PASCAL语言程序设计

陆 倜等 编著

航空工业出版社

PASCAL 语言程序设计

陆 倜等 编著
林定基 审订

航空工业出版社

1995

（京）新登字 161 号

内 容 提 要

PASCAL 语言是国内外广泛使用的一种结构化程序设计语言，常用于编写系统软件、应用软件及进行科学计算。本书从基本概念出发，循序渐进地介绍了标准 PASCAL 语言的数据类型和数据结构以及 PASCAL 语言程序设计方法。在介绍 PASCAL 语言的枚举、子界、集合、记录、数组、文件、栈和队列、指针和动态数据结构的同时，结合基本原理给出大量程序实例。书中所有例题都上机调试通过。书中还设专章介绍 Turbo PASCAL 对于标准 PASCAL 的扩充及上机操作方法。为方便读者上机，每章后都附有习题。

本书既可作为大专院校非计算机专业计算机基础教学的教材，也可以作为有关专业技术人员的参考书。

图书在版编目(CIP)数据

PASCAL 语言程序设计/陆倜等编著. —北京：航空工业出版社，1995.1

ISBN 7-80046-854-2

Ⅰ.P… Ⅱ.陆… Ⅲ.PASCAL 语言-程序设计 Ⅳ.TP312PA

中国版本图书馆 CIP 数据核字（94）第 10208 号

JS25-64

航空工业出版社出版发行

（北京市安定门外小关东里 14 号 100029）

北京市通县向阳印刷厂印刷 全国各地新华书店经售

1995 年 1 月第 1 版 1995 年 1 月第 1 次印刷

开本：787×1092 1/16 印张：23.5 字数：584 千字

印数：1—4000 定价：22.50 元

前　言

Pascal 语言是 70 年代初由瑞士的 N. Wirth 教授提出的，是目前使用最广泛的几种计算机高级程序设计语言之一。Pascal 语言优点很多，应用广泛，很适合于教学、科学计算以及编写各种应用软件和系统软件。

Pascal 语言数据类型丰富、描述数值问题和非数值问题灵活、严谨，用 Pascal 编写的程序有良好的模块化结构，是由独立的子程序组成的程序，可以充分体现结构化程序设计思想，有益于培养学生运用结构化程序设计方法来开发程序的能力，有益于学生建立良好的程序设计风格和素质。又由于它结构清晰、算法描述简明，便于学习，因此建议在学习高级语言时，以先学 Pascal 为宜，这样可以养成良好的基本素质，再去学习像 Fortran、Prolog、Foxbase 等专业性较强的语言。目前 Pascal 语言已广泛地配置在微型机和各种计算机上，为其推广和普及提供了环境基础。基于这些情况，我们为初学者编写了本书，作为高等学校非计算机专业计算机基础课的教材。

本书全面介绍了 Pascal 语言，而且通过对 Pascal 语言的讲述，介绍了自顶向下逐步求精的结构化程序设计的基本思想和基本方法。本书还给出了大量例题和习题，这些题不但考虑到由易到难，而且类型多样，以帮助学生们逐步建立使用计算机程序设计语言的丰富思路。

本书主要介绍标准 Pascal 语言，但也简要地介绍了 Turbo Pascal 的特点和主要内容。

本书的主要编写人有陆倜（主编）、刘梅彦（副主编）、李桂芝、赵玉双等。

本书由林定基教授审阅，他提出的许多宝贵意见和指导性思路，使我们受益匪浅，在此深表感谢。

编者

1994. 10.

目　录

第1章 概 述

1.1 计算机系统

通常所说的计算机是指电子数字计算机。它是一种能自动、高速地进行数值计算、信息处理、自动控制等方面工作的电子设备。

计算机自1946年问世以来，发展十分迅速，短短的四十几年，已经经历了电子管计算机、晶体管计算机、集成电路计算机和大规模集成电路计算机等四代，现在正进入第五代计算机的研制。随着计算机技术的发展，计算机的应用也越来越广泛。目前，计算机技术主要用于以下几个方面：①科学计算；②数据处理或称信息处理；③生产过程自动控制；④计算机辅助系统；⑤人工智能；⑥消费领域。

