高等学校计算机基础教育系列教材

PASCAL 语言程序设计

谭浩强　田淑清　编著

高等教育出版社

高等学校计算机基础教育系列教材

PASCAL 语言程序设计

谭浩强 田淑清 编著

高等教育出版社

内 容 提 要

本书是高等学校计算机基础教育系列教材之一，按照“程序＝算法＋数据结构＋结构化程序设计方法”的公式组织体系，选材恰当、实用性强，同时，针对广大初学者的特点，叙述力求深入浅出、通俗易懂，并通过大量实例介绍编写程序的方法。即使没有学过其它程序设计语言的读者，学习本书内容也不会感到困难。

本书可作为大专院校计算机应用专业和非计算机类专业教材，也可供计算机培训班和中专使用，也可供初学者自学。

本书配有教学录像片 30 讲，每讲 25 分钟。录像片由高教出版社出版。

高等学校计算机基础教育系列教材

PASCAL 语言程序设计

谭浩强　田淑清　编著

*

高等教育出版社出版

新华书店北京发行所发行

北京印刷一厂印装

*

开本 787×1092　1/16　印张 21.75　字数 540 000

1989年 5 月第 1 版　1989 年 5 月第 1 次印刷

印数 0 001—15 110

ISBN7-04-002285-0/TP・51

定价 4.50 元

出 版 说 明

在计算机科学技术迅速推广应用的今天，在高等学校中向各个专业的学生普遍进行计算机的教育，使每个学生具有必要的计算机知识和应用计算机的能力，是高等教育的一项重要任务.

近年来，全国许多高等学校的理工、农医、经济管理等专业，相继设置了计算机的课程.非计算机专业中的计算机基础教育发展很快.非计算机专业的学生占全体大学生的95%以上，在这个领域中开展计算机教育的情况，在相当大的程度上决定了我国新一代知识分子应用计算机的水平.现在，人们已经认识到，计算机的知识和应用计算机的能力是当代知识分子知识结构中不可缺少的一个重要部分.

全国高等院校计算机基础教育研究会，在总结了许多学校进行计算机基础教育的经验的基础上，1985年提出了在非计算机专业中按四个层次开展计算机教育的方案设想(即：计算机应用入门和程序设计——→微型机的原理与应用——→计算机软件技术基础——→结合各专业的计算机专门课程)，得到各校的赞同，许多学校已按此规划设置计算机课程.

在非计算机专业中开展计算机教育，其要求内容、教材体系和教学方法等都和计算机专业有很大的不同，不能照搬计算机专业的做法.必须根据非计算机专业的特点和规律组织教学.课程设置应该以应用为出发点，以应用为目的，教材编写应该考虑到学生的基础，将科学性、实用性和通俗性结合起来，使之容易被接受.

为了向全国各类学校和专业提供一套适用的教材，我们决定组织编辑出版“高等学校计算机基础教育系列教材”，由高等教育出版社出版.系列教材包括四个层次中的课程所相应的教材.

本系列教材具有以下特色：内容新颖、实用性强、概念清晰、通俗易懂、层次配套.本系列教材的适用对象是：高等学校中非计算机专业、计算机应用专业、中专计算机专业、计算机培训班，以及计算机的自学者.也可供计算机其它有关专业选用.多数教材的写法便于自学.计算机的初学者可以根据需要按照四个层次循序渐进地学习各门课程，以获得所需的知识.

由於计算机科学技术的迅速发展，本系列教材的书目和各书的内容也会不断更新.期望全国各校专家和广大读者对本系列教材的内容和编写方法提出意见，以便不断改进，以满足全国广大读者的要求.

为了编辑好这套系列教材，特成立了由全国高等学校中具有丰富教学经验的专家组成的编辑委员会，负责制订规划、组织稿件、择优出版.

