第一章　计算机系统初步和程序设计简介

§1.1　引　　言

本章简要介绍计算机系统和程序设计基本知识，涉及到计算机的组成、计算机语言、PASCAL 语言介绍等内容，以期使读者对计算机系统和程序设计有一个初步的了解。

我们所说的计算机或电子计算机，其全称应是电子数字计算机，是一种能够自动、高速、精确地完成各种复杂数值计算的电子设备，被广泛地应用于科学研究和工程设计方面；同时也是一种能存储大量数据信息并可对其进行快速处理的电子设备，因而在办公室自动化、信息传输、情报检索、企业管理等方面有着广泛的应用；最后它还是一种在自动控制方面大显身手的电子设备，它代替人们对生产过程进行监测和控制，提高了产品质量，减轻了劳动强度，还可以在不适于人类工作的环境下工作，从而保护了人类健康。

电子计算机的发明，是 20 世纪最重大的科学成就之一，现在它正以前所未有的深度和广度影响着人们的生活，改变着我们周围的世界。可以毫不夸张地说，计算机是人类社会实现由工业化到信息化转变的强大武器。

计算机的工作过程就是执行程序的过程。所谓程序就是用某种语言编写的精确定义的指令序列，用以指挥计算机的操作。最基本的程序设计语言是机器语言，或称机器指令系统，由于机器语言难以掌握，不便移植，因而形成了制约计算机普及和发展的瓶颈。为了去掉这个瓶颈，人们设计出从低级到高级的许多种计算

机语言，为计算机在各行各业中的应用与自身发展铺平了道路。本书所介绍的 PASCAL 语言是一种高级程序设计语言，它在教学、科研及软件编制方面都有着重要应用。

§1.2 计算机系统

世界上第一台电子数字计算机是 1946 年 2 月在美国宾西法尼亚大学诞生的，被称为 ENIAC，是英文 Electronic Numerical Integrator And Calculator 的缩写，意为电子数值积分计算机。这台计算机共用了 18 000 只电子管，占地 170 平方米，重达 30 吨,耗电 140 千瓦，每秒可进行 5 000 次加减运算。与现代计算机相比较,ENIAC 除了显得庞大、笨重和高能耗外，还有如下一些致命弱点：它无法存储程序，在计算题目时需要事先根据计算步骤用很长的时间接好外部连线，连线的时间往往比计算的时间还长；其次，由于使用的电子管太多，很容易出现故障。尽管如此，人们还是把 ENIAC 称为世界上第一台电子计算机。

为了解决 ENIAC 所暴露出来的问题， 1946 年 6 月著名的美籍匈牙利科学家冯·诺伊曼(Von Neumann)提出了关于计算机组成和工作方式的基本设想：

一、计算机应包括运算器、控制器、存储器、输入和输出设备五大基本部件，各基本部件功能如下：

(1) 运算器能高速、准确地进行加、减、乘、除等基本算术运算和‘与’‘或’‘非’等基本逻辑运算。

(2) 控制器能自动分析来自计算机其他部件的信息（如来自存储器内某一程序的一条条指令等），并发出相应的控制信号，以指挥计算机各部件协调工作。

(3) 存储器不仅用来存放数据，而且也能用来存放指令。

(4) 操作人员可以通过输入、输出设备与计算机交换信息。

二、计算机内部采用二进制来表示数据和指令，每条指令一

般都具有一个操作码和一个地址码，其中操作码表示运算性质，地址码则指出操作数在存储器上的位置。

三、将编写好的程序和原始数据送入主存储器中并启动计算机工作，计算机不需工作人员干预就能自动地从存储器中逐条地取出指令并执行之。

冯·诺伊曼的这些设想，不仅在当时第一台存储式计算机 EDVAC(Electronic Discrete Variable Automatic Computer)上得到了完全实现，即便在计算机产业飞速发展的今天，计算机系统的基本架构，依然没有超出冯·诺伊曼的设想。

