```
20 REM Data,1987.9
30 REM Topic,The shortest path of a maze
40 REM Function,Find out a shortest path
                of the maze if it is present
100 DEFINT I-N,F,Q,X,Y
110 INPUT "Size of maze--m,n";M,N
120 GOSUB 1000 'Product a maze
130 IF MAZE(1,1)+MAZE(M,N)=0 THEN
      GOSUB 2000:GOSUB 3000
    ELSE      "There is no
199 END
1000 REM .....Product a maze
1010 REM Arrays,MAZE--Maze array
1020 D
1030 E
1040 F
1050 E
1060 F
1070
1080
1090
1100
1110
1120
1130 NEXT I
1140 PRINT "    ";:FOR I=1 TO N:PRINT "---";:NEXT I:PRINT
1199 RETURN
2000 REM .....Subroutine for searching
2010 REM Arrays,Q-----Queue for searching with tactics of BFS
                MARK--Marks array of checked points
2020 REM Variables,H-----Head pointer of queue
                   T-----Tail pointer of queue
                   FLAG--Mark,FLAG=1 if the shortest path is found
2030 DIM Q(M*N-1,2),MARK(M+1,N+1)
2040 T=1:H=1:FLAG=0
2050 Q(1,0)=-1:Q(1,1)=1:Q(1,2)=1:T=T+1
2060 MARK(1,1)=1
2070 WHILE T>H AND FLAG=0
2080   I=1
2090   WHILE I<=8 AND FLAG=0
2110   READ MOVX,MOVY
2120     X=Q(H,1)+MOVX
2130     Y=Q(H,2)+MOVY
2140     IF X<1 OR X>M OR Y<1 OR Y>N THEN I=I+1
         ELSE IF X=M AND Y=N THEN FLAG=1
              ELSE GOSUB 2500:I=I+1
2150   WEND
2160   H=H+1
2170   RESTORE
2180 WEND
2199 RETURN
2500 REM ******
2510 IF MAZE(X,Y)+MARK(X,Y)=0 THEN
       GOSUB 2600
2599 RETURN
2600 REM ******
2610 Q(T,1)=X
2620 Q(T,2)=Y
2630 Q(T,0)=H
2640 MARK(X,Y)=1
2650 T=T+1
2699 RETURN
3000 REM .....Printing
3010 IF FLAG=0 THEN PRINT "There is no path.":RETURN
3020 PRINT "The shortest path is,"
3030 PRINT "(";M;",";N;")";
3035 H=H-1
3040 WHILE H>=1
3050   PRINT "<==(";Q(H,1);",";Q(H,2);")";
3060   H=Q(H,0)
3070 WEND
3099 RETURN
3100 REM .....Data block
3110 DATA 0,1,1,1,1,0,1,-1,0,-1,-1,1,-1,0,-1,1
3199 REM .....Data end
```

```
Size of maze--m,n? 5,9
The maze is,
     1   2   3   4   5   6   7   8   9
   ---------------------------------------
1    0   0   0   1   0   0   0   0   0
2    1   0   0   0   1   0   0   0   1
3    0   0   0   0   0   0   1   1   0
4    0   0   0   0   0   0   0   1   1
5    0   0   0   0   0   0   1   0   0
   ---------------------------------------
The shortest path is,
( 5 , 9 )<==( 5 , 8 )<==( 4 , 7 )<==( 3 , 6 )<==( 3 , 5 )<==( 2 ,
```

BASIC 程序设计教程

谭浩强 张基温 编著

高等教育出版社

高等学校试用教材

BASIC程序设计教程

谭浩强　张基温 编著

高等教育出版社

内 容 提 要

本书是一本介绍结构化程序设计的教材。本书的重点不放在BASIC语言语法上，而是按照“算法＋数据结构＝程序”和结构化程序设计为中心，引导读者掌握规范化的程序设计方法。主要内容包括，算法、计算机与程序，BASIC程序设计初步，BASIC程序的基本数据结构，模块化程序设计，输入与输出设计，数据文卷，动态数据结构，算法设计举例，True BASIC程序设计。

