Higher Education Press

高等学校试用教材

PASCAL语言程序设计

钟伯刚　主编

高等教育出版社

内 容 提 要

PASCAL 语言是使用最广泛的一种结构程序语言．本书从程序设计的观点出发，对 PASCAL 语言的语句、数据类型等基本概念及其使用方法作了详细介绍．通过大量例题，阐述了自顶向下、逐步求精、层层构造算法的程序设计方法，使读者对结构程序设计思想有明确具体的了解．全书共九章，每章后都附有较多习题．在附录部分还介绍了几种常用机型对标准 PASCAL 语言的扩充．

本书是为高等院校非计算机专业编写的程序设计语言教材，也可作为计算机专业的教材或参考书，还可供有关科技人员阅读．

高等学校试用教材

PASCAL 语言程序设计

钟伯刚 主编

*

高等教育出版社出版

新华书店北京发行所发行

人民教育出版社印刷厂印装

*

开本 850×1168 1/32　印张 18.25　字数 410 000

1989 年 2 月第 1 版　1989 年 3 月第 1 次印刷

印数 0001—6 331

ISBN7-04-002542-6/TP·58

定价 4.60 元

前　言

PASCAL 语言是目前使用最广泛的几种计算机高级语言之一，它是由瑞士的 N. Wirth 教授于七十年代初在 ALGOL 60 的基础上提出的. 它的语句功能强、数据类型丰富. 用它编写出的程序清晰明了、易读易懂、运行效率高，因此受到普遍的欢迎和重视，被广泛用于科学计算、事务处理和编写各种系统软件.

由于 PASCAL 语言的成分体现了结构程序设计的基本思想，它特别适于培养学生运用结构程序设计方法来开发程序的能力和良好的程序设计风格及习惯. 此外，与 PASCAL 语言相近的 ALGOL 语言在我国已流行多年，加上 PASCAL 语言目前已几乎配置在所有计算机上，所以在我国，特别是在大专院校中，有必要也有可能普及和推广 PASCAL 语言，把它作为各类专业的计算机教学的一种重要的语言. 为此，我们为非计算机专业编写了这本书.

本书在编写上力求通俗易懂，由简到繁，循序渐进. 从使用的角度对计算机系统结构和程序设计基础知识作了简单介绍. 从程序设计的观点出发，对 PASCAL 语言的语句、数据类型等基本概念及其使用方法作了详细分析讨论. 书中附有大量例题，以帮助提高读者的编程能力. 所以在讲解具体程序时，先采用自顶向下、逐步求精的方法层层构造算法，然后用已学过的 PASCAL 语言的语句和数据结构来描述算法，实现程序的分层和模块结构. 从而使读者对结构程序设计方法有明确具体的了解，逐步地引导读者用这种先抽象、后具体，先粗后精的思维方法来分析问题、设计算法、编写程序.

本书主要讲解标准 PASCAL 语言，但在附录部分介绍了几种常用机型对标准 PASCAL 语言的扩充.

本书是根据全国高校计算机基础教育研究会提出的大纲，由华东分会组织编写的. 参加编写的有：江西工业大学钟伯刚（第一、二、三章），上海冶金专科学校黄安健（第四章及附录），江苏工学院韩国华（第五章），南京机械专科学校谢培均（第六、七章），南京工学院罗军舟（第八、九章）. 全书由钟伯刚主编. 初稿完成后，全国高校计算机基础研究会的谭浩强、谢柏青、陶士清等同志对书稿提了许多宝贵意见. 根据这些意见我们又进行了修改，最后由清华大学杨德元副教授审阅，并再次进行了修改. 特别要感谢谭浩强教授，他对本书的编写给予热情支持和大力帮助. 沈树铭副教授对本书的编写也做了许多有益工作，黄安健对书稿的誊抄和校稿做了不少工作，其它兄弟院校的老师也给予我们诸多帮助，在此一并致以衷心感谢.

由于水平所限，书中难免有错误和不妥之处，敬请广大读者批评指正.

编　者

1987. 11.

目　录

第一章 绪 论

计算机是大量存储和快速处理各种信息的工具.从它问世以来发展迅猛,它的应用已渗透到人类社会的各个方面,因此计算机的应用水平是一个国家现代化的重要标志.要想使用计算机,让它做任何一点事,都必须编制程序,所以,学习计算机程序设计知识是各种专业人员的必修课.