从冯·诺伊曼的设想中，我们看到，计算机系统由两部分组成：一部分是构成计算机的物理装置，如运算器、控制器、存储器、输入输出设备等，这些都是看得见摸得着的实实在在的有形实体，称之为硬件。它们是计算机进行工作的物质基础，计算机性能（如运算速度、精度、存储容量、可靠性等）在很大程度上取决于硬件的配置。另一部分是计算机为达到某种特定目的而逐条执行的一系列指令——程序及运算该程序所需要的数据和有关的文档资料，这些便是计算机软件。如果将计算机硬件比做汽车的话，则计算机软件便是如何驾驶汽车的技术，由此便知，软件是计算机系统中必不可少的组成部分，没有软件，计算机无法工作。

软件内容丰富，种类繁多，通常根据软件种类将其分为两大类：系统软件和应用软件。系统软件是指管理、监控和维护计算机系统正常工作的程序和有关资料，主要包括操作系统、各种语言处理程序(如 PASCAL 编译程序）和一些服务性程序（如机器调试、故障检查及诊断程序等）。在系统软件中，操作系统处于核心位置，它直接和硬件接触，管理和控制硬件资源，属于最底层的软件，又为上层软件提供支持，还为用户提供一个友好的界面，让用户简单、方便、高效地使用计算机，而不必过问和了解计算机硬件的具体细节。个人电脑常用的操作系统有 DOS 和 Windows。应用软件则是为解决某个实际问题而编制的程序和有关资料，包括各种应

用软件和用户程序。

一个完整的计算机系统可用图 1.1 来描述。

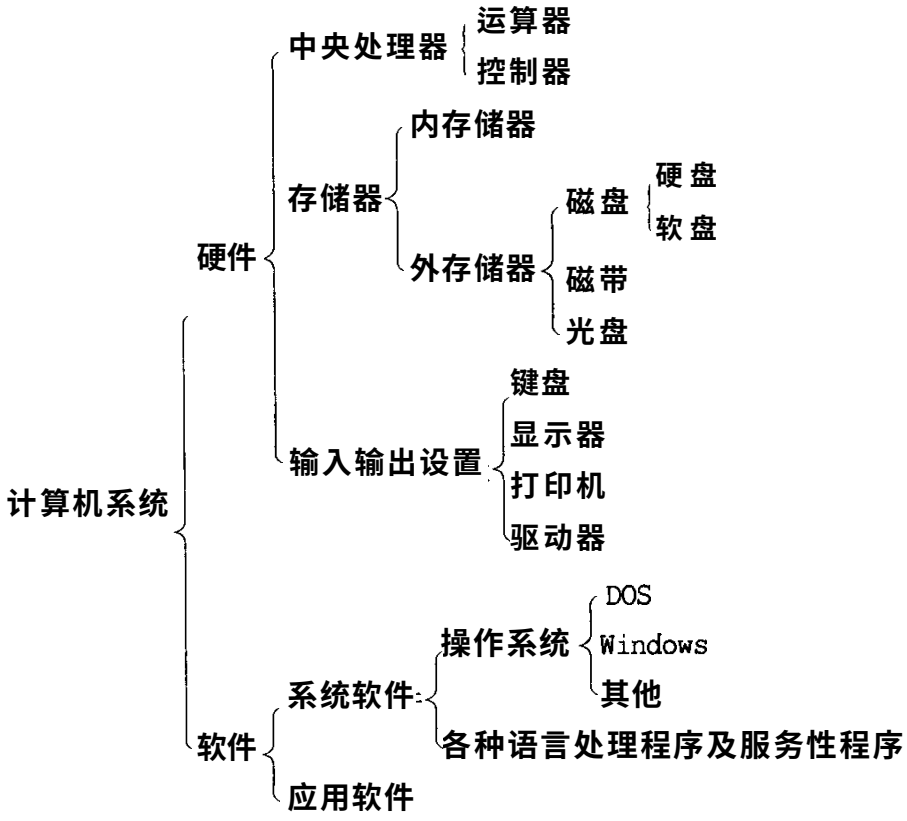

图 1.1　计算机系统

如上所述，完整的计算机系统包括软件和硬件两部分。软件又可分为很多种类，那么从宏观上了解计算机系统内部的相互关系便很有必要,图 1.2 直观地表明计算机系统的层次结构：

