高等职业技术电子信息类专业教材

程序设计技术

鲍有文　主编

鲍有文　张　翼　李京平　编著

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

全书共包含 10 章内容。第 1 章至第 3 章全面介绍了与程序设计相关的基础知识。第 4 章和第 5 章主要介绍了线性表、栈和队列、树和二叉树等几种基本类型的数据结构，以及在程序设计中经常遇到的常用算法。第 6 章、第 7 章和第 9 章结合当前程序设计技术的新发展，介绍了可视化程序设计、面向对象程序设计和数据库设计的基础知识。第 8 章结合软件工程学中有关软件测试和调试的基础知识，介绍程序测试及调试的基本方法和过程。第 10 章通过引进三个实训题目，使读者对所学知识有一全面、综合的实践过程。

本书可作为高等学校计算机应用技术专业的本科或高职学生的专业基础课教材，也适用于已掌握了一门程序设计语言、需要进一步学习程序设计基本方法和算法的读者自学。

图书在版编目(CIP)数据

程序设计技术/鲍有文主编. —北京：电子工业出版社，2002.9
高等职业技术电子信息类专业教材
ISBN 7-5053-7668-3

Ⅰ. 程… Ⅱ. 鲍… Ⅲ. 程序设计—高等学校：技术学校—教材 Ⅳ. TP311

中国版本图书馆 CIP 数据核字(2002)第 041140 号

责任编辑：应月燕
印　　刷：
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×1 092　1/16　印张：12.75　字数：326 千字
版　　次：2002 年 9 月第 1 版　2002 年 9 月第 1 次印刷
印　　数：5 000 册　定价：18.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。
联系电话：(010)68279077

前　言

本书遵循循序渐进、深入浅出、难点分散的编写思想，从强调与程序设计技术相关知识的基础性、实用性和先进性的角度出发，把程序设计的基本理论及方法、基本数据结构与常用算法、面向对象程序设计基础、数据库基础知识和软件工程基础等内容有机地融合为一体。全书内容可划分为五个部分。第一部分由第 1 章至第 3 章的内容组成，全面介绍与程序设计有关的基础知识。第二部分包含了第 4 章、第 5 章的内容，主要介绍了线性表、栈和队列、树和二叉树等几种基本类型的数据结构，以及在程序设计中经常遇到的常用算法。第三部分由第 6 章、第 7 章和第 9 章的内容构成，目的是结合目前程序设计技术的新发展，介绍可视化程序设计、面向对象程序设计和数据库设计的基础知识。第四部分指第 8 章内容，这章结合软件工程学中有关软件测试和调试的基础知识，介绍程序测试及调试的基本方法和过程。最后一部分即第 10 章内容，本章通过三个实训，力求使读者对所学知识有一全面、综合的实践过程。

为了强调程序设计技术这门课程的实践性和便于读者自学，本书具有以下特点：

(1) 不过分强调程序设计的理论知识，也不强求数据结构和算法内容介绍上的面面俱到。

(2) 在内容安排上，既注意介绍程序设计的基本原理和方法，又注意引进程序设计方面的新概念、新技术及多种程序开发环境的应用，以适应当前计算机技术的发展。

(3) 通过引进大量的例题、自测习题，从问题分析和算法设计入手，使读者尽快掌握程序设计的完整过程，掌握以工程方法和风格进行程序设计的基本功。

全书由具有丰富教学经验和工程经验的一线教师编写，出版前已作为相关课程的教学用书。本书可作为高等学校计算机应用技术专业的本科或高职学生的专业基础课教材，建议讲授 48 ~ 64 学时。本书也适用于已掌握了一门程序设计语言、需要进一步学习程序设计基本方法和算法的读者自学。

在本书的筹划和编写过程中，曾得到谭浩强教授和高林教授的帮助和指导，在此，对他们表示衷心的感谢。

由于水平有限，本书中难免出现错误和疏忽之处，恳请得到专家和读者的批评和指正。

作　者

2002.2

目　录

第 1 章　程序设计导论

本章学习目的：

(1) 了解算法的基本概念和计算机算法的特性。

(2) 掌握常用的算法描述工具。

(3) 了解数据结构的基本概念。

(4) 掌握程序设计的基本步骤和要求。

虽然计算机能够处理的问题是千变万化的，但是计算机最终只能识别和执行由计算机语言编写的命令，即通常所说的“程序”，由此就引出了一系列与程序相关的概念，并要求学习计算机知识的人必须掌握程序设计的基本技术。一般地说，程序设计通常要涉及下面几方面的内容：