计算机具有以下特点：①运算速度快；②计算精度高；③具有记忆和逻辑判断功能；④能自动连续地高速运算而不需要人的干预。

计算机是由电子元器件组成的机器，它之所以能脱离人的直接干预而自动地进行计算，是由于人们把需要计算机做的工作写成一定形式的指令。若干条指令构成一个程序，它规定了为解决某一特定问题而让计算机执行的一系列的动作。这些指令存储在计算机的存储器中。因此，在给出命令使计算机开始工作以后，计算机完全根据程序的指定，完成一系列的操作。也就是说，计算机系统由两部分组成，一部分是机器本身，另一部分是程序，它们分别称为：硬件系统和软件系统。下面分别介绍计算机硬件系统和软件系统的基本组成及它们的功能。

1.1.1 计算机硬件系统

硬件系统是计算机系统中物理装置的总称，包括电子、磁性、机械、光学等元器件，以及由这样的元器件组成的部件和装置。它是计算机系统的物质基础。硬件系统的基本功能就是能够执行预先设计好的由若干条指令组成的各种程序。

计算机硬件系统的基本组成见图1-1。它包括如下几个部分：

1. 存储器

它是计算机的记忆装置，用来存放程序指令和数据。它分为内存储器和外存储器。

在计算机内部设有一个内存储器，简称内存。内存由半导体元件组成，它的最大优点是存取速度快，体积小，但是它的存储内容断电即失。计算机进行计算以前，程序和数据通过输入设备送入内存，计算开始后，内存不仅要为其他部件提供必要的信息，也要保存运算中间结果及最后结果。

内存由许多存储单元组成，存储单元可以存放指令或数据。为区分不同的存储单元，就把内存中全部存储单元按一定顺序统一编号，把这种编号称为地址编码，简称地址。计算机对内存采用按地址存取的方式。当计算机要把一个数据存入某存储单元或从某存储单元取出时，首先要给出该存储单元的地址，然后按地址查找对应的存储单元，查到后才能进行数据

的存取。要把存储单元的地址和存储单元

的存取。要把存储单元的地址和存储单元
的内容严格区别开来。

从某一存储单元取一个数据后，该存储单元中的数据并不消失，除非向该单元送入一个新的数据后，此单元的内容才改变。

设在计算机以外的存储器称外存储器，简称外存。它存储的内容不会像内存那样断电即失，可以用它来存放需要长久保存但使用不频繁的程序和数据。外存的容量比内存的容量大得多。通常存储在外存上的数据要调入内存才能被计算机使用。它存取信息的速度比内存慢。计算机对外存不是按单个数据进行存取，而是以成批数据进行处理的。

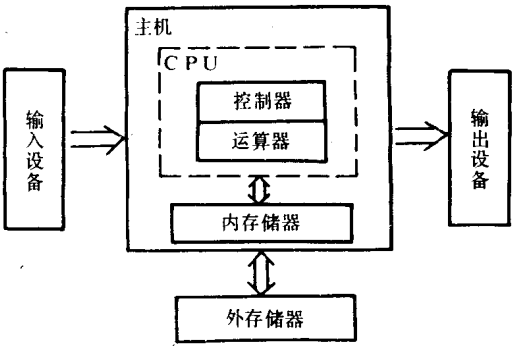

图 1-1

外存种类较多，如磁带、软磁盘、硬磁盘、以及光盘存储器等等。

2. 运算器

它从内存中取出数据，按照控制器发出的运算命令，执行算术运算或逻辑运算，并把运算的结果送回内存。

3. 控制器

它的主要作用是使计算机各部件协调地工作，执行内存中存储的程序。控制器分析从内存中取出的指令，根据指令的功能向各部件发出控制命令，控制它们执行相应的动作。当各部件执行完控制器发出的命令之后，会发出汇报命令执行情况的“反馈”信息。当控制器得知一条指令执行完后，会自动顺序地取下一条要执行的指令。对不同的指令，发出的控制命令是不同的。这些命令是由控制器按一定的时序发出的各种电信号。

在微型计算机系统中，通常把运算器和控制器合起来称为中央处理器（Central Processing Unit），简称 CPU。它是计算机的核心部分。中央处理器、内存储器和输入输出接口组成计算机的主机。

4. 输入输出设备

单是主机仍无法工作，因为程序和数据无法送到计算机内存中存储起来。因此，还必须有输入设备。输入设备的作用是把要输入的程序和数据通过输入接口顺序地送往计算机内存中。常用的输入设备有：键盘、磁盘机、触摸屏、光电输入设备、声音输入设备等。