从图 1.2 可以看出，最内层是硬件（俗称裸机），与裸机直接接触的是操作系统，这说明操作系统是直接控制和管理硬件的软件，而操作系统外层则为其他软件，最外层是用户程序。各层之间的关系是：内层是外层的支持环境，而外层则可不必了解内层细节，只须根据约定学会使用就行。由此可知，操作系统向下控制硬件，向上支持其他软件，也就是说所有其他软件都必须在操作系统支持下才能运行。这样操作系统最终将用户与计算机实体隔开了，使得用户对计算机的操作一律化为对操作系统的调用，因而一

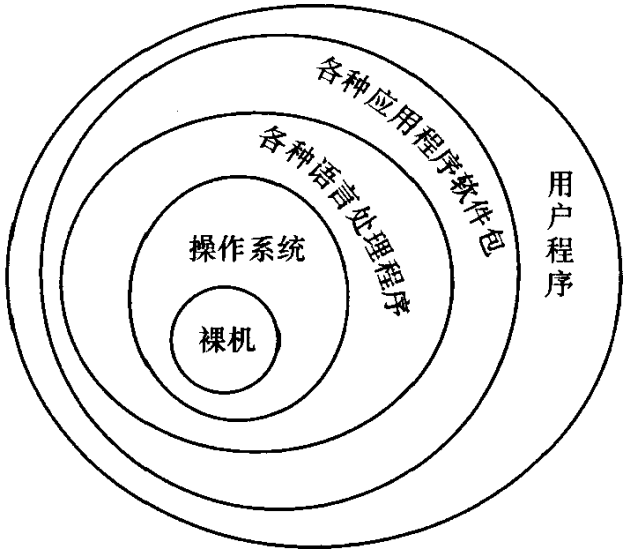

图 1.2

般用户不必熟悉计算机的硬件系统，只要学会使用操作系统就行了，操作系统成了用户与计算机的接口，这就为软件的开发、扩充和使用提供了极大的方便。

§1.3 算　　法

.3.1 简单算法举例

为解决一个问题而采取的方法和步骤就称为算法，也可以说，算法是解题方法的精确描述。

例 1.1　两个瓶子分别盛满酱油与醋，现欲将两瓶所盛的调换，试写出算法。

根据生活常识，欲调换两瓶内盛的酱油和醋，须要有一个空瓶子作‘ 中介 ”,于是有如下算法。

设装酱油的瓶子为 A，装醋的瓶子为 B,空瓶为 C,“A→B ’表示将 A 瓶中的液体倒入 B 瓶，则具体解决这个问题的具体算法如下：

$A \rightarrow C$;

$B \rightarrow A$:

$C \rightarrow B$;

通过以上三步即完成了酱油与醋的互换。

例 1.2 求 $\sum_{i=1}^{10} x_i$,即求和 $x_1 + x_2 + \cdots + x_{10}$。

我们将用 sum 来表示待求之和，因此只要将诸 x_i 逐次加到 sum 上去就可以了，于是有下列算法。

① $0 \rightarrow sum, 1 \rightarrow i$;

② 给出 x_i;

③ $sum + x_i \rightarrow sum$;

④ $i + 1 \rightarrow i$;

⑤ 如 $i \leqslant 10$ 则转②，否则即停止。

显然，经过以上五步计算，即可求出 $\sum_{i=1}^{10} x_i$,且将和数放在 sum 中。

例 1.3 求 $\prod_{i=1}^{10} x_i$,即求积 $x_1 \cdot x_2 \cdots x_{10}$。

我们用 P 表示待求之积，因此，只要将诸 x_i 逐次与 P 相乘即可，于是写出下列算法。

① $1 \rightarrow P, 1 \rightarrow i$;

② 给出 x_i;

③ $P \cdot x_i \rightarrow P$;

④ $i + 1 \rightarrow i$;

⑤ 如 $i \leqslant 10$ 则转②，否则即停止。

为了让计算机按人们指定的步骤有效地工作，必须事先编制好一组让计算机逐次执行的指令，这就是程序。如何来进行程序设计呢？首先要对求解的问题进行分析，确定解题的方法和步骤，即确定算法，如上面几例中所介绍的算法便是如此，然后再根据相