(1) 为待解决的问题确定正确的解题方法和步骤。

(2) 对问题所涉及的数据特性进行分析并选择合适的数据结构。

(3) 选用适当的程序设计方法，提高程序设计质量。

(4) 选用优质的软件工具，实现程序的编码和测试。

程序设计中涉及到的这四个方面是互相依赖、互相补充、不可分割的，最终达到简化问题、解决问题的目的。

1.1　算法

1.1.1　什么是算法

人们在处理日常生活中的各类事情时，都会有意或无意地按照一定的方法和步骤来进行。例如，某人为了尽量减少上班路途上花费的时间，就会考虑选择一个最佳的交通方案，它可能是：离开住处→步行→乘地铁→步行→转乘公共汽车→再步行→到达目的地。这个交通方案其实就是解决一个特定上班路线问题的“算法”。广义地讲，所谓“算法”就是解决问题时所采用的方法和步骤，换句话说，算法就是对问题处理过程的一种描述。也正是从这个意义上说，处理任何事情都存在着算法。显然，解决同一个问题的算法通常不止一个，比如上班的路线可能有多条。但是一般地说，对一个具体问题而言，总存在着一个最佳算法，人们在解决问题时实际上就是在为问题寻找着一个最佳算法。本书只讨论用计算机处理问题时所涉及到的算法，即计算机算法，并在以后的叙述中简称为算法。

1.1.2　计算机算法的特性

首先看一个计算机求解 $\sum_{n=1}^{10}\frac{1}{n}$ 的例子。

【例 1.1】　计算 $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{10}$。

问题分析：计算 $\sum_{n=1}^{10}\frac{1}{n}$ 的过程中具有两个特点。

(1)重复进行加法计算(通常称做“累加”),每加1个数,“和”都要发生变化。

(2)加数的变化是有规律的,即每加一次后加数的分母值增1。

解决这个问题的算法可以写做:

(1)设用sum表示累加结果,用n表示加数的分母。

(2)先分别为sum和n赋初值0和1,表示为$0\rightarrow$sum和$1\rightarrow n$。

(3)进行加法运算,每次求和结果赋给sum,表示为$\frac{1}{n}+\text{sum}\rightarrow\text{sum}$。

(4)加数的分母增1,表示为$n+1\rightarrow n$。

(5)每次分母增1后判断n值:当$n<10$或$n=10$时,返回第(3)步,重复执行(3)~(5)步;当$n>10$时,执行第(6)步。

(6)输出sum值,sum值即所求。

以上算法就是用计算机解决$\sum_{n=1}^{10}\frac{1}{n}$问题的方法和步骤。显然这个算法不是惟一的,例如,计算$\sum_{n=1}^{10}\frac{1}{n}$时还可以采用先把10个加数通分后再相加的方法。一般而言,计算机算法应当具有以下5个特性。

(1)有穷性。任一个算法只能包含有限个执行步骤,即任何算法必须在有限时间内完成。例1.1中的操作步骤是6个,操作的结束条件是$n>10$。如要改成计算n为非定值时的$\frac{1}{n}$累加和,显然就不能用有限个步骤和有限时间来完成,即不满足有穷性的要求。

(2)确定性。算法中的每个步骤都必须有明确的定义,不能有“多义性”。例如,“当x大于1时,把小于x的若干数相加”或“当x很大时,停止x增1($x=x+1$),否则重复执行$x=x+1$计算”等都属于含糊不清且具有多义性的算法描述。

(3)有效性。计算机算法中的每个步骤都应当能被计算机有效地执行,并能产生有效的结果。例如,“1个数被零除”就不是有效的算法。再例如,对于一台只能表示最大整数值为32 767的计算机,让其表示大于32 767的整数也是不能被有效执行的算法。

(4)零个或多个输入。开始执行算法时,可有零个或若干个输入值。如例1.1的算法就属于零个输入的情况。

(5)一个或多个输出。算法执行结束后,必须输出算法的执行结果(结果可能不止一个)。如例1.1中有一个输出,即输出$\sum_{n=1}^{10}\frac{1}{n}$的计算结果。