本书文字叙述通俗生动、阐述方式深入浅出。书中讲授的内容具有很强的应用价值，使读者能够容易地将所学的数据结构知识、典型的算法设计方法及结构化程序设计方法有机地结合起来并应用到实际问题中，提高解决实际问题的能力，同时也增加了读者的学习兴趣。书中介绍的程序算法具有普遍性，它们很容易用其它语言实现。书中还给出了丰富的例题与习题。

本书的编写体系考虑到不同层次的需要，可作为各类高等学校和各种计算机学习班的教材，也可供自学。

高等学校试用教材

BASIC程序设计教程

谭浩强　张基温　编著

高等教育出版社出版

新华书店北京发行所发行

北京顺义县印刷厂印装

开本787×1092　1/16　印张 21　　字数480 000

1988年10月第1版　1988年10月第1次印刷

印数0001— 23,130

ISBN7-04-001635-4/TP·34

定价4.75元

前　言

随着计算机科学技术日益深入人心，计算机基础教育的重心，也相应由计算机语言转向程序设计。

算法是程序的精髓。要设计出好的程序，必须先设计出好的算法。本书以算法设计贯穿始终．列举了大量典型算法，并介绍了算法设计方法。通过对典型算法的了解，读者不仅可以掌握算法设计的基本方法，使读者遇到实际问题时，能很快理出一条正轨的解题思路，不致于束手无措、盲目乱凑，而且还丰富了读者的“算法库”。

程序的处理对象是数据。数据结构的形式，直接或间接地影响着算法的复杂性和解题效果。因此在本书中也较系统地介绍了数据结构知识，并把数据结构的设计看作程序设计的重要内容之一。

决定程序质量的另一关键是程序的风格和程序设计者所采用的设计方法。结构化程序设计方法是近年来在国际上兴起的一种新的程序设计方法。它要求程序设计者必须遵循一定的规范，而不能随心所欲、无规则地编写程序。这样使得程序质量不再主要取决于个人的素质与兴趣；程序也不再是一种“艺术品”，而是可以用工程方法处理的规范化的产品。

本书把算法、数据结构、结构化程序设计方法三者有机地融合在一起。也就是说：程序＝算法＋数据结构＋结构化方法。

离开计算机语言，就谈不上程序设计。本书中的程序都用BASIC语言来描述。它是一种简单易学、应用最为广泛的程序设计语言。为此，我们使用了一定的篇幅来介绍BASIC语言的基本知识。语言的介绍．仅仅是为了使读者掌握一种工具并对程序设计有更生动而真实的理解。本书并不把重点放在各种语句及语法规则的介绍上，以免喧宾夺主。读者需要更进一步地掌握语言知识，是很容易找到一本参考手册的，需说明的是，BASIC语言具有“方言性”。本书以在长城0520，IBM-PC，王安PC，APPLE—Ⅱ（加Z80插件，并在CP/M支持下），ALTOS等多种微型计算机上都可运行的Microsoft BASIC为蓝本，也兼顾了一般BASIC的特点，以便大多数读者都有实践的机会。

有人可能会问，当今像Pascal，Ada这样一些优秀的计算机语言已经问世，为什么还要选用BASIC语言作为本书中描述程序的工具呢？诚然，Pascal，Ada语言是一些学者所推崇的计算机语言，它们是按照最新的观念设计的。然而，BASIC语言毕竟是一种应用最为广泛的并已解决了大量实际问题的计算机语言。英国学者M．詹姆斯说得好：“忽略了这一事实，将使我们回到与中世纪类似的一种情形中。那时候，上流社会喜欢用书面拉丁语进行社交，绝大多数人是用口头英语表达思想，后来，英语得到了发展，而拉丁语却僵化了。”当然，我们不应当用一种语言排斥另一种语言，每种语言都有它的应用领域。我们认为，对初学者来说，BASIC有它的优势。事实上，能得到广泛应用本身，就足以说明BASIC语言也具有优于其它语言的诱人之处，而且面对那些熟悉早期的BASIC版本的人们的责难，BASIC已得到发