应的算法，编写出程序并交计算机求解。

由上面几例可以看出，算法应该在有限步内实现，算法的每一步含义都是明确的，不能含有歧义，并能被有效地执行。所谓能被有效地执行，理所当然排除了像用零作除数之类的运算，所谓不能含有歧义，是指算法的每一步都只能有唯一的一种情形被执行，不允许有二义性。因此，诸如“李明让张方把他的字典拿来”这样的语句是不能写进算法的，因为从字面上判断不出张方该把谁的字典拿来。为了加强人机对话，在一个算法内应该允许有一个或多个输入输出。

1.3.2 算法的表示——流程图

在 1.3.1 里，我们使用自然语言描述了几个具体算法，这种表达方式有其好处：通俗易懂，但也有其缺点。有时为清楚说明算法的某一步，须要做过多的解释，特别是在解释判断、转移时，往往不够清晰，而且还会产生歧义性。这种歧义性对身临其境的人来说，可通过内心的判断加以消除，但对计算机来说便会无所适从。鉴于用自然语言表述算法的种种弊端，人们用流程图来表述算法，即用若干种几何图形框来表示不同性质的操作，ANSI(美国国家标准化协会）规定的一些常用的流程图符号（见图 1.3)已被大多数国家所接受。

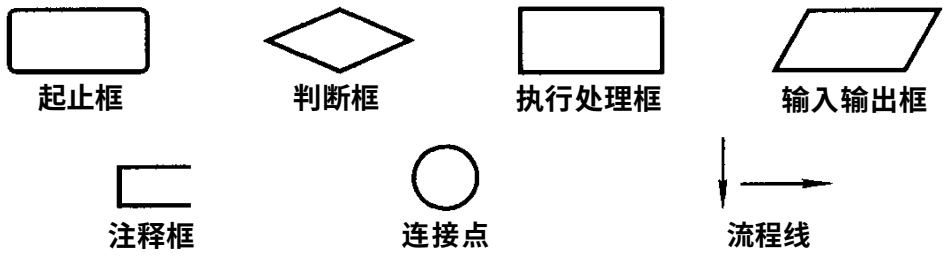

图 1.3

现在我们用流程图来表示 1.3.1 中介绍的两个算法。

例 1.1　A 表示盛酱油的瓶子，B 表示盛醋的瓶子，C 表示空瓶，记号 A→C 表示将 A 中所装的液体倒入 C 中，实现 A，B 互换的算法可用图 1.4 中的左图表示。