本章首先介绍电子数字计算机(简称计算机)系统的基本知识, 然后对 PASCAL 语言的概况作一简介.

§1.1 计算机系统基础

先讨论有关二进制的一些问题. 在目前的计算机中, 数和其它所有信息的表示都采用二进制. 在十进制中,10 是基数,所以计数用的数码也有十个,它们是 0,…,9. 表示具体一个数是用位权法,小数位、个位、十位、百位等,所对应的位权分别为$\cdots 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, \cdots$, 主要的计数法则是逢十进一. 仿此,在二进制中, 2 是基数,数码只有 0 和 1 两个,从低位到高位的位权分别为$\cdots 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, \cdots$,负指数表示小数点后的位权,主要的计数法则是逢二进一.

在计算机中采用二进制,主要是它的运算法则比较简单,数码只有两个, 所以数的表示也比较容易,只需要两个状态分别代表两个数码,在电子计算机中就是用电位的高低和脉冲的有无来表示二进制的两个数码.

计算机系统由硬件和软件两部分组成.

一、硬件

组成计算机系统的物理设备称之为硬件,硬件包括: 存储器、中央处理机(CPU)、输入输出设备.

(1) **存储器** 存储器是计算机的记忆装置,用来存放各种信息(即二进制信息),它由许多按顺序排列的存储单元组成. 每个单元按顺序给一个编号, 称为它的地址. 一个单元通常由八个二进位(bit)组成,称为一个字节. 若干个字节组成一个字,微型或小型机通常是一个字节或两个字节组成一个字,大中型机则更多些. 总的字节数称为存储器的容量,用 K 作单位来表示,$1K = 2^{10} = 1024$ 个单元. 如 48K,128K,512K 等.

从物理结构上看, 存储器又分内存储器(有时也称为主存储器)和外存储器. 内存储器存放系统常用信息及当前要处理的信息和数据,它速度快,容量较小, 在硬件布局上它通常和中央处理机放在一起称为主机. 外存储器是存放当前暂不处理的信息和数据,它速度较慢但容量大,磁盘、磁带都可作为外存储器,它可以放在主机内部,也可独立成为一台相对主机而言的外围设备.

对存储器进行的主要操作是按地址存取信息和数据, "存取"有时也称为"访问". 上面谈到的速度,就是指存取信息和数据的速度,它是存储器的主要性能指标之一. 高速和大容量一直是

存储器发展的主要战略目标.

(2) **中央处理机** 中央处理机由运算器和控制器组成. 计算机科学的理论已经证明，只要运算器具有加法和移位功能，就可完成各种计算. 因此运算器的主要部件是加法器和移位寄存器，它的基本操作就是加法和移位.

控制器是计算机的神经中枢，它用一条条的指令控制各部分执行各种操作. 指令是进行最基本、最简单操作的命令. 执行一条指令，机器的有关部件和线路就要完成某些特定的基本动作，因此指令是机器硬件能直接执行的最小操作单位. 与其它信息一样，指令也是用二进制数码(即0,1序列)来表示的. 任何一台计算机都有一个指令系统，少的有几十条指令，多的有几百条. 所以，要机器算题，就必须设计一系列指令交给机器去执行，这种指令序列就是程序. 程序是人们要求计算机完成某项工作意图的形式表示，利用有限的指令，可以编出各种各样的程序，让计算机去完成各种各样的工作. 这就如同作家可以利用有限的文字和词汇写出各种感人的作品一样. 指令如同文字和词汇，人和机器打交道必须通过指令，所以指令也是一种语言，称为机器语言.

(3) **输入输出设备** 输入输出设备是用于向计算机输入信息和由计算机输出结果的. 常用的输入输出设备有纸带输入机、读卡机、终端键盘、打印机、显示器及绘图仪等.