展和完善。特别是True BASIC的出现使BASIC并不逊色于PASCAL和其它语言，而它的绘图等功能却是其它一些语言所望尘莫及的。

考虑到读者的基础不同，并为满足不同层次的教学需要，本书采取了新的体系结构。书中内容，从第一章起，每增加一章都可构成一个独立体系，形成了可供各类读者自学或教学使用的多层次结构。具体说来，第一章到第四章，是最基本的部分，中级学习者则可以学习第一到第六章；第七章、第八章是提高部分，可供读者进一步深入学习使用；第九章介绍了BASIC语言的最新版本——TrueBASIC。在作教材使用时，如果学时较少，可以学习前六章为主，七、八章可供选修或将部分内容穿插到有关章节中介绍。对理工科各专业，如学时数够的话，建议最好学完第七、八两章，使读者在学完本课程后具有较强的程序设计能力。在计算机技术日益普及的今天，我们认为适当增加本课程的内容、提高本课程的深度是适宜的，也是可能的。对各章中一些内容较深的部分、在目录中加了“*”号、初次学习时可以不学、留待读者今后参考。

本书每一章后面都附有一定数量的习题，可以根据不同程度读者的需要选做其中的一部分。其中有一些习题难度较大，可供程度较高的读者选做或思考。

本书是根据全国高等学校计算机基础教育研究会和全国非计算机专业教材评审组多次会议上许多同志提出的建议而编写的。在正式出版前，曾作为内部交流教材，提供给部分高校使用以征询意见。许多单位的同志给予了热情的支持与鼓励，并对本书提出了不少宝贵的意见。本书出版前，清华大学计算机系吴文虎副教授审阅了该书稿。在此对曾经关心、支持和帮助过本书出版的同志致以诚挚的谢意。

本书作为深化计算机基础教育的一种尝试，定有不妥之处，乞请指教。

谭浩强　张基温

1987．10

目　录

第一章　算法、计算机与程序

本书是介绍怎样利用计算机去解决实际问题的。由计算机处理的问题分为两大类：一类是“数值运算”，如解一个联立方程，求一个函数的定积分等；另一类是“非数值运算”，例如，有一批人名要按它们的汉语拼音字母顺序排序，用计算机去检索图书馆中的书刊等。过去有的人认为计算机的作用就是进行数值计算，计算出某一个数值结果。这是一个误解。当今，计算机已深入到人类生产和生活的各个领域，在非数值运算领域的应用远远超过了数值运算。

我们所使用的计算机是靠人们给它一定的指令才能工作的，它自己是不会思考的（至少目前的计算机是这样）。计算机不过是一种工具，人们需要了解怎样使用这种工具，才能有效地使用它。众所周知，如果使用电子计算器进行87＋36的加法运算，必须先按健“8”、“7”，然后按“＋”，再按“3”、“6”，最后按“＝”，这样就可以得到结果123。也就是说，必须按一定的步骤进行操作，才能得到预期的结果。

§1.1 算　　法

1.1.1 算法的概念

计算机也称“电脑”，它的解题方法和人们思考问题、处理问题的方法有许多类似之处。我们先回顾一下人们是怎样思考问题和解决问题的。

无论进行数值运算或非数值运算，首先我们都需要事先整理好思路，设计好解题的步骤。在智力竞赛中，对于“1＋2＋…＋100”这样一个题目，有的人按原始的方法先1＋2，再加3，再加4，……；而有的人则把它变成这样的运算：100＋(1＋99)＋(2＋98)＋…＋(49＋51)＋50很快得到5050。显然后一种方法巧妙、速度快。为解决一个问题而采取的方法和步骤，通常称之为算法。或者说，算法是解题方法的精确描述。为了有效地进行运算，应当选择合适的算法。

不仅处理数值计算时需要考虑算法，处理非数值运算更有算法问题，例如有一百个无规则排列的数，要把它们按大小顺序排列，就有许多不同的方法（用不同的方法实现排序，效率差别很大）。

广义地说，处理任何事情，都有一定的步骤，也都有各自的算法。我们完成任何一件事情都是在实现一个算法，只不过在做那些极为熟悉的工作时，由于我们已经习惯了，而无须有意识地考虑它罢了。歌手唱歌，乐队伴奏，都是按乐谱进行的，乐谱规定了演唱和演奏的规则和各音符的顺序，可以把乐谱看作演奏过程的算法。同样，太极拳的动作图解，做菜的菜谱，珠算的口诀，工作计划，生产流程，……等等，都可以分别看作是一个算法。著名计算机科学家D.E.Knuth说：“一个算法是一个有穷规则的集合，其中的规则规定了一个解决某一特定类型问题的解答”。因此不要以为只有数值计算才有算法。当然我们在本书中所涉及的只是计算