这里需要强调的是,在为实际问题进行算法设计时,一定要使算法符合上述算法特性,以保证所设计的算法能被计算机正确地执行并得到预期的结果。此外,算法设计的初学者除了注意算法的正确性,还应尽量注意算法的简明性,以使算法易于理解、交流和修改。

1.1.3 常用的算法描述工具

在进行算法设计时,首先要解决的问题是如何把算法正确地表示出来,这就要依赖于算法的描述工具。目前虽然存在着多种形式的算法描述工具,但是都可以归结为文字描述工具和图形描述工具这两类。在文字描述工具中,常用的描述形式有自然语言和伪码两种;在图形描述工具中,常用的有流程图、N-S图、PAD图等工具。限于篇幅,本节只介绍自然语言、伪码、流程图和N-S图这4种算法描述工具。

1. 使用自然语言描述算法

所谓“自然语言”指的就是日常生活中使用的语言，如汉语、英语或数学表达式等。请看下例。

【例 1.2】 用自然语言描述求一元二次方程 $ax^2+bx+c=0(a\neq 0)$ 全部根的算法。

问题分析 根据代数知识可知，一元二次方程的根可能存在下面 3 种情况。

(1) 若 $b^2-4ac>0$，则有两个不相等的实根，即：

$$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

(2) 若 $b^2-4ac=0$，则有两个相等的实根，即：

$$x_{1,2}=-\frac{b}{2a}$$

(3) 若 $b^2-4ac<0$，则有一对共轭复根，即：

$$x_{1,2}=\frac{-b}{2a}\pm\frac{\sqrt{-(b^2-4ac)}}{2a}\mathrm{i}$$

用自然语言描述算法如下。

(1) 输入 a,b,c 的值，设 $d=b^2-4ac$；

(2) 根据 d 值的正负，求出根是实根还是复根。

① 如果 $d>0$，则得到两个不相等的实根：

$$x_1=\frac{-b+\sqrt{d}}{2a}\quad,\quad x_2=\frac{-b-\sqrt{d}}{2a}$$

② 如果 $d=0$，则得到两个相等的实根：

$$x_{1,2}=\frac{-b}{2a}$$

③ 如果 $d<0$，则得到两个不相等的复根：

$$x_1=\frac{-b}{2a}+\frac{\sqrt{-d}}{2a}\mathrm{i}\quad,\quad x_2=\frac{-b}{2a}-\frac{\sqrt{-d}}{2a}\mathrm{i}$$

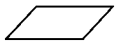
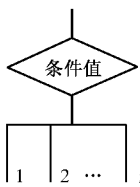
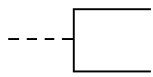
(3) 输出 x_1 和 x_2 的值。

可以看出，用自然语言描述的算法与人们的日常生活用语基本相似，只不过更简单明了。一般地说，使用自然语言进行算法描述时，应尽量省略主语，不用任何虚词，并尽可能采用直接以动词开头的祈使句。

2. 使用流程图描述算法

流程图又称为“框图”，它是一种由几何图形、流程线和文字说明组成的图形。目前普遍采用的流程图符号如表 1.1 所示。

表 1.1　流程图常用符号

流程图符号	含　义
	数据输入/输出框，用于表示数据的输入或输出
	处理框，描述基本的操作功能，如“ 赋值 ”操作、数学运算等
	两分支判断框，根据框中给定的条件是否满足，选择执行两条路径中的一条
	开始/结束框，用于表示算法的开始和结束
	连接符，用于连接流程图中不同地方的流程线
	流程线，表示流程的路径和方向
	多分支判断框，根据框中的“ 条件值 ”，选择执行多条路径中的一条
	注释框，框中内容是对某部分流程图做的解释说明

下面使用表 1.1 中的流程图符号描述计算 $\sum_{n=1}^{10}\frac{1}{n}$ 的算法。

【例 1.3】　用流程图描述计算 $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{10}$ 的算法。请见图 1.1。