计算机还需配置输出设备，以便把运算结果送出。主机通过输出接口把数据送至输出设备。常用的输出设备有显示器、打印机、声音输出设备等。磁盘机也可以作为输出设备。

输入设备、输出设备和外存储器都属于计算机外部设备。

1.1.2 计算机软件系统

要使计算机运行起来并解决各种问题，仅有硬件是不够的，还必须给它编制由一条条指令组成的各种程序。随着计算机技术的发展，不仅有用户自己编制的程序，还要有专为计算机

配置的具有特殊功能的各种程序，这些程序通称为软件。一个不包含任何软件的计算机称为裸机。软件系统要解决如何管理机器和使用机器的问题，也就是说怎样通过软件的作用更好地发挥计算机的功能。

计算机软件主要有两大类：系统软件和应用软件。系统软件是计算机厂商在出厂时提供的。常见的有：操作系统、汇编程序、编译程序、控制程序、数据库管理系统等，用户可以使用它但不能随意修改它。应用软件指计算机用户利用计算机的软硬件资源为某一专门应用目的而开发的软件。例如工资管理软件、财务管理软件等。此外还有工具软件，用以方便用户进行软件开发，例如编辑软件（如EDLIN）、绘图软件（如AUTOCAD）等。计算机软件系统的基本组成如图1-2所示，其中外层可以利用内层提供的手段。

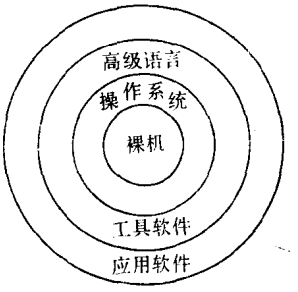

图 1-2

软件中最重要的是操作系统，它是所有软件的核心。操作系统是对计算机系统资源（包括硬件和软件）进行管理和控制的程序，使计算机系统能协调地、高效地对数据进行处理，为用户创造方便、有效和可靠的计算机工作环境。

1.2 计算机语言

1.2.1 机器语言

语言是人们交流思想的工具。计算机语言是人与计算机之间进行对话的工具。要使计算机按人的意图工作，就必须使计算机懂得人的意图，接受人向它发出的命令和信息。

计算机并不能直接理解和执行人们使用的自然语言，而只能接受和执行二进制的指令。每一条二进制的指令对应着一种基本操作，例如某一种型号的计算机以10000000表示“进行一次加法”，以01110000表示“传送一次数据”。计算机能够识别和执行的这种指令称为机器指令。每一种类型的计算机都有它自己的一套指令系统，这是设计计算机时所确定的，是计算机所固有的。

所谓机器语言就是机器指令的集合。机器语言是“面向机器”的。用机器语言编写的程序，就是根据要解决的具体问题选用指令系统中的某些机器指令组成的程序。因为这些由“0”和“1”组成的指令，难学、难懂、难记、难修改，所以用机器语言编写程序十分不方便，只能由计算机专业人员编写。而且每一种机器都有自己特定的机器指令系统，编出来的程序互不通用。例如已编好能用于甲型号机器的机器语言程序，想拿到乙型号机器上去运行，一般是不行的，必须用乙型号机器的指令重新编程。这给计算机的推广使用造成很大障碍。

1.2.2 汇编语言

由于用机器语言编写程序难于阅读，后来出现了“汇编语言”。它用特定的助记符号来代表二进制编码的机器指令。它和机器指令是一一对应的，即用一条汇编语言指令代替一条机

器指令。例如：用 ADD A，B 代表将 A 寄存器中的数与 B 寄存器中的数相加，结果放在 A 寄存器中。显然，这种表示形式比机器语言容易理解、容易使用。

用汇编语言编写的程序称为汇编语言源程序。但计算机对它不能直接执行，必须经过汇编程序把它翻译成机器语言程序后才能执行。这个翻译过程就称为汇编。经过转换后得到的可以由计算机执行的机器语言程序称为目标程序。

汇编语言虽然比机器语言前进了一步，但仍比较繁琐，无通用性。汇编语言也是针对特定的计算机系统的，不同类型的计算机所用的汇编语言是不同的。所以称机器语言和汇编语言为面向机器的语言，也称低级语言。用汇编语言编写程序必须了解计算机的内部结构，在存储数据时要具体写出存储单元的地址，对程序编写人员要求比较高。

1.2.3 高级语言

从 50 年代末开始，陆续推出了各种“程序设计语言”即高级语言，如 FORTRAN、COBOL、BASIC、PASCAL 和 C 语言。它是由各种类似自然语言的具有不同意义的“符号”和“表达式”按照一定的语法语义规则组成的。用高级语言编写的程序同人们习惯用的自然语言（英语）、数学公式类似。例如：PASCAL 语言以“write（A＋B）”表示“输出 A＋B 的值”。显然，这种表示方法是很容易理解的。