机算法，即由计算机可以实现的操作所构成的题解步骤。

尽管我们所要求解的问题成千上万，各有自己的算法，但是我们在研究算法的时候总是将它们进行分类，找出各类问题的共同性规律。如前面提到的求$1+2+\cdots+100$的问题，假如出题者把题目改为求$1+2+\cdots+1000$，应试者决不会重新冥思苦想另找算法，而仍然会用$1000+(1+999)+(2+998)+\cdots+(499+501)+500$的方法求得结果为500500。因为这是同一类的问题。我们在研究算法时往往要从具体问题提炼出共同的特点，然后对这些特点进行研究、归纳，找出解决这些问题的规律。这种方法就叫做问题的抽象。例如我们要求5！（即$1\times2\times3\times4\times5$），7！（即$1\times2\times3\times4\times5\times6\times7$），9！（即$1\times2\times3\times4\times5\times6\times7\times8\times9$），它们的算法基本上是相同的，只是乘法的次数不同。我们只需研究求n！的算法即可。将9代入n，就得到9！；将1000代入n，就得到1000！。抽象是用于处理复杂问题的一种手段，其目的是为隐藏细节，更好地面向问题、分析问题、构造算法。

下面我们举几个算法的例子：

例1.1.1 求两个正整数m和n的最大公约数（m为大数，n为小数）。这个算法称为欧几里得（Euclid）算法。为使读者易于理解，我们将m和n代入具体数值（$m=24$，$n=16$）。

第一步：以n除m，求$\frac{m}{n}$的整数商及其余数r_1（现$m=24$，$n=16$，有$\frac{m}{n}=\frac{24}{16}=1$余8）。

第二步：判断余数r_1是否等于零？如$r_1=0$，则n就是解（最大公约数）；如果$r_1\neq0$，就执行下面的第三步（今$\frac{24}{16}$的余数为8，不等于零。所以要执行第三步）。

第三步：将n作为被除数，将r_1作为除数，求$\frac{n}{r_1}$的余数r_2，若$r_2=0$，则除数r_1就是最大公约数；若$r_2\neq0$，则执行第四步（今$n=16$，$r_1=8$，$\frac{n}{r_1}=\frac{16}{8}=2$余0，即$r_2=0$）。

第四步：若$r_2\neq0$，则再使r_1为被除数，r_2为除数，求$\frac{r_1}{r_2}$的余数r_3。若$r_3=0$，则除数r_2就是最大公约数；否则再求$\frac{r_2}{r_3}$的余数r_4。……，如此一直做下去，直到某一次的余数$r_n=0$为止，则除数（即它前一次的余数r_{n-1}）就是最大公约数。

这种方法就是辗转相除算法。人工求最大公约数就是这样一次又一次地做除法，直到求出解为止（请读者用上述方法求312和582的最大公约数）。

我们能否用简单明瞭的方法表示出求任意两个正整数m和n的最大公约数呢？设有三个变数：m、n和r，m作为被除数，n作为除数，r作为余数。求最大公约数的步骤可以归纳如下：

第一步（置数）：将两个数中的大数放到m中，小的数放到n中。

第二步（求余）：求$\frac{m}{n}$的余数r。

第三步（判断）：若$r=0$，则n就是最大公约数。若$r\neq0$，执行第四步。

第四步（置换）：使变量m的值为n（或者说，将n的值放到变量m中），使变量n的值为

r（或者说，将r的值送到n中）。返回执行第二步，继续执行。

这种方法使用了循环，步骤简明，是一种一般的抽象的表示方法。对任何的正整数m和n都是适用的。要理解这个算法，或要写出这个算法，需要经过一定的抽象思维，进行提炼和归纳。我们要逐步学会用这种方式表示算法，后面我们会知道，用计算机处理判断和循环是很方便的，因此，在计算机上实现上面这个算法是轻而易举的。

例1.1.2 求最初的m个自然数之和，即求 $\sum_{n=1}^{m} n = 1+2+3+\cdots+(m-1)+m$。

我们在前面说过在人工计算 $\sum_{n=1}^{100} n$ 时，可以采用技巧：$100+(1+99)+\cdots+(49+51)+50$。但是如果我们采用笨的方法，即一个数一个数地相加，一直加到m为止，则怎样用算法来表示这个过程呢？