二、软件

由于机器语言是由二进制代码构成的，直接用机器语言编写程序很不方便. 因此，出现了用符号来表示指令的符号式程序设计语言，这种语言称之为汇编语言，例如，用英语单词词头来表示指令的操作，ADD 代表加、MOV 代表传送等，这样表示的指令含义直观清楚，便于记忆和阅读，大大地方便了编程. 但是机器只能识别用二进制代码表示的机器语言，因此用汇编语言编写的程序必须翻译成机器语言才能在机器上执行. 在计算机发展的早期，翻译都是由人来做，称之为代真，现在几乎都是由机器来做. 专门设计一个称之为汇编程序的程序，由它把由汇编语言编写的程序翻译成机器语言程序，然后这个机器语言程序就可在机器上执行. 因此，配有汇编程序的机器，可以看成是在实际机器级(其机器语言由硬件实现)上出现的"虚拟"机器级，其语言为汇编语言.

"虚拟"二字的含义是该虚拟机器级的语言(在这里是汇编语言)，没有直接的硬件支持，不能直接在机器上执行，必须经过翻译才能在机器上执行. 也就是说，只有配有汇编程序的机器才能执行用汇编语言写的程序. 遵循这样的思路，人们提出了更高的要求，从算题的目标出发，希望能用数学语言来直接书写程序. 因此又出现了面向题目的高级语言，如 FORTRAN, ALGOL, PASCAL 等. 用这些高级语言写的程序不依赖具体机器，可以适用于任何一种类型的计算机. 它们可以看成是在汇编语言级之上又出现了高级语言级，它的实现是把高级语言程序翻译成汇编语言程序(或直接翻译成机器语言)，而后再逐级实现. 翻译高级语言的程序称为编译程序，用高级语言书写的程序称为源程序. 经过翻译所得到的机器语言程序称为目标程序. 目标程序在机器上运行就得到最后结果. 计算机系统的这种层次结构如图 1-1 所示.

底层是实际机器的物理设备，有时俗称为裸机．从宏观功能上看，每构造一层虚拟机就使机器能识别的语言提高一级，编程的方便性也就提高一步．

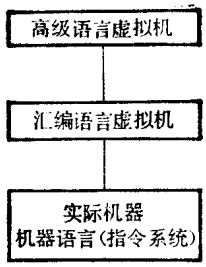

图 1-1　计算机系统的层次结构

用高级语言写的源程序，实质上就是目标程序的抽象和浓缩，它仅仅指出主要的东西，而次要的、细节的东西统统抽象和浓缩掉了．例如，要从输入机输入一个数到存储器的某个单元中，用 PASCAL 语言写出是 read(A)；这个语句同自然语言很接近，很好理解．它只指出要从输入机上输入一个数到 A 单元中，至于怎样起动输入机把一个数的各个位一步步的送到指定的单元中去，这些具体的实施细节它没有指出，这正是编译程序的任务．编译程序在进行编译时，把它进一步细化，用机器指令描述具体的实施细节．因此高级语言的一个语句，通常要对应一段机器语言程序．前者称抽象级，是面向题目的(即用户的)，后者是实现级，是面向具体机器的，用户是觉察不到的．所以编译程序也可看成是沟通抽象级和实现级之间的桥梁．另外，从方法学的角度来看，高层是低层的抽象，高层指出(用高级语言指出)要做什么，如 read(A)；而低层解决怎样做的问题，即给出相应的实现细节(一段机器指令程序)，这种抽象级-实现级的方法，在计算机技术中使用很广泛，下面我们还要用到它．

运行编译程序，以便把源程序翻译为目标程序要占用宝贵的机器时间和存储空间，因此使用高级语言是要付出代价的．然而一旦把源程序翻译成目标程序后，目标程序可以保存起来，以后要运行该程序时，只要直接运行目标程序就可以，即一次编译可换来多次运行，因此付出的代价是值得的．

高级语言的出现，在计算机的发展上具有划时代的意义，它使计算机从少数专家才可以掌握的神秘东西，变成大多数人都能使用的工具．

实现虚拟机的手段就是配置汇编程序或编译程序，它与解题程序没有直接的关系，是一种工具性的程序．通常，我们把程序和有关的文档资料统称为软件，软件又可以分为应用软件和系统软件两大类，前面所提到的汇编程序和编译程序等工具性程序都属于系统软件的范围．