高级语言的语句功能较强，一个语句往往相当于许多条机器指令，因而使用高级语言编写程序更方便。高级语言是独立于机器的。一般在编写程序时人们不需要对机器的指令系统有深入的了解，而且用高级语言编写的程序可以在不同型号的机器上运行。这种语言称为“面向过程的语言”，只需根据所求解的问题写出处理过程即可，而不涉及计算机内部的结构。

显然，计算机不能直接执行用高级语言编写的程序（称为高级语言源程序），只有经过“编译程序”翻译成机器语言程序后才能执行。这个翻译过程称为编译。如图 1-3 所示。

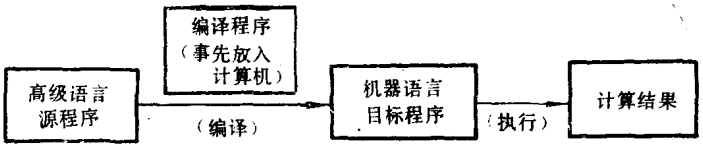

图 1-3

由此可见，用高级语言编写的源程序送入计算机执行时，应分两步进行：第一步是编译，调用编译程序，将源程序翻译成机器语言目标程序，如果编译无错误，就可进行第二步；第二步是运行，运行的程序把接受进来的数据进行处理加工，然后输出结果。

每一种高级语言都有自己的“编译程序”。例如：FORTRAN 语言源程序只有用 FORTRAN 语言编译程序才能把它翻译成机器语言程序；PASCAL 语言源程序只有用 PASCAL 语言编译程序才能把它翻译成机器语言程序。同一种高级语言对于不同种类计算机其编译程序也是不同的，因为每种机器有不同的指令系统。每一种计算机在出厂销售时，都已配置了该型号计算机上使用的各种语言的编译程序。

另外还有一种处理高级语言的方法。如 BASIC 语言解释程序，在执行 BASIC 语言源程序

时，一条源程序由 BASIC 解释程序翻译成机器指令后执行，接着再进行下一条，这样一条一条地边解释边执行。这种方法，程序运行速度比编译方法慢。

1.3 程序设计与算法

1.3.1 程序设计

有人认为计算机是“万能”的，只要把问题告诉计算机，计算机就会自动完成一切，给出结果。其实这是一种误解，要使计算机按照人们确定的方案，执行指定的操作步骤，必须事先编写相应的让计算机执行的程序。程序设计是指从拿到任务分析问题开始，到确定算法、编写程序、上机调试、直到分析结果、整理资料的全过程。见图 1-4。

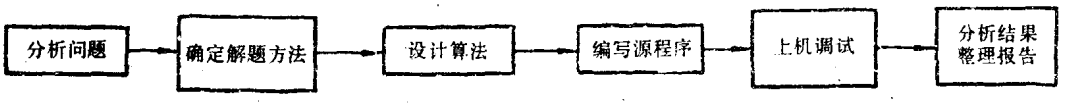

图 1-4

在学习程序设计时，既要掌握所采用的某一种计算机语言，如 PASCAL 语言，更要掌握解题的方法和步骤，即设计算法，这是程序设计的关键。语言只是一个工具，只懂得语言的规则并不保证能编制出有效的、高质量的程序。如果算法是正确的，将它转换为任何一种高级语言程序并不困难。程序设计人员水平的高低在于他们能否设计出好的算法。

1.3.2 算法的概念

人们希望计算机按自己的愿望进行某项计算或者数据处理任务。但怎样才能使计算机按照一定的步骤进行工作，最后达到预定的目的呢？在多数计算机系统中，要求使用者不仅要先告诉计算机“做什么”，还要告诉计算机“怎么做”。例如，让计算机解一个一元二次方程：

$$ax^2+bx+c=0$$

计算机是解不出来的，必须确定用什么方法去解这个方程，以及一步一步怎么解。人们做任何事情都需要事先想好进行的步骤和方法，这就是“算法”。让计算机完成一个确定的任务也必须事先设计好计算机具体操作的步骤，这就是“计算机算 法”，即计算机可以实现的算法。例如让计算机从两个数中挑选出大者并打印出 来。写出如下算法：