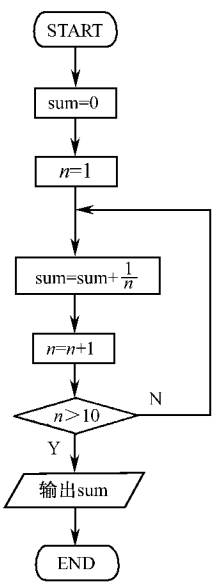


图 1.1　求 10 个自然数的倒数之和的流程图

用流程图描述算法时，一般需要注意以下几点。

(1) 应根据解决问题的步骤从上至下顺序地画出流程图，各图框中的文字要尽量简洁。

(2) 为避免流程图的图形显得过长，图中的流程线应尽可能短。

(3) 用流程图描述算法时，流程图的描述可粗可细，总的原则是：根据实际问题的复杂性 流程图达到的最终效果为，依据此图就能使用某种程序设计语言实现相应的算法(即完成编程)。

3. 使用伪码描述算法

伪码(Pseudo-code)也称过程设计语言 PDL(Process Design Language)。它是利用自然语言的功能描述和若干基本控制结构来描述算法。伪码没有统一的标准化。本书中的伪码将使用以下 6 种控制结构。

(1) 顺序结构。

由类似自然语言(汉语或英语)中的语句顺序排列构成，通常用于表示赋值、输入、输出等顺序执行的步骤。

(2) 选择结构。

IF 〈条件〉 THEN
〈语句组 1〉
[ELSE
〈语句组 2〉]
ENDIF

(3) CASE 结构。

CASE 〈状态选择因子〉 OF
〈状态 1〉:〈语句组 1〉
〈状态 2〉:〈语句组 2〉
……
〈状态 n〉:〈语句组 n〉
[OTHERWISE:〈语句组 $n+1$〉]
ENDCASE

(4) 当型循环结构。

WHILE 〈条件〉 DO
〈循环体〉
ENDDO

(5) 直到型循环结构。

REPEAT
〈循环体〉
UNTIL〈条件〉

(6) FOR 循环结构。

① FOR 〈变量名〉=〈初值〉 TO 〈终值〉[〈STEP 正增量〉]
〈循环体〉
ENDFOR

② FOR 〈变量名〉=〈初值〉 DOWNTO 〈终值〉[〈STEP 负增量〉]
〈循环体〉
ENDFOR

除了上述 6 种控制结构外,本书对伪码还有如下约定。

(1) 每个基本算法均用 BEGIN 和 END 表示算法的‘开始’和‘结束’,在 BEGIN 和 END 后也可跟有本算法的名称。

(2) 赋值操作:〈变量名〉=〈表达式〉
〈变量名 1〉=〈变量名 2〉=〈…〉=〈变量名 n〉=〈表达式〉

(3) 输入操作:INPUT 〈变量名 1〉,〈变量名 2〉,〈…〉,〈变量名 n〉
输出操作:PRINT 〈变量名 1〉,〈变量名 2〉,〈…〉,〈变量名 n〉

(4) 注释行是以//开头的文字序列。

(5) 关键词均用大写英文字母表示。

(6) 逻辑运算符:&& 表示‘与’运算,|| 表示‘或’运算。(注:对于 a&&b,当 a 为‘假’时不再对 b 求值。对于 a||b,当 a 为‘真’时不再对 b 求值。)

此外,用伪码还能进行数据说明及函数调用等操作的描述,有关规定将在后续的相关内容

中介绍。

下面是一个用伪码描述算法的实例。

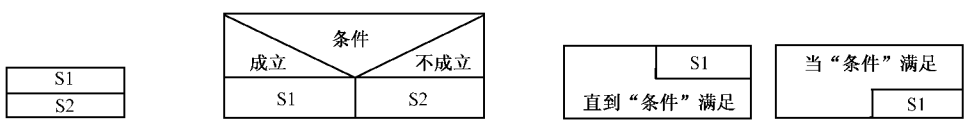
【例 1.4】 用伪码描述求一元二次方程 $ax^2+bx+c=0(a\neq 0)$ 全部根的算法。

```
BEGIN
    INPUT  a,b,c
    d = b^2 - 4ac
    IF  d > 0  THEN
        x1 = (-b + √d) / 2a
        x2 = (-b - √d) / 2a
    ENDIF
    IF  d < 0  THEN
        x1 = -b/2a + (√-d)/2a i
        x2 = -b/2a - (√-d)/2a i
    ELSE
        x1 = x2 = -b/2a
    ENDIF
    PRINT  x1,x2
END
```