例 1.2　求 10 个数的和，流程图如图 1.4 中的右图所示。

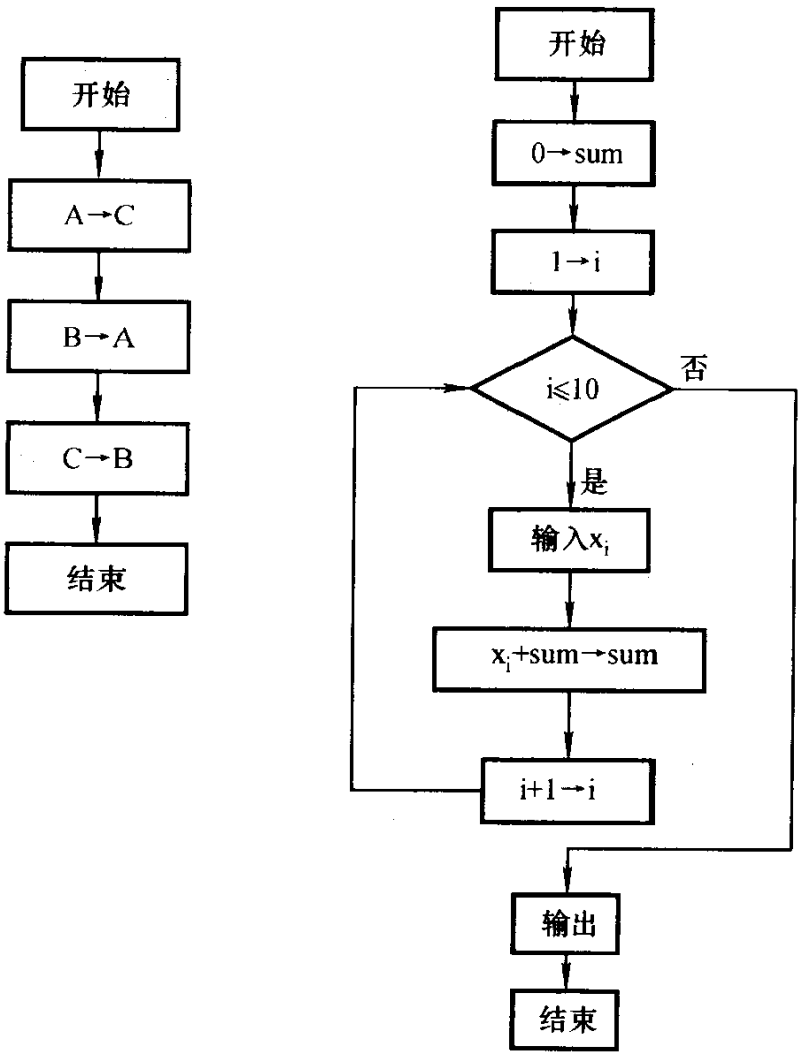

图 1.4

3.3　程序设计的三种基本结构及其流程图表示

由前一小节可知，对于给定的问题，依据选定的算法，画出相应的流程图，便可据此编写计算机程序了。但是，如果允许流程图

中的流程线可随意指向任何一个框，就可能使画的流程图失去直观易懂的优点而成为令人难以理解的“一团乱麻”。因此，必须限制滥用转向箭头，为此人们规定三种基本结构，就像搭积木一样，用这三种基本结构按一定规律就可以组成一个算法，现将这三种基本结构介绍如下：

一、顺序结构

在此种结构中，各框是依据顺序执行的，顺序结构是最简单的一种结构，图 1.5 所示的便是仅含两个执行框的顺序结构。

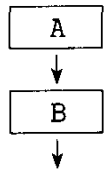

图 1.5

二、选择结构

在此种结构中有一个判断框，它有两个分支，依据条件是否满足而分别执行相应的分支，图 1.6 所示的便是仅含有一个判断框的二分支结构，依据 P 是否满足而分别执行 A 或 B。

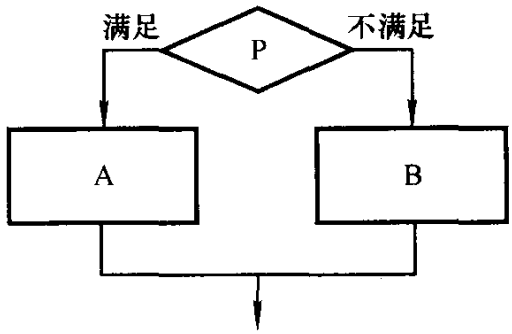

图 1.6

三、循环结构

所谓循环结构，即指重复完成相同性质的运算或操作的一种算法结构，共有两类循环结构。

(1) 当型循环　如图 1.7 所示，当条件 P 满足时，反复执行 A 框，一旦条件 P 不满足便结束此循环而执行其下面的一个基本结构。显然，若一开始条件 P 不被满足，则根本就不执行 A 框，但若一开始条件 P 被满足，则 A 框便被执行，在 A 框中必有相应的操作，其作用在于改变条件 P，使其由满足变为不满足，从而在有限次地执行 A 框后结束本循环而转去执行下面一基本结构。

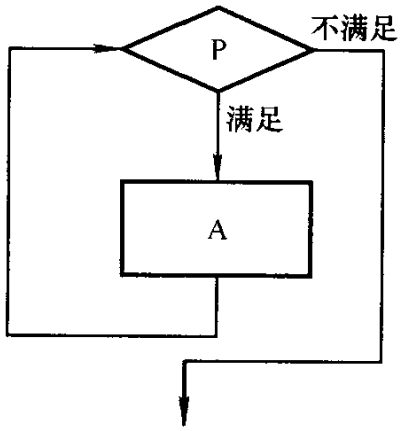

图 1.7

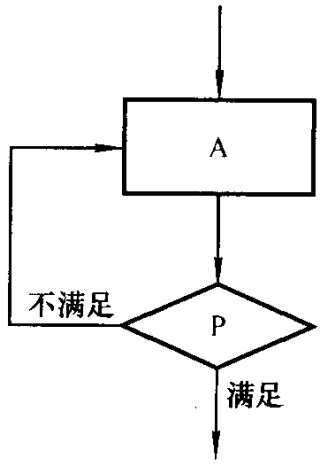

图 1.8

(2) 直到型循环　如图 1.8 所示，先执行 A 框，然后判断条件 P 是否被满足，如条件 P 不被满足，则反复执行 A 框，直到条件 P 被满足时才结束本循环，接着执行其下面的一个基本结构，与当型循环的说明类似，为使此种循环不被无限次地执行，在 A 框内应有相应的操作，以使 P 由不满足变为满足。