全国高等院校计算机基础教育研究会

1988.11

高等学校计算机基础教育系列教材
编辑委员会

前　言

PASCAL 语言是 70 年代初产生的一种结构化的计算机语言，目前已成为世界上广泛流行的一种程序设计语言．在结构化程序设计方法提出之后，PASCAL 与其它同期的语言相比，在实现结构化程序方面有着显著的优势．PASCAL 语言的数据类型丰富、语句功能较强、结构清晰、书写格式自由、表达能力较强、可移植性好、语言风格优美．这些使 PASCAL 成为一种较理想的教学用的语言，它有利于培养学生良好的程序设计风格．

PASCAL 语言不仅已在高等学校中的计算机专业中学习使用，而且逐渐成为所有计算机软件人员的一种必修的语言，在高等学校中，不少非计算机专业也已经或准备开设 PASCAL 语言课程．根据这个需要，我们为初学者编写了这本书．本书的对象是非计算机专业的师生和在职干部．本书的写法力求做到深入浅出，不从抽象概念出发去讨论问题，而是以应用为目的，通过大量具体实例说明 PASCAL 的应用．如果有一门其它语言的基础，学习 PASCAL 语言是不会困难的．即使从未学习过任一种计算机语言，也可以学懂本书，并掌握 PASCAL 程序设计方法和技巧．

本书的写法不以语法规则为重点，也不从语法规则入手．著名计算机科学家沃思提出了一个公式：程序＝算法＋数据结构．在结构化程序设计已被普遍接受的今天，程序已不再是反映程序设计人员个性的一种“艺术品”，应当按照工程的方法来组织软件的生产．结构程序设计方法已成为程序设计人员必须遵循的一种规范．本书把算法、数据结构和结构化程序设计方法三者有机地结合起来．实际上，我们的公式是：程序＝算法＋数据结构＋结构化方法．

算法是解决一切问题的钥匙，离开算法就谈不上程序．本书采取与其它书籍不同的写法，从算法入手，使读者首先从算法的角度去思考问题，而不致一下陷入语言的细节．只要掌握算法设计，用任何一种计算机语言去实现算法都不会太困难．本书通过大量的例子介绍各种基本的和典型的算法，希望读者在研究它们时，首先注意在接受一个题目时，如何构思算法，要按照“自顶向下、逐步细化、模块化”的方法由粗到细地设计算法，在掌握了算法之后再去考虑如何用一种计算机语言实现它，这时就要了解和掌握语言的规则．如果采取这样的方法学习，那么通过本书的学习，首先学到的不是一种具体的语言本身，而是具有普遍意义的程序设计技术．如果将来使用的不是 PASCAL 语言而是其它任一种语言，读者完全可以通过自学，很容易地将一个 PASCAL 程序转换成其它语言的程序．

数据结构丰富是 PASCAL 语言的一大优点，本书不是孤立地、枯燥地介绍 PASCAL 中有关数据结构的知识，而是随着算法设计的不断深入，由简到繁地介绍各种数据结构以及在 PASCAL 中如何实现，这些都是结合具体问题展开的，使读者学习时不致感到抽象难懂和“台阶”太大．

结构化方法贯穿于本书的始终，本书不采用传统的流程图而采用结构化流程图——N-S 流程图．读者通过本书的学习，会自然摒弃过去那种随心所欲地、无规律地设计程序的过时的方法，而自然养成结构化程序设计的良好习惯．在本书的程序中基本上不采用 GOTO 语句，而用三种基本结构组成一个结构化程序．

离开计算机语言，也谈不上程序设计．因此也要正确掌握语法规则的使用．

总之，在本书中，算法、数据结构、结构化方法和语言规则几个方面是自然地、有机地融合在一起的．希望读者在学习每一部分中都注意掌握这几方面的基本知识，不要孤立地去学习，更不要企图用死记硬背的办法来掌握本书的内容．

考虑到非计算机专业学生的特点以及学时有限，本书对动态数据结构和文件这两部分没有作太深入的讨论，而只介绍其基本概念和一般的使用，有了这个基础，今后读者完全有条件向深入发展．