由例 1.4 可以看出,使用伪码能把算法描述得十分具体,且更贴近于计算机程序设计语言的形式,有利于由算法向程序的转换。

4. 使用 N-S 图描述算法

N-S 图是由美国两位学者(I. Nassi 和 B. Schneiderman)提出的一种图形描述工具,N-S 就是取自这两个人名字的首字母。N-S 图的主要特点是:完全去掉流程图中的流程线,所有算法的描述只通过 3 种基本控制结构来描述。这 3 种基本控制结构如图 1.2 所示。



(a) 顺序结构　　(b) 选择结构　　(c) 循环结构

图 1.2　N-S 图的 3 种基本控制结构

图中‘S1’或‘S2’既可以是简单功能的操作,如数据赋值、数据的输入或输出等,也可以是 3 种基本控制结构中的 1 种。请看下例。

【例 1.5】 用 N-S 图描述求 3 个正整数中最大值的算法,如图 1.3 所示。

由图 1.3 可以看到,组成一个 N-S 图的基本单元是 3 种基本控制结构,每个基本单元都要根据算法的执行顺序由上至下顺序构成。

1.1.4 简单算法的设计举例

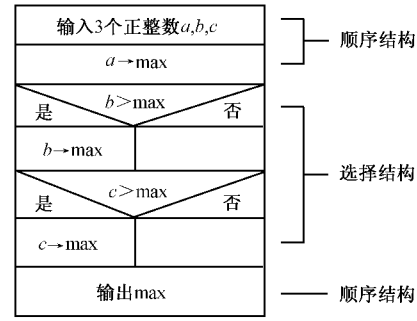


图 1.3 求 3 个正整数中最大值的 N-S 图

前面介绍了算法的基本概念及 4 种算法描述工具的使用。下面再给出一些简单算法的设计实例,通过这些算法设计的练习,熟练掌握算法设计的基本功。

【例 1.6】 输入一个公元年份,判断它是否为闰年。

问题分析 闰年的条件如下。

① 能被 4 整除,但不能被 100 整除。

② 能被 100 整除且能被 400 整除。

(1) 用伪码描述算法:

```
BEGIN
    INPUT   year        //year 表示公元年份
    IF   year 能被 4 整除但不能被 100 整除   THEN
        PRINT   year 是闰年
    ELSE
        IF   year 能同时被 100 和 400 整除   THEN
            PRINT   year 是闰年
        ELSE
            PRINT   year 不是闰年
        ENDIF
    ENDIF
END
```

(2) 用流程图描述算法 如图 1.4 所示。

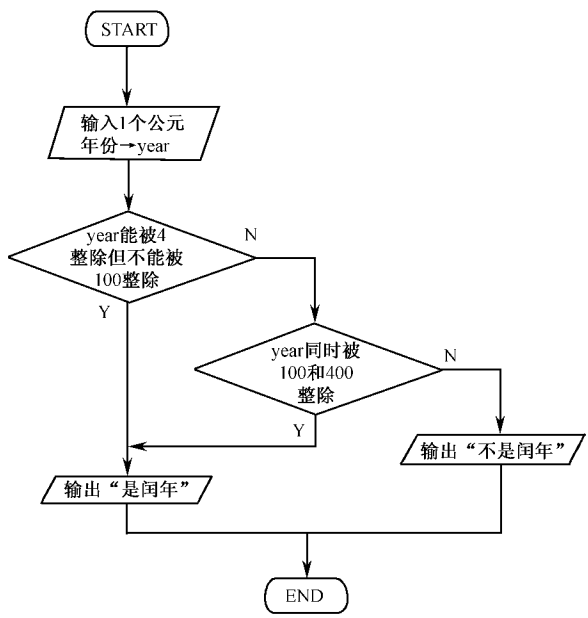


图 1.4 判断闰年的流程图

【例 1.7】 使用以下公式计算 π 的近似值(取 $n=500$):

$$\frac{\pi^2}{6} = \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots + \frac{1}{n^2}$$