思路是这样的：用变量sum代表最初的n个自然数之和。未进行累加前，sum等于0。然后使sum加1（这时相当于已实现了 $\sum_{n=1}^{1} n$），然后再使它加2（此时sum已等于1+2，即sum = $\sum_{n=1}^{2} n$），如此一直累加，直到将m加到sum中为止。我们现在用n代表加数，每次使sum+n→sum（使sum的值增加n），为了使加数n由1变到2，再变到3，需要在每次加法运算后使n自动加1。可以将算法表示如下：

① 使sum = 0；

② 使n = 1；

③ 使sum + n→sum（即将sum的原来的值加上n的值去代替sum的原值，例如，sum原值为0，n值为1，则实现sum + n→sum后，sum的新值为1）；

④ 如果n≥m，则输出sum的值。结束。否则，使n + 1→n，然后返回③继续执行。

这也是一个循环算法。请读者用这个算法进行 $\sum_{n=1}^{5} n$ 的运算，以对这个算法有更具体的体会。

例1.1.3 求正整数A和B的积，A和B为任意位数。

我们先看一下人工计算是怎样进行的。假如A = 3412，B = 513，乘法竖式为：

```
      3 4 1 2……A
  +     5 1 3……B
  ---------------
    1 0 2 3 6
    3 4 1 2      | 部分积
  1 7 0 6 0      |
  ---------------
  1 7 5 0 3 5 6 ……积
```

它们是这样进行的：以A为被乘数，分别以B的个位数、十位数、百位数作为乘数，求出它们的部分积。个位数与A相乘，得第一行部分积，十位数与A相乘得到的部分积（第二行）应左移一位，或者认为将该位数字乘以10再和A相乘；同样由乘百位数得到的第三行部分积应左移二

位，或认为将乘数乘以100再和A相乘，把各部分积相加即得最后的积。

现在A和B的位数都是任意的，那么怎样用一般的抽象的方法表示此算法呢？

将乘数B表示为$b_nb_{n-1}\cdots b_2b_1b_0$，$b_0$表示个位的数字，$b_1$表示十位的数字，$b_2$表示百位的数字，其余类推。用C表示某一时刻的部分积之和。用i表示乘数的位数，i=0表示个位（因为$10^0=1$），i=1表示十位（因为$10^1=10$），i=2表示百位（因为$10^2=100$），……。

算法可以表示如下：

① 使i=0，C=0（未进行乘法运算前，部分积之和为0）；

② $C+A\times b_i\times 10^i\rightarrow C$，乘$10^i$即左移i位（第一个部分积为$0+A\times b_0\times 10^0=A\times b_0$）；

③ 若i<n，则使i+1→i（i增加1），然后返回②继续进行；若i≥n则C就是最后的乘积。

请读者按上述算法进行853×28654的运算。

例1.1.4 求两个整数X（X≥0）和Y（Y>0）的整数商和余数（规定只能用加法和减法运算）。

除法运算实际上是用加法和减法来实现的。其办法是，从X中一次又一次地减去Y，直到X<Y为止，所减次数即为商。例如5÷2的商为2，余数为1。可以将5先减2，余3，再减一次2，余1。每减一次，商加1。5能减两次2，所以商就得2。此时余数为1，不能再减2，结束。用Q代表商，R代表余数，算法可以表示为：

① 使Q=0，R=X（在开始减以前，商为零，所余的数就是X本身）。

② 如果R≥Y，则使R-Y→R（余数减去一个Y），Q+1→Q（商加1），并返回重新执行②，直到R<Y，就不再进行上述工作，此时的Q就是需要求的商，R就是需要求的余数。

例1.1.5 用筛法求1到1000之间的全部素数（质数）。

求素数表的方法很多，但是有一种方法却与众不同。它就是“埃拉托色尼(Eratost—henes)筛法”。埃拉托色尼是欧几里德差不多同时代的希腊著名数学家，他求出了世界上第一张素数表。他的办法是：在一张纸上写上1到1000的全部整数。然后进行以下操作：