本书力求做到科学性、实用性、通俗性三者的统一．科学性和通俗性并不是排斥的，通俗性应当遵循科学性，而科学性只有通过通俗易懂的叙述才能为广大人们所接受．在介绍一个概念时，重要的是使读者正确理解它的实质以及能正确地使用它，而不是引导读者去咬文嚼字地死抠定义．我们的想法是：根据不同读者的情况采取不同的方法去介绍有关概念．既使读者能有效地接受，又使读者建立起正确的科学概念．在我们所编写的一些书籍中，用这种方法，取得较好的效果．我们愿意在这一方面继续进行探索．

最后我们还想说明：每一种计算机语言都有自己的适用领域．每一种语言都有自己的优缺点，随着科学技术的发展，每一种语言又都会暴露它的不足和局限性，需要不断地修改补充，使之完善．我们不应当用一种语言去排斥另一种语言，而应当采取“百花齐放、推陈出新”的态度，在应用中推动各种语言的发展．只要是在实际中有用的语言，都应当推广使用．在广泛使用中，必定会提出新的要求，暴露出某些缺陷，然后推动它的发展．无论 BASIC，FORTRAN，COBOL，PASCAL 都在不断发展和完善．PASCAL 也不是十全十美的计算机语言，尤其是标准 PASCAL 没有动态可调数组和随机文件的功能，致使在实际应用中感到不便．此外，PASCAL 没有提供可供分别编译的模块，而要求在一个程序模块中写出所有的过程和函数，这对于一个大任务分别进行程序编写是不方便的．为了克服 PASCAL 的不足，PASCAL 的作者 N．Wirth 于 1980 年又发表了 Modula-2 语言，它是 PASCAL 的扩充和发展．但是，目前 Modula-2 还没有得到广泛推广．有了 PASCAL 的基础，以后需要时再学习 Modula-2 或其它语言也是不难的．

本书由谭浩强、田淑清编写，谭浩强编写第一至第八章，田淑清编写第九至第十三章．在编写过程中得到许多同志的鼓励和帮助．高教出版社以很大的热情组织了本书的出版．科普出版社的朱桂兰同志和高教出版社的唐兆亮同志为本书的出版做了许多工作．清华大学计算机系郑人杰副教授在本书出版前审阅了全书，提出许多有益的意见．我们对此一并表示感谢．本书的不足之处，请专家和读者指正．

为方便读者学习，我们另编了一本《PASCAL 程序设计习题集》，提供四百多个 PASCAL 习题和参考解答，由高教出版社出版，供读者参考．

作者

1988.9

目　录

第一章 算 法

学习本书的目的是学会编写 PASCAL 语言程序，利用电子计算机去解决各种问题．应当说明，要使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执行的指令，这就是程序．

计算机为什么能处理各种不同的问题呢？这是由于人们事先对各类问题进行了分析、确定了解决问题的方法和步骤，然后根据它编写出计算机程序，再让计算机执行这个程序，才得出最后结果．

在学习程序设计时，既要掌握所使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，这是程序设计中的关键．语言只是一个工具．只懂得语言的规则并不能保证能编制出有效的、高质量的程序．本章所讨论的“算法”，就是研究解题的步骤和方法的．这是以后各章的基础．希望读者能对它给予充分的注意．

§1.1 算法的概念

做任何事情都有一定的步骤．例如，要计算 1+2+3+4+5 的值，一般是先将 1 和 2 相加得 3，再将 3 加 3 得 6，再将 6 加 4 得 10，最后再加 5 得 15．无论手算、心算或用算盘、计算器计算，都要经过有限的、事先设计好的步骤．为解决一个问题而采取的方法和步骤，称为“算法”（Algorithm）．或者说，算法是解题方法的精确描述．解决一个问题的过程就是实现一个算法的过程．