以上我们介绍了程序设计的三种基本结构，可以证明，由三种基本结构组成的算法可以解决任何复杂问题，由基本结构所构成的算法称为结构化算法，依据结构化算法进行程序设计称为结构化程序设计。

由于计算机在运算速度和存储量方面已取得了长足的进步，因此现在衡量程序质量的标准已由在保证正确性的前提下刻意追求运算量和存储量的节省，转向程序的可读性或清晰性为第一要

素。同时程序的编写方式也由一人一户的小作坊式生产转向大产业化生产,因此要胸有全局、通盘考虑、逐步加细、分工合作、模块化生产。于是结构化程序设计的基本要求是：

(1) 基于自顶向下、逐步细化的设计方法 —— 将一个程序分成若干个模块，每个模块具有独立的功能，模块之间的联系要尽可能简单，每个模块只有一个入口和一个出口。模块内部均由顺序、选择和循环三种基本结构组成，实现程序编制模块化的具体方法是子程序。

(2) 程序要整齐、清晰，便于阅读和维护，为此须逐步养成良好的程序设计风格。

§1.4 计算机语言

想利用计算机来解决问题，必须让计算机“明白”你要它作什么和怎样做，为此必须解决一个“语言”沟通问题，如§1.2所述，计算机只能接受和执行二进制指令，换言之，为每台计算机设置的指令系统便是该种型号的计算机语言。那么，人要和计算机打交道，该使用什么语言才能让计算机服服帖帖地为人们工作呢？

1.4.1 机器语言

计算机只能接受和执行为其设置的二进制指令，所有这些指令的集合，便是计算机的机器语言指令系统，简称为机器语言。为解决一个特定的问题，在确定具体算法之后，选用指令系统中的相应指令按顺序写出就组成一个“程序”。

机器语言实际上是一条条二进制指令，这就决定了这种语言难学、难记、难懂、难修改,而且用这种语言写程序单调、乏味、效率低，又加之不同型号的机器有自己的指令系统，因而在甲种机器上用机器语言编写的程序一般不能在乙种机器上使用，要想使用乙种机器，必须再学乙种机器的机器语言并用其重新编写程序，程序

的可移植性极差。

由于以上所述的种种原因，使用机器语言编写程序已成为计算机发展和普及的瓶颈。

1.4.2 汇编语言

汇编语言是由机器语言发展演变来的，它使用一些“助记符号”来代替那些难懂难记的二进制指令，如用 ADD A B 表示将 A 与 B 相加，显然，这种语言要比机器语言易于理解，便于记忆，容易使用，也不难看出，汇编语言的指令和机器语言的指令基本上是一对一的，即一条汇编语言指令代替一条机器语言指令。

汇编语言确实将人们从单调乏味、效率低、难学、难记、难修改、不便于移植等用机器语言编写程序的困难境地中解放出来，但计算机并不懂汇编语言，因此也就不能执行用汇编语言写的程序，为此必须将用汇编语言写的程序翻译成用机器指令写的程序。这种翻译工作是由一种专门的汇编程序来完成的。我们将用汇编语言（或其他高级语言）写的程序称之为源程序，而将经过“翻译”后得到的可由计算机直接执行的机器语言程序称之为目标程序。

汇编语言和机器语言一样，都是针对具体计算机系统而设定的，计算机的型号不同，所用的汇编语言也不同，这就是为什么我们称机器语言和汇编语言为“面向机器的语言”的原因，它们也被称为“低级语言”。在用汇编语言写程序时，要求对计算机的硬件要有基本了解。