由以上分析可见，要使计算机工作，不但要硬件，同时也要有相应的软件．只有硬件没有软件，计算机是不会做任何工作的，配置软件就是向计算机注入知识，所以硬件和软件都是计算机系统的资源．现代计算机系统的硬件、软件资源十分丰富、复杂，人是无法管理的．因此，计算机的系统软件还包括另一个极其重要的工具性程序，称为操作系统，它是用来控制和管理计算机系统资源、方便用户使用计算机的程序集合．操作系统是计算机系统自己管理自己的工具．从计算机系统的层次结构上看，操作系统应当紧靠机器硬件，它实际上是硬件指令的扩充．它的一条命令也是对应一段指令程序，在它之上的各级语言要使用系统内的任何资源，都是通过它发出相应的命令去执行一段程序，从而调用相应的资源．所以，操作系统是现代计算机系统的一个重要支撑软件，没有它的支持，其它程序就无法运行，因此它也是任何程序运行(生存)的不可缺少的

环境. 今天任何一台计算机都有一个操作系统，它是由计算机厂商提供给用户的. 一开机操作系统就开始运行，整个计算机系统就处于它的控制之下，通过发各种命令（这些命令是操作系统提供的），人们就可以很方便地使用计算机，所以操作系统为用户提供了一个友好的、舒适的工作环境.

以上简述了计算机系统的结构. 即计算机是由运算器、控制器、存储器和输入输出设备组成，采用二进制运算，所有信息都是用二进制表示. 这些观点是由 Von. Neumaun 1964 年提出的. 此外，Von. Neumaun 还提出了著名的存储程序原理，即任何程序，作为一种信息可以存放到存储器中长期保存，需要时可以调出来按顺序一条条地执行，从而实现了计算机自动计算. 上面所说的为计算机配置软件，就是在计算机开始工作前，把预先编好的程序和数据，用某种方法送到存储器中保存起来. 存储程序原理是现代电子计算机发展的里程碑. 目前几乎所有的计算机都是按这个模式构造和工作的，所以统称为 Von. Neumaun 型机器.

§1.2 程序设计基础

以上简单地介绍了计算机系统，一个计算机系统应是由硬件和软件组成的. 所谓软件，即是程序系统，现在都把程序的有关文档资料（如程序说明书，程序流程图，编码规格等）作为软件的重要组成部分. 从使用的观点来看，用户只能通过软件来使用计算机. 这里再强调一遍，要计算机做任何一点工作，都必须给它装上相应的程序. 从这点来说，掌握程序设计是使用计算机的必由之路. 过去掌握程序设计要经过长期的训练和实践. 评判程序的质量标准是节省存储单元，执行速度快. 随着软件规模的日益庞大，程序设计变得愈来愈复杂，常常要花费大量的资金和人力，而且研制出来的程序系统错误多、可靠性差、维护和修改也很困难. 这些问题单凭个人的经验和技巧是不可能解决的，人们迫切感到应当把程序设计作为一门科学来研究，以便用科学的、规范化的方法来指导程序设计. 另外，随着硬件性能价格比的不断上升，以追求节省存储单元为主要目标的传统的评判程序质量的观点已经改变，这就为程序设计科学的发展提供了有力的物质基础. 因此，程序设计从早期的手工作坊的技艺转变为一门新型科学，越来越受到人们的重视. 算法的研究、正确评价程序质量的标准、程序设计的方法学等问题都是程序设计科学研究的基本问题. 为了更好地学习 PASCAL 语言程序设计，对这些基本问题应有个概略的了解，以下就这些问题作一简单介绍.

一、算法的概念

通俗地说，算法就是解决某类问题的方法、步骤. 所以算法是一个有限规则的有序集合，对特定问题的任何初始输入，能一步步地按照给定的规则进行计算，经过有限步后，计算终止，输出结果. 因此算法有以下五个特点：

（1）有限性. 算法必须在执行有限步后结束.

（2）确定性. 算法中给出的计算步骤（即规则），必须是精确定义的、无二义性.

（3）能行性. 算法中的每一步骤都是基本的，可以在有限时间内做完.

（4）输入. 算法都要求有零个或多个输入信息，它们是算法要求的初始数据，取自特定的对象集合.

（5）输出. 算法都有一个或多个输出信息，这就是问题的结果.