对同一个问题，往往有不同的解题方法，例如要计算 $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-\cdots+\frac{1}{99}-\frac{1}{100}$，可以采用先进行 $(1-\frac{1}{2})$ 的计算，得结果 $\frac{1}{2}$，再加 $\frac{1}{3}\cdots$，即自左而右逐项相加或相减．也可以将该多项式写为两个多项式之差：$(1+\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{99})-(\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{100})$．还可以有其它解法，例如先通分再运算，当然，各种方法有优劣之分．为了有效地进行运算，应当选择合适的算法．

不仅数值计算的问题要研究算法．实际上，做任何事情，都需要事先设想好进行的步骤和方法，这就是“算法”．例如，打太极拳、跳迪斯科、乐队奏曲、操纵机器、厨师炒菜，都是按照一定的步骤进行的．太极拳动作图解就是“太极拳的算法”，一个菜谱也是一个“算法”，厨师炒菜就是实现这个算法．同样，一个工作计划、教学计划、生产流程、乐谱、珠算口诀等都可称为“算法”．当然，我们只讨论计算机算法，即计算机可以实现的算法．例如，让计算机执行“1+2”的运算是可以实现的，而让计算机执行“西红柿炒鸡蛋”的算法是不行的（至少在目前阶段如此）．

计算机算法一般分为两大类：数值运算和非数值运算．如，求若干数之和，求方程的根，求一个函数的定积分等，都属于数值运算．而将若干个人名按字母顺序排序，图书情报资料检索、计算机绘图等则属于非数值运算．目前，数值运算的算法比较成熟，对各类数值计算问题都有成熟的

算法可供选用.通常把这些算法分类汇编成册(给出计算机程序),称"算法汇编",还往往把这些程序存放在磁盘或磁带上,供用户调用.例如有些计算机系统提供"教学程序库".而非数值运算的种类繁多,情况各异,难以一一罗列.因此,只能对一些典型的非数值运算算法(如排序的算法)作比较深入的讨论研究,其余的许多问题需要使用者参照已有的类似算法重新设计解决特定问题的专门算法.

本书不可能罗列所有算法,只是通过一些典型算法的讨论,帮助读者了解如何构造一个算法,推动读者举一反三.希望读者在阅读本书时,首先把注意力放在理解和研究算法上.

§1.2 简单算法举例

为了使读者理解如何设计一个算法,下面举几个简单的计算机算法的例子.

[**例1**] 有两个杯子A和B,分别盛放酒和醋,要求将它们互换(即A杯原来盛放酒,现要改盛醋.B杯则相反).

根据常识,必须增加一个空杯C作为过渡.见图1.1.其算法可以表示为:

步骤1:先将A杯中的酒倒在C杯中.

步骤2:再将B杯中的醋倒在A杯中.

步骤3:最后将C杯中的酒倒在B杯中.

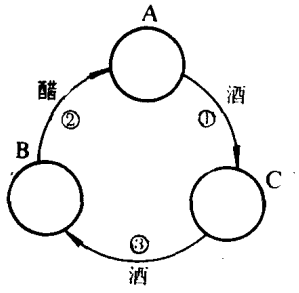

图1.1

这就是以后要用到的使两个变量的值交换的方法.上面的算法可以简化表示如下:

①A⇒C

②B⇒A

③C⇒B

[**例2**] 从十个数中挑选出最大的数.

这个问题的思路可以用"打擂台"来比喻:先有任意一人在台上,然后第二个人与他比武,胜者留在台上,如此继续下去,直到第十个人比完为止(一共比九次),最后留在台上者为胜.

算法当然可以表示如下:

①先任选一数放在一个匣子A中.

②将第二个数与A匣子中的数相比,大者放入A匣子中.

③再将第三个数与A匣子中的数相比,大者放入A匣子中.

⋮

⋮

⑩最后将第十个数与A匣子中的数相比,大者放入A匣子中.此时A匣子中的数就是最大的数.

这样写算法虽然是正确的,但太繁琐.可以简化如下:

①选一数放在A中.设一记数器N,开始时N的值置为零(表示此时已比较了零次).