① 挖掉数1；

② 用下一个未被挖掉的数p除它后面每一个没有被挖掉的数，把能被整除的数挖掉；

③ 看刚才的数p后面还有无未被挖掉的数，如果有，则返回②继续执行；如果没有，就结束，纸上剩余的数就是1到1000之间的全部素数（质数）。

执行上面第①步操作的原因是由于人们已经从数学定义中知道1不是素数，应当去掉。根据素数的定义：如果一个整数除了1和它本身以外不能被其它整数整除，它就是一个素数，所以2是素数，应保留。第一次执行第②步是把2以外的所有偶数去掉。第③步返回②，将3以外的3的倍数去掉。以后逐次将5,7,11,13,17，…的倍数去掉。显然，最后剩下的就是1到1000间的全部素数。如图1.1所示，由于纸被挖得纸象一个筛子一样，所以这种算法被称为“筛法”。筛法的思路很独特，很有意思，至今它还是求素数的一种重要方法。

① 2 3 ④ 5 ⑥ 7 ⑧ ⑨ ⑩ 11 ⑫ 13 ⑭ ⑮ ⑯ 17 ⑱ 19 ⑳ ㉑ ㉒ 23 ㉔ ㉕
㉖ ㉗ ㉘ 29 ㉚ 31 ㉜ ㉝ ㉞ ㉟ ㊱ 37 ㊳ ㊴ ㊵ 41 ㊷ 43 ㊹ ㊺ ㊻ 47 ㊽ ㊾ ㊿
……

图 1.1 埃拉托色尼筛子

从上面几个例子中可以看到：每一个算法都是由最基本的操作组成的。这些操作包括加、减、乘、除、判断、循环等，无论是人还是计算机要实现这些基本的操作都是容易的。任何复杂的问题也都可以由这些最基本的操作按一定顺序组成。研究算法的目的就是要研究怎样把各种类型的问题分解成如上一些基本的操作。

其实，在古代，当数学还没有充分发展起来的时候，人们只会利用算术方法进行计算，不论解决什么问题都要用算术方法去解决，这就创造出许多技巧。例如有名的“鸡兔同笼”问题，既可以用代数方法去解决，也可以用算术方法解决。用代数方法固然简单，但是在代数方法出现之前，人们不得不用最简单的四则运算求解，因而把精力集中放在算法研究上了。可以说算术就是研究算法的一门科学，算术能够解决的数学问题的复杂程度很高，其中包含有很高的技巧。可是，随着数学的发展，人们的注意力和兴趣渐渐转向以公理系统为基础的、内容丰富、结构复杂的演算上了。而算法，这种曾经在人类历史上创造过辉煌业绩的数学思想，逐渐被人冷漠了，遗忘了。

随着自动机器的出现，尤其是电子计算机的出现，人们又开始注意转向算法的研究，使算法获得了新生。任何自动机器（包括计算机）的复杂操作都是由最简单的操作组成的。从本质上说，计算机只能完成一些最简单的基本操作（如加、减、判断、转移……等等），它用极快的速度相继地执行许许多多个简单的操作序列，从而实现了看起来复杂的任务。正如玩魔方一样，巧妙地把简单的操作组织起来，就能达到预期的目的。研究如何利用计算机解题的步骤，就是计算机算法。

从根本上说，计算机科学就是研究算法的科学。五十年代初期，计算机在数值计算方面的应用，使得讨论计算方法一类算法的计算数学迅速发展并成熟。五十年代中后期，计算机开始广泛应用于非数值运算领域。非数值运算的特点是其处理对象大多数是非连续性的数据（例如银行各存户的数据便是不连续的），这种数据称为“离散性”的数据。非数值运算领域的应用促进了以研究离散对象为目标的组合算法的研究。

著名计算机科学家D.E.Knuth说：“对所有计算机程序设计来说，算法的概念总是最基本的。”因此，我们要了解算法，学会设计处理各种问题的算法，并使之在计算机上实现。

1.1.2 算法的特点

一个算法应当具备以下几个特性：

1.有穷性。一个算法应包括有限个操作步骤，或者说它由一个有穷的操作序列组成，而不应该是无限的。例如，二十四式的太极拳或七十二式的太极拳便可以称为太极拳算法。如果有人说他设计一套永远打不完的太极拳的规则，人们将难以接受，不能称为算法。