为了求解某一问题，首先必须了解问题的实质，即已知条件是什么？要求得到什么？其次是选择、制订描述问题的数学模型，最后即可进行算法设计. 模型选择适当与否，对算法设计的质量和算法本身的效率有很大关系，因此应重视模型的选择. 算法设计是一种复杂的、创造性的脑力劳动，也是程序设计中的一个难点，往往不是一次就能成功的，需要多次反复推敲，精雕细刻，逐渐形成算法的基本思想，有时可能会发现模型不合适，要更换模型，重新设计算法. 构造了算法的基本思想后，就可以用图解或语言来描述算法，所用的语言可以是自然语言，也可以是计算机语言. 用计算机语言描述的算法就是程序，它可以在配有该语言的机器上执行，最后就可以得到所求解问题的结果.

从以上分析可以看出，算法设计是程序设计的关键，而算法的正确性在算法设计中占有突出的位置，要证明一个算法是正确的，就要证明对于一切可能的输入，算法都能在有限次计算后产生正确的输出，这是一种枚举证明法，当一个算法的可能有的输入很庞大时，是很难做到的. 因此，正确性只能是相对的，即相对于有限的输入，它是正确的. 在这种意义下，程序的测试和调试也是检验算法正确性的有效办法.

二、程序的质量

程序质量的首要问题是程序的正确性，只有正确的程序才是有意义有价值的. 因此正确性是程序设计最基本、最起码的要求，是任何一个程序设计者必须首先给予保证的. 程序的正确性是以算法的正确性为前提的. 此外，还要求设计者对所使用的算法描述语言（简称算法语言）有深入的了解，掌握其语义和语法规则，对算法进行准确无误的精细描述. 程序的测试和调试（包括编译时的查错）是检验程序正确性的重要手段.

其次是程序的可读性、易修改性. 可读性好的程序容易查错，便于交流，易于保证其正确性. 易修改性指的是对程序的某一部分进行修改时，不会对程序的其它部分产生大的影响，即修改引起的副作用是局部的.

最后是简单性和高效率. 简单程序的结构容易理解，可读性好，也容易查错和修改. 但是把一个复杂的问题简化为一系列的简单结构，决非易事，它向设计者提出了更高的要求. 高效率曾经在一个时期内是程序设计追求的一个重要目标. 高效率主要指的是程序执行的速度快，以及在运行中和程序本身占用的内存小，因此要求使用更多的技巧，这样就可能使程序的可读性差，正确性得不到保证. 所以随着软件的日益庞大复杂和硬件价格的不断下降，高效率应服从程序的正确性和可读性的观点，为越来越多的人所接受，认为这是程序设计中应遵守的一条原则.

三、程序设计的方法

如何用科学的方法来保证程序的质量，不是在程序的终结而是在程序的开发过程中就保证

它的正确性，并且不完全因人而异，这些是程序设计方法学要研究的重要课题．目前公认比较好的程序设计方法是结构程序设计方法，它是按照一组能提高程序的可阅读性与易维护性的规则而进行程序设计的方法．

结构程序设计的基本思想，可以归纳为以下几点：

（1）逐步求精的设计方法

为了解决一个复杂的问题，人们往往不可能一下就触及到问题的各个方面和细节，因此采用抽象的方法．在对问题进行全面系统的分析之后，抓住主要的、本质的东西，先设计一个初步的算法，它只提供解题算法的大致轮廓，这个算法又称为抽象算法．抽象算法的实质就是在抽象数据上实施一系列抽象操作，这些数据和操作反映了问题的本质，而所有的次要方面、细节都不考虑．当然，这个抽象算法应具备以上所说的算法的五个特点，并选用合适的语言用有序的条款（步骤）加以描述．然后，作为下一步再考虑这些抽象数据和操作的具体实现．即把抽象算法中不够具体的步骤进一步精细化．所以在抽象级只解决“做什么”，而在实现级再解决“如何做”的问题．抽象级和实现级是相对的，一个抽象级仅仅是对下一级而言的，对它的上一级来说，它就是实现级．所以由抽象级到实现级的细化过程，可以一级一级地继续做下去，一直进行到数据和操作所体现的算法能直接用计算机语言准确地描述出来为止．这个过程就是自顶向下、逐步求精、不断细化的过程．经过这一过程，一个抽象算法就变换为一个具体的、包括全部细节的、能直接用计算机语言描述的算法了，一个复杂的问题就分解为一系列的简单结构了．换句话说，逐步求精的过程就是算法不断地由抽象蜕变为具体的过程．因此，逐步求精法，要求设计者先考虑好整体和全局，再考虑局部和细节．在全局问题尚未筹划清楚之前，不必考虑局部细节，更不要匆忙地开始编程．这种方法符合人的思维方式，有助于问题的顺利解决，因而是一种较好的方法．