设：pi 表示 π 的近似值，num 表示公式右端每项的分母（取平方前），sum 表示公式右端的累加之和。

（1）用自然语言描述算法：

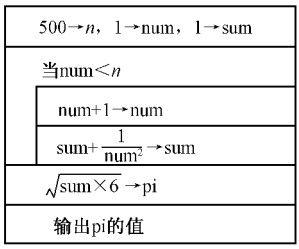


图 1.5　计算 π 近似值的 N-S 图

① 500→n；

② 1→num，1→sum；

③ num + 1→num；

④ sum + $\frac{1}{\text{num}^2}$ →sum；

⑤ 当 num < n 时：返回③，重复执行③～⑤；

⑥ 当 num ≥ n 时：$\sqrt{\text{sum} \times 6}$→pi；

⑦ 输出 pi 的值。

（2）用 N-S 图描述算法：如图 1.5 所示。

【例 1.8】　由键盘输入若干字符，遇到‘．’结束输入，并统计其中的数字字符个数（注：数字字符是指‘0’～‘9’中的某个字符）。

设：ch 表示所输入的字符，count 表示待统计的数字字符个数。

（1）用伪码描述算法：

```
BEGIN
    count = 0
    INPUT  ch
    WHILE  ch 不是‘．’ DO
           IF  ch 是数字字符  THEN
           count = count + 1
           ENDIF
           INPUT  ch
    ENDDO
    PRINT  count
END
```

（2）用流程图描述算法：如图 1.6 所示。

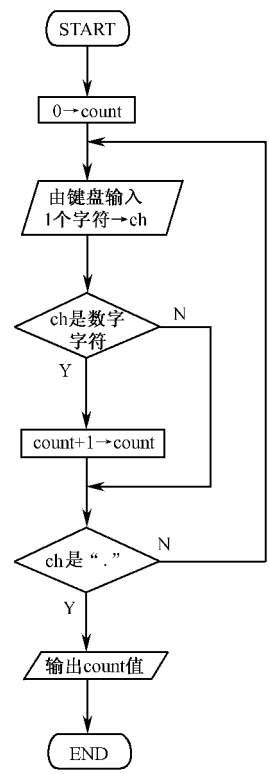


图 1.6　统计数字字符个数的流程图

【例 1.9】　在 n 个不相等的实数中找出最小的 1 个数。

问题分析：这是一个求极值的问题。要想求出最小值，就要把 n 个数逐一进行比较。既然要比较就得有一个进行比较的基准，这里我们用第 1 个数作为基准（当然第 1 个数不一定是最小值）。在比较的过程中，一旦输入的数小于当前的比较基准，就得更换基准值，以保持基准值是当前最小的数。具体步骤是：输入 1 个实数，假设它是最小值，放到 min 中；再输入 1 个实数，与 min 比较，较小者放到 min 中，取代 min 中原来的数；然后输入第 3 个数，……；重复上述过程，直到输入的第 n 个数与 min 作比较（仍保持较小数放入 min 中）后，min 中的值就是 n 个数中的最小值。

设：用 i 表示比较过程中所输入的数值个数，用 temp 存放每次读

入的数值。

(1)用伪码描述算法：

```
BEGIN
    INPUT   n   //n 表示数值个数
    i = 1
    INPUT   第 i 个数
    min = 第 i 个数
    WHILE   i < n   DO
            i = i + 1
            INPUT   第 i 个数
            temp = 第 i 个数
            IF   temp < min   THEN
                 min = temp
            ENDIF
    ENDDO
    PRINT   min
END
```