②将下一个数与 A 中的数相比，大者放入 A 中.

③使 N 的值增加1.（表示增加一次“比较次数”）

④如果 N 的值小于9，则重新执行第②步.如果 N 的值大于或等于9（表示已比较了九次以上），停止循环.此时 A 中的数就是最大的数.

显然，这种用“循环”来表示的算法比较简练.如果题目要求改为“从1000个数中挑选最大者”，只要将算法中第④步中的“9”改为“999”即可.

[例3] 求两个正整数 m 和 n 的最大公约数.

我们先回顾一下在算术中是如何用“辗转相除法”来求最大公约数的.为易于理解，代入具体的数值，m＝64，n＝14.算法如下：

①以 n 除 m，求 $\frac{m}{n}$ 的整数商和余数 r_1. $\frac{64}{14}$ 的整数商为4，余8，即 $r_1 = 8$.

②判断余数 r_1 是否为0，如 $r_1 = 0$，则 n 就是最大公约数.如果 $r_1 \neq 0$，则还未找到最大公约数，要继续做下去，执行第③步.

③将 n 除以 r_1，求 $\frac{n}{r_1}$ 的整数商和余数 r_2. n＝14，$r_1 = 4$，$\frac{14}{4}$ 的余数为 2，即 $r_2 = 2$.

④判断 r_2 是否为0？如 $r_2 = 0$，则除数 r_1 就是最大公约数.如果 $r_2 \neq 0$，则还未找到最大公约数，要继续做下去，执行第⑤步.

⑤将 r_1 作为被除数，r_3 为除数，再求 $\frac{r_1}{r_2}$ 的整数商和余数 r_3. $\frac{r_1}{r_2} = \frac{4}{2} = 2$，余数 $r_3 = 0$.

⑥判断 $r_3 = 0$？如 $r_3 = 0$，则本次除法中的除数 r_2 就是最大公约数.如果 $r_3 \neq 0$，则再求 $\frac{r_2}{r_3}$ 的余数 r_4，…一直做下去，直到某一次除法 $\frac{r_{n-1}}{r_n}$ 的余数为零时，r_n 即为最大公约数.

可以用以下方法表示上述算法：

用 m 做被除数，n 做除数，r 做余数.

①求 $\frac{m}{n}$ 的余数 r.

②若 r＝0，则 n 为最大公约数.若 r≠0，执行第③步.

③将 n 的值放在 m 中，将 r 的值放在 n 中.

④返回重新执行第①步.

如果 m 和 n 两个数中，事先不知道孰大孰小，则应在第①步前增加一个步骤：使大数放在 m 中，小数放在 n 中.

这样表示的算法既严格又简洁，但是需要经过一番思考以后才能正确地写出来.这个思考的过程就是“由具体到抽象”的过程.在本例中，用 m 统一代表各次除法中的被除数，以 n 统一代表除数，用 r 统一代表余数，就是一种抽象的方法，是经过对各个数据归纳整理后得到的一种抽象.在程序设计中我们将要大量使用这种方法.请读者逐渐学会和掌握这种方法.这种方法便于用计算机进行循环处理.在本例中，每次都是求 $\frac{m}{n}$ 的余数 r，而不出现 $r_1, r_2, r_3, r_4 \cdots r_n$.计算机只需简单地循环执行某些操作即可.在写计算机算法时，应当将算法写成使计算机便于执行的形式.

[例4] 求 n!.

如果 n＝5，即求1×2×3×4×5.根据上面指出的原则，我们先设 S 代表累乘之积，以 T 代表乘数.

①使S=1,T=1.

②使S=S×T,得到的积仍放在S中.

③使T的值加1.

④如果T≤n,返回重新执行第②步.如果T>n,则不再返回②,而停止循环,此时S中的值就是n!.

如果求5!,则第1步就是判断"T≤5?".