（2）分层结构与模块结构

这是涉及程序的宏观结构问题．由以上讨论可知，自顶向下、逐步求精、不断细化的过程是一级一级地实现的．由这种方法构造的算法，在总体结构上必须是一种分层结构的，即上一层功能的实现依赖于下一层．而每一层又可划分为若干个模块，一个模块执行一个具体的、相对来说简单得多的操作，完成某个特定的任务．所以一个模块就是一个独立的功能单元，它完全可以单独地进行设计，也可以被不了解其内部细节的人所使用．这样，一个复杂的大问题就被分解为一系列的小问题了，这种解决问题的策略可归结为：“化整为零，各个击破”．因此，一个大程序就是由若干模块按一定的层次关系装配起来的．其总体结构十分清晰，同时有很好的可扩充性和易修改性．此外，这种方法有利于进行软件的生产组织．从这里可看出分层与划分模块是使程序有一个良好的、合理的结构，是程序设计中关键的一步，对程序质量是至关重要的．

（3）三种基本程序流程控制结构

结构程序设计规定，顺序结构、判断选择结构及重复结构作为程序流程控制的三种基本结构．由它再派生出其它形式的结构．因此在模块的内部，以及组合各模块构成一个大程序时都应采用这三种基本结构形式．这三种基本控制结构将在第四章详细介绍．

（4）使程序的静态结构与动态执行情况相一致

过去由于硬件昂贵，人们为了节省存储单元和减少运行时间，想出各种各样的“技巧”，随心所欲地使用转移语句，使程序执行的过程兜来兜去，纠缠在一起，很难跟踪程序的执行过程，因此也很难查错，可读性差，破坏了程序的静态结构与动态执行情况的一致性．结构程序设计要求设计者限制使用转移语句，尽量保持程序的静态结构与执行过程相一致，反对采用令人费解的“技巧”，来追求所谓的高效率．认为只有在保证程序正确性、可靠性和可读性的前提下，尽量节省时间和空间，才是真正的高效率．

(5) 在程序设计的各个阶段应写出必要的文档资料

程序设计者应当重视技术文件的编写．因为计算机软件设计好并投入正式使用以后，并不是一成不变的．在运行中可能发现程序中有错误，要进行修改；或觉得功能不够要增加某种功能，从而要扩充程序，这些统称为程序维护．由于有程序维护，使程序好象是一个有生命的东西，在不断生长(扩充新的功能)和完善(改正错误、克服不足)，这一过程称为软件的生命期．一个软件不用了，它的生命期也就结束了．程序的维护贯穿于整个生命期，因此程序维护是软件的一个重要环节．文档资料产生于程序设计的各个阶段，它包括的内容很多，主要的有：任务书、设计说明书、程序流程图、程序清单、测试说明书、操作使用说明等．这些不但是指导程序设计的技术文件和进行技术交流的重要资料，更重要的，它是对程序进行维护的根据和必不可少的工具．因为只有运用文档资料，才能迅速准确地掌握程序设计的算法思想，从而对它进行维护．所以没有文档资料，就很难对程序进行维护．同样地，在维护过程中也必须写出相应的文档资料．当然对于简单的程序设计，它的文档资料也是很简单的，有时只要在程序单上写上简短的注解就可以．但是，作为程序设计者必须重视文档资料，养成随时书写文档资料的良好习惯．

结构程序设计方法能较好地保证程序质量，因此本书的以后各章将贯穿结构程序设计的这些基本思想，并通过实例进行具体分析、讲解，以便更好地帮助读者熟悉和掌握这种方法．同时也希望读者一开始就用这种观点来学习 PASCAL 语言程序设计，并逐步练习使用这种方法来编程，培养科学、严谨的工作作风．