1.4.3 算法语言

人们期望能用一种接近于自然语言或数学语言的专用语言来表示算法，这种想法导致了 FORTRAN, ALGOL 60 等算法语言的相继问世，算法语言与具体的计算机无关，即用它所写的程序可以在任何一种计算机上运行（必要时只须做一些小小的修改）。这种语言称为面向过程的语言，只需根据所求解问题的算法，写出处理的过

程即可，而不必涉及计算机的内部结构。一个算法语言程序由许多语句组成，一个语句可以对应多条机器指令，所以用算法语言写程序自然、简单、方便、直观、不易出错。但是计算机不能直接执行算法语言程序，而必须先翻译转换成“目标程序”才能执行。每种算法语言都有自己的翻译转换程序——编译程序，一个计算机能否运行用某种算法语言写的源程序，关键是该计算机是否配备了这种算法语言的编译程序。

由于学习用算法语言编程序比学习使用机器语言和汇编语言编程序容易得多，算法语言的出现使各行各业的人们都能十分方便地使用计算机，为计算机的普及推广应用打开了方便之门。

PASCAL 语言是算法语言，算法语言是高级语言。

计算机硬件在日新月异地发展着，计算机软件技术也正以前所未有的速度前进。今天，人们已不满足于编写程序告诉计算机做什么和怎么做——这正是算法语言所胜任的工作。人们期望提高应用系统开发的速度，最大限度地降低应用系统调试工作量，而且能使其升级换代变得既容易又快捷，仅从需求的高级描述中便可产生无差错代码，一句话，人们只要指出“做什么”，由计算机自己去解决“如何做”，这就是非过程化语言。

显然非过程化语言比算法语言功能更强、更进一步，人们还希望计算机能像人一样进行逻辑推理、应用知识、直至会学习，这便是正在研制的人工智能语言。

目前国内外大多数计算机上运行的程序多是用算法语言编写的。

§1.5 PASCAL 语言介绍

1.5.1 PASCAL 语言的特点

PASCAL 语言是瑞士苏黎世联邦工业大学的 Niklaus Wirth (沃思)于 1968 年设计完成的,1971 年正式发表，是目前世界上使用最广的程序设计语言之一。

1975 年，在对 PASCAL 语言介绍进行修改的基础上，“标准 PASCAL 语言”被推出，国际标准化组织 ISO(International Standard Organization)在对“标准 PASCAL 语言”作了一些修改之后，已将其定为 ISO 标准 PASCAL 语言。我国制定的 PASCAL 标准与 ISO 标准一致，不同的计算机系统在实现 ISO 标准时都作了不同程度的增删。本书则以 ISO 标准和我国的标准为准。

PASCAL 语言具有如下特点：

(1) 它是结构化语言，提供了直接实现三种基本结构（顺序、选择、循环）的语句，可方便地书写结构化程序。在结构化这一点上,PASCAL 比 FORTRAN 更好一些，因而用 PASCAL 语言写出的程序，易于保证正确性和可读性。

(2) PASCAL 语言有丰富的数据类型，因而可以很方便地用来描述复杂的算法，广泛地应用于数值计算和非数值计算领域，且 PASCAL 语言程序和 C 语言程序之间的转换十分容易。

(3) 用 PASCAL 语言编写程序，书写格式远较 FORTRAN 自由，编译出的目标程序质量高，运行效率好。虽然 C 语言比 PASCAL 更灵活,但 PASCAL 更为严谨。

(4) PASCAL 语言的结构化程度好于 C，易于培养结构化编程的观念。

正是由于以上诸特点，使 PASCAL 语言不仅得到了广泛的实际应用，而且成为很多学校首选的程序设计课程中的主要语言。

1.5.2 PASCAL语言的基本符号

PASCAL语言只允许使用以下几类基本符号：

(1) 大、小写英文字母 A～Z a～z

(2)数 字 0～9

(3) 其他符号 + - * / = < > < <= > >= () [ ] { } , ; . : .. ' ↑ :=

(4)保留字

保留字（或称关键字）是 PASCAL 语言选定的具有固定意义和用法的专用英文单字或缩写，程序中不允许做规定意义之外的使用，使用时无须区分大小写。标准 PASCAL 的保留字有：