显然,只要n的值不小于零,这个算法都是正确的.这种算法具有一般性,不因n的值不同而改变算法的写法.

[例5] 给一个正整数N,判定它是否素数.

素数亦称质数,它的特征是:除了1和该数本身之外,不能被任何整数整除.例如17是素数,因为它除了能被1和17整除以外,不能被2～16之间的任何整数整除.而16不是素数,因为它能被2,4,8整除.

判断一个数N是否素数的最基本的方法是将N被2,3,…,(N-1)除,如果都除不尽,则N必为素数.

根据此思路写出以下算法:

设除数为I,I的值由2变化到(N-1).

①置I的初值为2.

②将N被I除,得到余数R.

③判断R=0?,如果R=0,表示N能被I整除,N不是素数,不再进行下去,算法结束.如果R≠0,表示N不能被I整除,N有成为素数的可能,应继续进行下去,执行第④步.

④使I的值加1.

⑤如果I≤N-1,则返回重新执行②.如果I>N-1,表示N已被2到(N-1)除而不能被整除,可以判定N是素数.算法结束.

实际上,N不必被2到(N-1)各数除,只需被2到$\sqrt{N}$除即可.例如判断17是否素数,只需将17被2,3,4除,如都不能整除,则17必为素数($\sqrt{17}$应为4.1231056,取其整数部分得4).又如判断97是否素数,不必使97被2到96除,只需被2到9除即可.这是为什么,请读者自己思考.

§1.3 算法的特性

一个算法应具有以下特点:

1.有穷性.一个算法应包含有限个操作步骤,而不能是无限的.例如让歌唱家唱一支永远唱不完的歌,这是不合理的,不能称为"算法".

事实上,所谓有穷性往往指在合理范围之内,如让计算机执行一个循环算法,循环到10000年才结束,虽然它不是无限的,但超过了合理的程度,也不能算作"算法".究竟以什么为限度,并无严格规定,是由人们的常识和需要判定的.因为设计算法是为了解决问题,如果执行算法所化的时间超过了人们能容忍的合理限度,人们便会摒弃它.

2.确定性.算法的每一步都应当是明确无误的,不能含义模糊,使执行者无所适从.例如,"吃

完饭上公园去”这句话是不明确的，是指吃完早饭，还是中饭或晚饭？如果在特定的条件下（如二人正在吃中饭）或二人相互默契（经常吃完晚饭去公园散步），对方可以根据其它因素把它补充成可以执行的，否则就难以执行．这种可以被理解为二种（或多种）可能的，称为“歧义性”或“多义性”（即有多种含义）．一个算法不应有“多义性”．

3．有零个或多个输入．上面判断素数的例子有一个输入（N），求最大公约数的例子有两个输入（m 和 n），十个数找最大者的例子则有十个输入．有的算法也可以没有输入，例如，舞蹈演员按指定节目表演舞蹈，可以说她在执行一个算法．在执行这个算法过程中无需任何输入，演员会自动做出各种动作．请注意，不要把指定算法误认为是“输入”．所谓“输入”是指执行指定的算法时，需要外界提供某些信息．

4．有一个或多个输出．算法的目的是求“解”，“解”就是结果，就是“输出”．例如求最大公约数例子中的输出是最大公约数．判定素数例子的输出是给出“N 是素数”或“N 不是素数”的信息．不给出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，执行算法最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．同样，“到火星上拣一块石头回来”，在目前来说是无法执行的．

对于使用现成算法的人来说，可以把一个算法看作是一个“黑箱子”，在“黑箱子”中是怎样工作的，他可以不必关心，他只要提供某些输入，便可得到确定的输出．例如，有一个制造零件的自动化的机器，人们只需将毛坯输入，便可得到加工好的零件．至于机器内部是怎样工作的，对使用者来说可不必深入了解．又如求最大公约数的算法，可以用图1．2所示．只要输入 m 和 n，就能得到最大公约数．