§1.3 PASCAL 语言程序的结构

PASCAL 语言是七十年代初瑞士苏黎世瑞士工学院 **N.Wirth** 教授提出的．它是由 ALGOL 60 语言衍生而成的，具有丰富的数据类型，语句通俗易懂，功能强，特别是它的语言成分是按结构程序设计原则而设计的．因而是推动结构程序设计的一种极好的语言，所以受到人们的好评和日益广泛的重视．它是目前使用最普遍的一种高级语言，适用于教学、管理、科学计算及编写各种系统软件．PASCAL 语言的版本很多，本书仅介绍标准 PASCAL(国际标准化组织 ISO 公布的)，常用的几种机型对标准 PASCAL 的扩充，收集在本书的附录中，读者可以查阅．

与自然语言相似，程序设计语言也有它自己的语法和语义．语法指的是语言的规则和结构，其实质是人和机器的一种协议，不遵此协议人机无法通讯．语义指的是语言的实际意义，它或是指明某种操作(运算)，或是对某种状态进行说明．因此语法和语义是任何一种语言的两个重要

方面. 当然也是计算机语言的两个重要方面. 学习计算机语言必须在理解语义的基础上记忆其语法规则，才能熟练地使用它进行程序设计. PASCAL 语言各成分的语义将在介绍各种成分时逐一解释，它的语法通常用语法图或巴科斯范式来描述. 本书将采用前者.

下面通过一个简单例子先来熟悉一下PASCAL 语言程序的大致结构.

```
PROGRAM addthree(input,output);
VAR
   n1,n2,n3,s:real;
BEGIN
   read(n1,n2,n3);
   writeln(n1,n2,n3);
   s:=n1+n2+n3;
   write('s=',s)
END.
```

这个程序是读入三个数，对它求和，最后把计算结果输出. 这个简单程序反映了 PASCAL 语言程序的概貌.

第一行是程序首部，它是程序的标志. 由 PROGRAM 后接程序名及程序参数表组成. 程序名是程序的标识，由用户自己定义，一般取有意义的具有一定特征的名字. 参数表由圆括号和参数组成，它表示此程序要用到的外部文件.

第二、三行是程序的说明部分，它包括 PASCAL 语言的各类说明及定义，在本程序中只用到了变量说明，它的作用是给编译程序提供必要的信息，开辟变量要用到的内存空间等.

最后是程序的执行部分，程序的执行部分也称为程序体，由一系列语句组成，它们是算法的描述. 一行可以写一个语句，也可写多个语句，两个语句之间用分号隔开，最后一个语句后不加分号. 整个执行部分用 BEGIN 和 END 作为语句括号括起来. 程序的结尾(即 END 后面)用

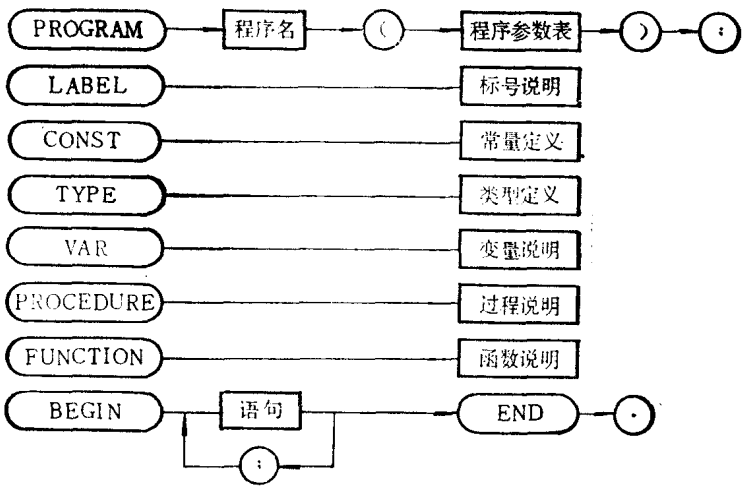

图 1-2 PASCAL 语言程序结构语法图

一个圆点作为结束.

任何一个 PASCAL 语言程序都是由这三部分组成,三部分的顺序不允许变更.

PASCAL 语言程序结构的语法如图 1-2 所示.