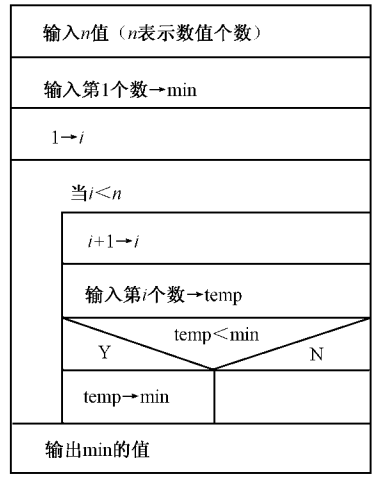


图 1.7　求 n 个不相等的实数中最小值的 N-S 图

(2)用 N-S 图描述算法 如图 1.7 所示。

【例 1.10】　比较两个人的年龄大小。

问题分析：比较年龄大小时，应先比较出生的年份，如果为同年出生则还要比较出生的月份，如果为同年同月出生则还要比较出生的日期。

设：p1、p2 分别代表两个人的年龄，y1、m1、d1 为 p1 的出生年、月、日，y2、m2、d2 为 p2 的出生年、月、日。

(1)用自然语言描述算法：

① 输入 y1，m1，d1，y2，m2，d2 的值；

② 若 y1≠y2：当 y1 < y2 时，输出 p1 > p2；当 y1 > y2 时，输出 p1 < p2；

③ 若 y1 = y2 且 m1≠m2：当 m1 < m2 时，输出 p1 > p2；当 m1 > m2 时，输出 p1 < p2；

④ 若 y1 = y2 且 m1 = m2 且 d1≠d2：当 d1 < d2 时，输出 p1 > p2；当 d1 > d2 时，输出 p1 < p2；

⑤ 若 y1 = y2 且 m1 = m2 且 d1 = d2：输出 p1 = p2。

(2)用 N-S 图描述算法 如图 1.8 所示。

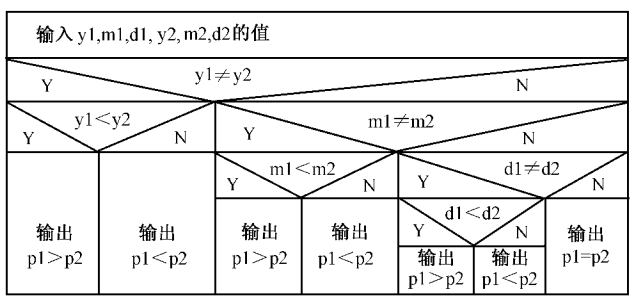


图 1.8　比较两个人年龄大小的 N-S 图

【例 1.11】　输入 3 个不相等的实数 a，b，c，将它们按照从大至小的顺序输出(输出时，a

表示最大值；b 表示中间值；c 表示最小值）。

问题分析 这是一个 3 个数排序的问题。先把 a 与 b 作比较，若 $a<b$，则 a 和 b 交换；再把 a 与 c 作比较，若 $a<c$，则 a 和 c 交换，这时 a 的值即为 3 个数中的最大值。再比较 b 与 c，若 $b<c$，则 b 和 c 交换，这时 b 的值即为 3 个数中的中间值；c 的值为 3 个数中的最小值。

（1）用伪码描述算法：

```
BEGIN
    INPUT  a,b,c
    IF  a<b  THEN
            a 和 b 交换
    ELSE
        IF  a<c  THEN
            a 和 c 交换
        ENDIF
    ENDIF
    IF  b<c  THEN
            b 和 c 交换
    ENDIF
    PRINT  a,b,c
END
```

（2）用 N-S 图描述算法 如图 1.9 所示。

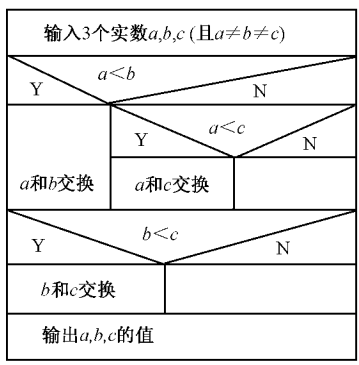


图 1.9　3 个数排序的 N-S 图

以上例 1.6 ~ 例 1.11 中都只使用了一种文字描述工具和图形描述工具，读者可尝试用其他工具完成这些例题的算法描述。这里还要说明一点，例 1.6 ~ 例 1.11 中的算法虽然是正确的但并非都是最佳的。例如，在例 1.6 的算法中，就存在着重复计算的问题。在第 1 个选择结构中，为了判断条件“ year 能被 4 整除但不能被 100 整除 ”是否满足，显然要将 year 被 4 除且被 100 除；而在第 2 个选择结构中，为了判断条件“ year 能同时被 100 和 400 整除 ”是否满足，又要将 year 被 100 和 400 除，这就造成 year 被 100 除了两次。为了避免这个重复计算的问题，可以修改算法如图 1.10 所示。

在上面的算法设计举例中，我们使用了不同的算法描述工具。一般来说，由于自然语言是人们日常生活中的用语，故用自然语言描述算法易于表达和理解，但是往往存在着算法不够简洁、表述不够严格、易产生理解上的“ 多义性 ”等问题。用伪码描述算法时，可根据实际问题的