不仅如此，有穷性还常常理解为实际上能够容忍的合理限度。例如，执行一个计算机算法需要让计算机进行10000年，便是不能容忍的。

2.确定性。算法中的每一步的含义都应当是清楚无误的，不能模棱两可，也就是说不应该存在“二义性”（即可作两种以上不同的解释）。请读者分析下面这句话：张三对李四说他的孩子考上了大学。这句话就是“二义性”的，可以理解为张三的孩子考上了大学，也可理解为李四的孩子考上了大学。例如上面例1.1.5(用筛法求素数表)中，如果第③步第一句话改写成：“后

面还有无未被挖掉的数……”，这就是模棱两可的，是哪个数的后面？因此在表示算法时要使用明确的文字或数学语言。

3. 一个算法应该有零个或多个输入。如例1.1.1（求最大公约数）中，有二个输入量m和n。有的算法也可以没有输入。例如演员唱歌，没有输入，但能不断输出歌声。

4. 一个算法应该有一个或多个输出。算法的目的是求解，“解”就是输出。对无解的问题也应给出“无解”的信息。没有输出的算法是没有意义的。例1.1.1中的输出是最大公约数。

5. 有效性。算法中的每一个操作，都应该是能有效地执行的，执行算法最后应该能够得到确定的结果。如果例1.1.1中给定的n为零，则执行第一步时就会遇到一个无意义的运算，便不是有效的操作。只要算法中有一个操作是不可执行的，整个算法就不具有有效性。

对使用算法的人（而不是设计算法的人）来说，可以把一个算法看作一个“黑箱”，你给它输入某些确定的量，它就会输出确切的输出量。如求m和n最大公约数的算法的“外部特性”可以图1.2表示。也就是说，从算法的外部（而不是研究算法的内部）看，它的作用是：给它输入m和n，它就输出一个最大公约数。从此例也可以理解算法的输入和输出的概念。

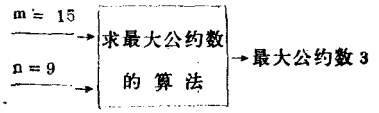

图1.2

§1.2 算法的表示

上一节介绍过，算法是解题方法的精确描述。那么用什么方法来描述算法呢？

在研究计算机算法和进行程序设计过程中，可以用许多工具来描述一个算法。常用的有以下几种。

① 自然语言；

② 流程图；

③ 结构化流程图（N-S图）；

此外还有其它一些工具，如伪代码、IPO图、Warnier图、PAD图、判定表等。

1.2.1 自然语言

用自然语言（英语或汉语）来描述算法比较通俗易懂，例1.1.1到例1.1.5都是用自然语言描述算法的例子。用自然语言描述算法虽然易懂，但它有一定缺点：

（1）自然语言容易有“二义性”，不大严格。例如前面说到的“张三对李四说他的孩子考上了大学”这句话，好懂但不严格，具有二义性。

（2）用自然语言描述比较冗长。例如：“置m的值为n，置n的值为r”，不如表示为“n→m，r→n”简洁。上节例1.1.2，用自然语言描述冗长，但写成下面这种形式就比较简练：

① 0→sum

② 1→n

③ sum+n→sum

④ if n≥m print sum;end

else n+1→n; go to ③

(3) 自然语言的表示方法是顺序的，即一句一句地顺序地叙述下来，或者说各步骤是串行的。如果算法中有分支或转移时，用文字表示就显得不那么直观。例如，例1.1.2中的第④步中有判断（假如…否则…）和转移（返回③继续执行），用自然语言描述不如上面那样用if…else和go to那么明白。

因此，在程序设计过程中一般不采用自然语言来描述算法。只有在不致引起混淆的情况下，才可用简明的文字来描述某一步骤的操作。

1.2.2 流程图

流程图(flow chart)是一种图形描述工具。它用一些几何图框表示各种类型的操作。在框内写上简明的文字或符号表示具体的操作，用箭头的流程表示操作的先后顺序。图1.3为ANSI(美国国家标准化协会)规定的一些常用的流程图符号。

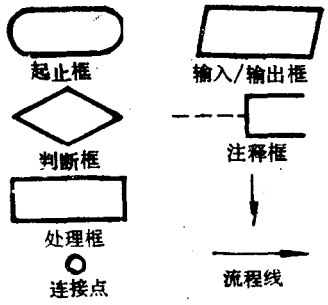

图 1.3

例1.2.1 用流程图表示例1.1.1（求最大公约数）的算法。见图1.4。图中用菱形框判断r = 0？如果r = 0，则输出n，结束；如果r≠0，则还要继续下去：使n→m，r→n，然后再求$\frac{m}{n}$的余数r。