①输入两个数 a 和 b。

②判断 a 和 b 谁大。如果 a>b，则把 a 的值送到一个名为 max 的存储单元中，否则（即 a 不大于 b）把 b 的值送到 max 中。

③将 max 的值打印出来。

此算法的每一步都是计算机能够执行的，因为利用计算机语言可以写出与之相应的语句。根据设计出的算法就可用某一种计算机语言编写出相应的程序，让计算机按人们的意图进行各种操作。因此，在编写程序之前应当设计出解该问题的算法。

注意：写出的算法应当具体到它的每一步都是计算机能够执行的。例如上例，如果只写成“输出 a 和 b 中的大者”还不够，因为计算机不知道怎样才能找出 a 和 b 中的大者。因此应对它进一步细化，分解成以上三步之后，才成为计算机能够执行的算法。

1.3.3 算法的特性

在计算机科学里，一个算法应具有以下特点：

(1) 有穷性

一个算法必须在执行有限个操作步骤之后结束，而不能是无限的。例如利用公式

$$\pi=4(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots)$$

求 π 值。这是一个无穷运算，不能称为算法。

所谓有穷性往往指在合理的运行时间之内，让计算机执行一个算法。如果用 10000 年的时间去运行一个算法，虽然它不是无限的，但超过了合理的限度，也不能算作算法。究竟以什么为限度，并无严格规定，是由人们根据需要和常识来判定的。因为设计算法是为了解决问题，如果执行算法所花的时间超过了人们能容忍的合理限度，这种算法就会被摒弃。

(2) 确定性

算法的每一步都必须准确定义。这是由于我们与之打交道的是机器，含义稍有模糊，则会使执行者无所适从。自然语言传递的信息有大量语义不明之处，人在处理时会根据上下文信息及推理来准确地接受它，而机器却不能。因此，在进行算法设计时应注意这类问题。每一步的含义都是明确的，不存在被理解为有两种或多种可能的情况，即一个算法不应有多义性。

(3) 输入

所谓输入是指执行算法时，需要外界提供某些信息。算法总是要施加到运算对象上，输入描述了运算对象的初始情况，输入是算法的起点。一个算法有零个或多个输入。当算法没有输入时，说明在算法本身中设定了初始条件。

(4) 输出

一个算法有一个或多个输出。

算法的目的就是求解，并且输出解。没有输出的算法是没有意义的。

(5) 有效性

算法中的每一步都应该能有效地执行，执行算法后应该能得到确定的结果。例如一个数被零除，就不能进行有效地操作。

1.3.4 算法的表示

算法可以用不同方式来表示，只要明确无误地写出操作步骤和操作内容即可。常用的有：自然语言、流程图、N-S 结构化流程图等。

1. 用自然语言表示

就是用人们日常用的语言来描述算法。例如上述解一元二次方程的例子就是用自然语言来表示的。用自然语言描述的算法通俗易懂，但它的缺点是比较繁琐冗长，而且容易出现歧义性（同一段文字，不同的人可以有不同的理解）。因此，一般不用自然语言表示算法。

2. 用流程图表示

流程图是指用图示的方法来表示算法，即计算机的操作过程。它用一些几何图形的框来代表各种不同性质的操作。ANSI（美国全国标准化协会）规定的一些常用流程图符号见图 1-5。其中的①是“起止框”，用来表示流程的终止和开始；②是“处理框”，用来表示一般的处理或运算，它只有一个入口，一个出口；③是“判断框”，有一个入口，两个出口。根据给定的条件是否成立决定选择其中一个出口路径；④是“输入输出框”，用来表示计算机输入或输出数据；⑤是“流程线”，用来表示流程的路径和方向；⑥是“连接点”，表示将两个流程图中各自的某一点连接起来。当一个流程图在一页纸内画不下时，可以用“连接点”表示从本页某一点（出口点）连接到下页某一点（入口点）。