其中六类说明的顺序不允许搞错,过程和函数说明可以互相颠倒,它们统称为说明部分.

在程序中可以设置注释部分,用花括号括起来. 注释起备忘和说明的作用,其内容由用户自定,可以是程序名称、功能解释、编制日期等. 注释可放在程序的任何地方,编译对它不起作用. 当然也可以不要它,但在复杂的程序中加注释,将大大提高程序的可读性.

§1.4 PASCAL 语言的基本语法单位

基本语法单位是用来描述语言成分的. 它们由基本字符组成. 在 PASCAL 语言中,允许的最基本符号有三类,即英文字母、阿拉伯数字及一些运算符和标点符号. 它们都是 ASCII 码的子集(ASCII 码见附录四). 这三类符号就是 PASCAL 语言的基本字符集. 任何语言都是由基本语法单位所组成的,PASCAL 语言的基本语法单位有以下几种.

一、保留字

保留字是用来描述 PASCAL 语言的重要语言成分的,是 PASCAL 语言的专有名词,具有固定意义,共有 35 个,如下所示:

AND	END	NOT	THEN
ARRAY	FILE	OF	TO
BEGIN	FOR	OR	TYPE
CASE	FUNCTION	PACKED	UNTIL
CONST	GOTO	PROCEDURE	VAR
DIV	IF	PROGRAM	WHILE
DO	IN	RECORD	WITH
DOWNTO	LABEL	REPEAT	NIL
ELSE	MOD	SET	

为了使程序便于阅读,建议保留字用大写,但对程序的运行,大写和小写都是一样的.

二、标识符

标识符是以字母开头的字母与数字的序列. 标识符的语法图如图 1-3 所示.

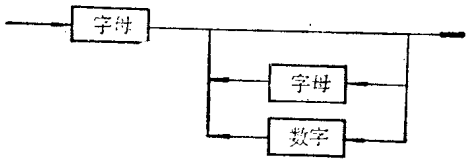

图 1-3 标识符的语法图

标识符有以下两种:

(1) 标准标识符

标准标识符是用来标识(命名)标准函数、标准过程、标准文件以及标准类型和常数的. 实质

上它们主要是一些内部功能模块，由 PASCAL 语言的编译系统直接向用户提供，用户在自己的程序中写上相应的标准标识符就可调用它们，使用起来非常方便．现将它们分类列出如下：

常　　数：**false, true, maxint**

标准类型：**integer, boolean, real, char, text**

标准文件：**input, output**

标准函数：**abs, arctan, chr, cos, eof, eoln, exp, ln, odd, ord, pred, round, sin, sqr, sqrt, succ, trunc**

标准过程：**dispose, get, new, pack, page, put, read, readln, reset, rewrite, unpack, write, writeln**

（2）自定义标识符

自定义标识符是用户在自己的程序中用于标识常数、变量、数据类型、过程、函数及程序的．用户可以任意选用，但应符合如上所说的标识符的定义，且应注意以下几点：

1. 绝对不允许与保留字同名，否则程序运行时一定出错，甚至编译都不能通过．

2. 尽量避免与标准标识符同名，因为标准标识符都对应着特定功能的模块．

3. 必须先定义后使用，即自定义标识符必须在程序的说明部分中经过说明后，才能在语句部分中使用．

4. 标准 PASCAL 语言规定，编译程序只识别标识符的前 8 个字符，若两个标识符的前 8 个字符完全相同，则被认为是同一个标识符．在标识符的字符串序列中，不允许出现空格或其它非字母非数字的字符．

下面是一些正确的标识符：

x, x3, sumthree, convocation

下面是一些不正确（非法）的标识符：

5x	不是以字母开头，与定义不符
VAR	与保留字同名
sum-three	出现非字母和非数字的字符
sum three	出现空格

此外，为使程序清晰易读、便于记忆，建议选用有相应含义的英语单词作为标识符，且一律用小写．

三、运算符及标点符号

这些符号共 24 个列出如下：

+ − * / := ↑ = <> <= >= < > ( ) [ ] { } : , .. ; '

运算符用于各类表达式中．标点符号起辅助的语法作用，如分隔语言成分、代表程序结束等．它的作用也很重要，使用不当程序也会出错，所以同样是必不可少的基本语法单位．