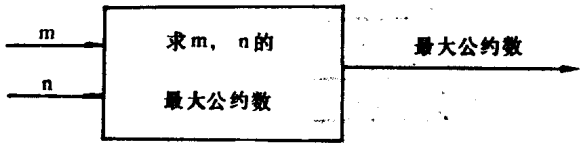

图 1.2

但对于程序设计人员来说，不仅会使用现成的算法，还要会设计算法，即要设计出算法中的每一个步骤．

§1.4 算法的表示

为了描述一个算法，可以采用许多不同的方法．常用的有：自然语言；流程图；结构化流程图；伪代码等．

1.4.1 自然语言

前面§1．2节介绍的几个例子都是用自然语言来表示的．自然语言就是人们日常使用的语言，可以是汉语或英语或其它文字．用自然语言描述算法通俗易懂，但是它的缺点是：

1．比较繁琐冗长．往往要用一段冗长的文字才能说清楚所要进行的操作．例如，“把名字为 n 的存储单元的值放到名为 m 的存储单元中”不如写成“n⇒m”简洁．“使 I 的值加1”不如写成“I+

1⇒1".

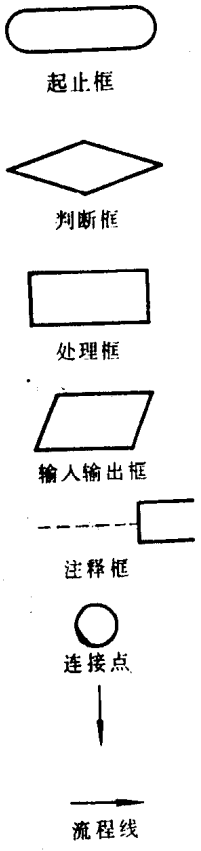

图1.3

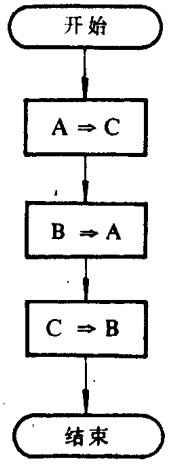

图1.4

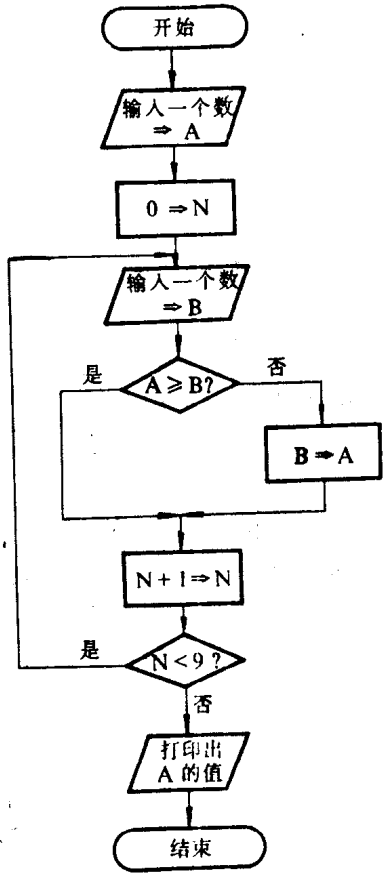

图1.5

2. 容易出现"歧义性". 自然语言往往要根据上下文才能正确判断出其含义，不太严格，例如"张三要李四把他的笔记本拿来"，究竟指的是谁的笔记本，就有"歧义性".

3. 用自然语言描述顺序执行的步骤比较好懂，(如§1.2中的例1). 但如果算法中包含判断和转移(如§1.2中例3中的第④步)时，用自然语言就不那么直观清晰.

因此，除了对那些很简单的问题之外，一般不用自然语言表示算法.

1.4.2 流程图

流程图是用图形来表示算法. 用一些几何图形的框来代表各种不同性质的操作. ANSI(美国国家标准化协会)规定的一些常用流程图符号(见图1.3)，已被大多数国家接受.