上面的“输出 a 和 b 中的大者”的解题算法可以用图 1-6 所示的流程图来表示。

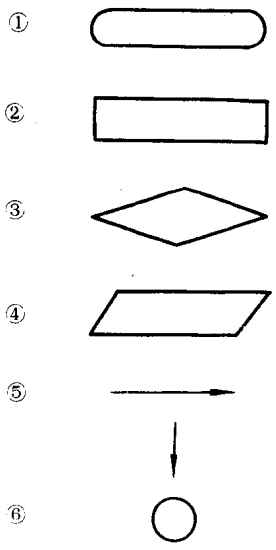

图 1-5

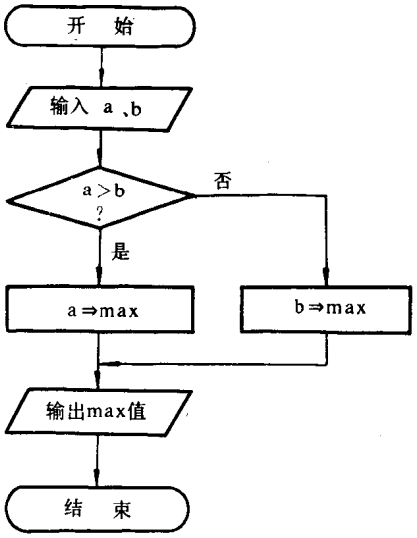

图 1-6

可以看出，用流程图表示算法，逻辑清楚，形象化，容易理解。用带箭头的流程线表示执行的顺序，一目了然。

流程图不是画给计算机的，计算机无法看懂它。流程图的作用是清晰地告诉人们如何一步一步进行操作。人们可以根据流程图来编写程序，这样在编程时思路清晰，减少出错。流程图也常和程序一起提供给用户，以帮助用户理解程序。因此，流程图是程序设计中的有力工具，学习程序设计的人都应当会用流程图。

3. 用 N-S 结构化流程图表示

1973 年美国学者 I. Nassi 和 B. Shneiderman 提出了适用于结构化程序的一种新的流程图形式。在这种流程图中完全去掉了流程线，全部算法写在一个矩形框内，在框内还可以包含其它的框。换句话说，由一些基本的框组成一个大框。这种流程图又称 N-S 结构化流程图（以二人名字的头一个字母命名），简称 N-S 图。

N-S 图用以下的基本元素框来表示程序的三种基本结构。

(1) 顺序结构

如图 1-7 (a) 所示。图 1-7 (b) 给出了相应的流程图表示。在此结构中 A、B 两个操作是顺序执行的，即先执行 A，后执行 B。

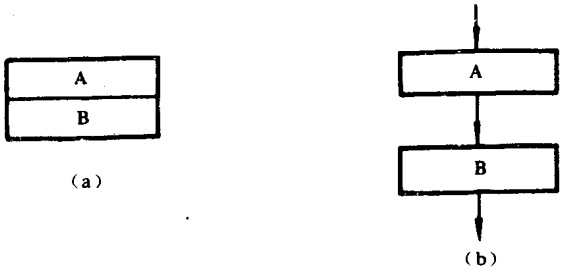

图 1-7

(2) 选择结构

如图 1-8 (a) 所示。图 1-8 (b) 给出了相应的流程图表示。在此结构中有一个判断框，表示当条件 P 成立时执行 A 操作，不成立时执行 B 操作。

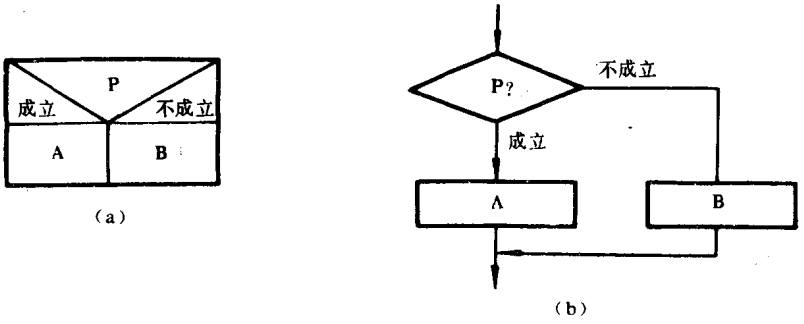

图 1-8

(3) 循环结构

又称重复结构。有两类循环结构：

①当型循环结构

如图 1-9 (a) 所示。图 1-9 (b) 给出了相应的流程图表示。表示当 P1 条件成立时，执行 A 操作，循环重复直到 P1 不成立时为止，此时终止循环，执行下面一个基本结构。如果在开始时，P1 条件就不成立，则一次也不执行 A。

②直到型循环结构

如图 1-10 (a) 所示。图 1-10 (b) 给出了相应的流程图表示。表示先执行 A 操作，然后判断 P2 条件是否成立，若不成立则执行 A 操作，循环重复直到 P2 条件成立时为止，此时终止循环，执行下面一个基本结构。可以看到，不论 P2 条件是否成立，至少要执行一次 A 操作。

用以上三种基本框可以组成复杂的 N-S 图。例如：用 N-S 图表示上面的“输出 a 和 b 中的大者”的算法如图 1-11 所示。