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

(5) 标准标识符（预定义标准标识符）

对于各类程序都经常要引用的处理对象，PASCAL语言为了减少程序中的重复说明，规定了一些标准标识符，它们都有约定的意义，可以在程序中直接使用而不必加以说明，使用时无须区分大小写。常见的标准标识符有：

标准常量：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 In Odd Ord Pred Round Sin Sqr Sqrt SuccTrunc

标准过程：Dispose Get New Pack Page Put Read Readln Reset Rewrite Unpack Write

Writeln

随着不同版本的 PASCAL 编译软件的发行，保留字和标准标识符会有少许差别，编程时可参考具体的使用手册。

（6）标识符

用来标识符号常量、变量、类型、函数、过程、文件等的有效字符序列称为标识符，简单言之，标识符就是一个名字。 PASCAL 语言规定标识符是以英文字母开头的字母、数字的组合，且不区分大小写英文字母，如 student、STUDENT、sTuDent 是同一个标识符。标识符的长度（即其所含字符的个数）是没有限制的，但标准 PASCAL 规定能区分的有效标识符的长度是 8 个字符，虽然超过 8 个字符的标识符也能使用，但以前 8 个字符为有效字符，因此，当两个标识符的前 8 个字符相同时，即认为这两个标识符相同，如 Myclassmates 和 Myclassmate 被认为是两个相同的标识符，因为它们的前 8 个字符相同，但 MS PASCAL 允许标识符的长度为 31。

标识符必须严格遵照“先定义后使用”的原则，即一个用户定义的标识符必须先出现在程序的说明部分，然后才能出现在程序的语句部分（唯一的例外之处在指针说明，请看有关章节）。在定义标识符时要严格按照标识符的定义规则，所定义的标识符不能与保留字相同，尽管语法上允许用户定义的标识符与预定义标识符相同，但一旦用户这样定义了标识符，原标识符就失去了其特有的定义,如 sin 原本表示某一角度（弧度制）的正弦函数，倘若用户重新定义其为变量名,则在该用户的程序内，sin 就不再代表正弦函数而代表被定义的变量名，为了使程序清晰易读，自定义的标识符最好不要和标准标识符相同。

为了能够区分关键字、标准标识符和用户自定义标识符，本书全部用大写字母表示关键字，将标准标识符的第一个字母用大写来表示，而全部用小写字母来表示用户自定义标识符。读者自己编写程序时，可以随意地用任何一种方式表示，因为 PASCAL 系统并不用大小写来区分标识符。 PASCAL 语言对用户自定义标识符

中所用的字符没有过多的规定，但为了程序便于阅读起见，建议读者编程序时选用见名知义的标识符，如用 volume 表示体积，用 area 表示面积,用 sum 表示和，用 pi 表示圆周率等等。

(7)分隔符

用来隔开程序语言中符号、标识符等的符号称为分隔符，在 PASCAL 语言中，空格、注解和行结束符都可以作为分隔符，有不少 PASCAL 语言符号，本身既具有明确定义，同时又有分隔相邻符号的双重功能，如分号、括号等。一般说来，在 PASCAL 语言程序中，任何保留字、标识符（包括标准标识符）、数、标号之内不得插入分隔符(如 BEGIN 不得写为 BE GIN 否则将被视为两个标识符 BE 和 GIN 而出错）。同样，任何相邻的保留字、标识符、数、标号之间至少要有一个分隔符，例如 PROGRAM hd 不能写成 PROGRAMhd。

5.3 PASCAL 语言程序结构与简单程序举例

PASCAL 语言程序的结构是有严格规定的，为了说明这点，我们先来看一个简单的例子。

例 1.3 已知球的半径，求球的体积和表面积。

设球的半径为 r,表面积为 area,体积为 volume ,则依据数学公式有：$area = 4\pi r^2$, $volume = \frac{4}{3}\pi r^3$, 其中 π 为圆周率，由于在 PASCAL 语言程序中不允许使用希腊、拉丁等字母，于是在程序内我们改由 pi 表示 π。计算机所需初始数据可由读语句从键盘输入，计算结果可通过写语句输出到显示屏或打印机上。

```
PROGRAM ball( Input, Output);
{已知球的半径求它的表面积和体积}
  CONST
    pi = 3.14159;
  VAR
    r, area, volume: Real;
```