下面是将§1.2中的五个算法例子用流程图来表示.

[例1] A 和 B 互换 见图1.4.

[例2] 从十个数中选出最大者. 见图1.5.

在画此流程图时，考虑了计算机解题的特点，每输入一个数据都应当存放在一个存储单元中(A，B 和 N 是存储单元的名称，关于存储单元的概念详见第二章). A 的值和 B 的值相比较后，如

A 的值大，或 A＝B，则 A 的值不改变，如 B＞A，则将 B 的值放入 A 中以取代 A 的值，这样就实现了“大者存放到 A 中”的要求.

请读者特别注意循环的条件，在最下面的一个菱形框中写的是“N＜9”，即当 N＜9时返回重新执行“输入一个数⇒B”. 请读者思考，能否将“N＜9”条件改为“N≤9”？为什么？许多读者常在这种地方搞错. 又：如果开始时不是使 N 的值等于零，而是“1⇒N”，则循环的条件“N＜9”要不要修改？为什么？

[例3] 求最大公约数. 见图1.6.

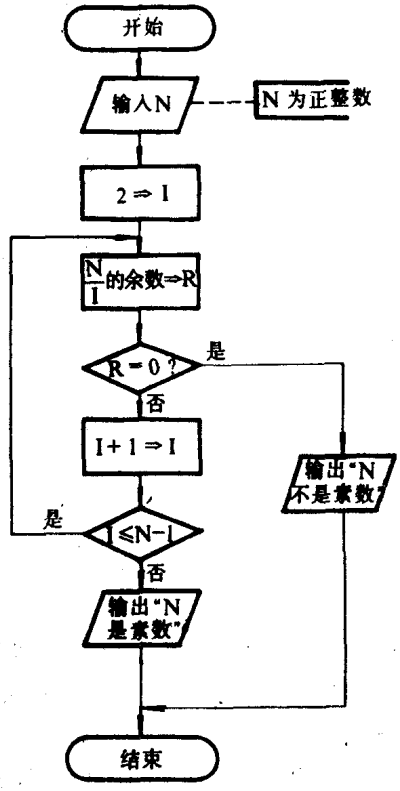

图1.6

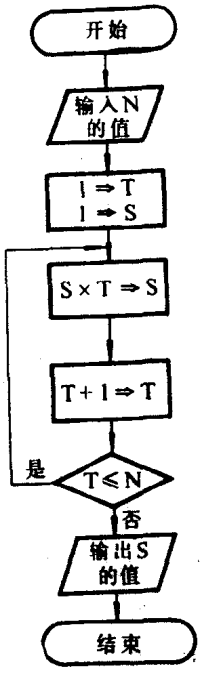

图1.7

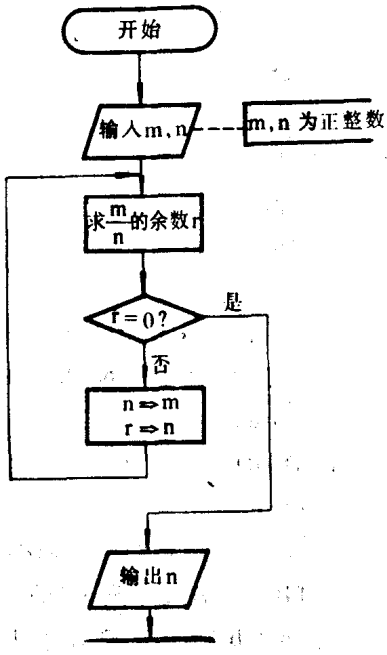

图1.8

图1.6中用了注释框对某一个步骤作必要的注释说明.

[例4] 求 n!. 见图1.7.

循环条件应是“T≤N”，不要错写成“T＜N”.

[例5] 判断 N 是否素数(N≥3). 见图1.8.

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. 但是流程图占用篇幅多，而且当算法比较复杂时，每一步骤要画一个框，比较费事.