从图1.4可以看到，用流程图表示算法比较直观形象，先后次序明确，不会产生用自然语言描述算法时可能出现的“二义性”。目前许多书刊中多用这种流程图来描述算法。

例1.2.2 用流程图表示例1.1.2（求$\sum_{n=1}^{m} n$）的算法（见图1.5）。

例1.2.3 求a + |b|的算法的流程图。见图1.6，即先计算a和b的值，接着判断b的符号，

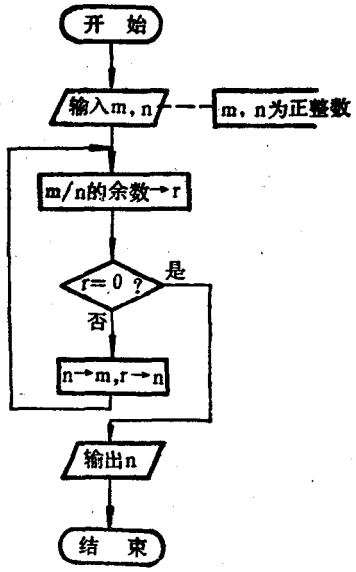

图 1.4

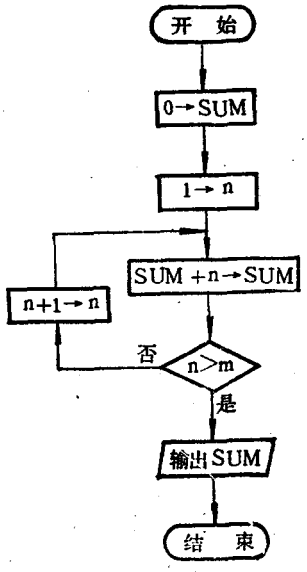

图 1.5

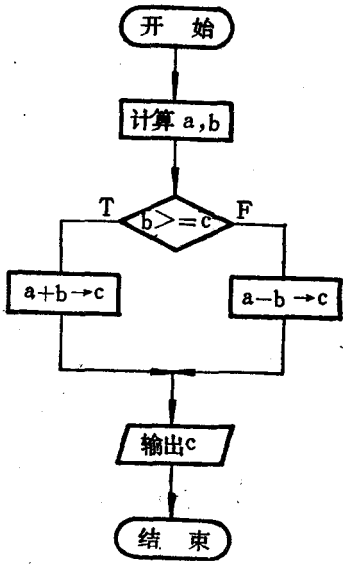

图 1.6

如b为负，则应执行a－b。

例1.2.4 有两个瓶子分别是装醋和桔子水的，以V（Vinegar）代表醋瓶，以O(Orangeade) 表示装桔子水的瓶子。如果买东西时把二者装反了，要求把它们换过来。设计换瓶的算法。

这个算法的关键是：必须用第三个瓶子M，作为中间过渡，才能实现互换。算法可以表示如下：

① 将V瓶中的桔子水倒入M瓶中，即V→M；

② 将O瓶中的醋倒入V瓶中，即O→V；

③ 将M瓶中的桔子水倒入O瓶中，即M→O。

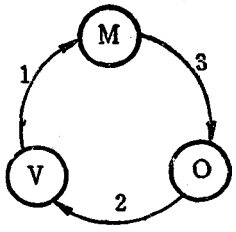

图 1.7

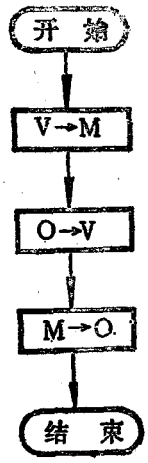

图 1.8

这样就实现了二瓶互换的目的。

上述操作可以用图1.7示意。图中的"1","2","3"表示先后顺序。

图1.8为这个算法的流程图。

这个算法很简单，却是很重要的。在计算机中实现两个变量互换就是用这种方法。如果把算法写成下面这样就错了：

① V→O，即将V瓶中的东西倒入O瓶中；

② O→V，即将O瓶中的东西倒入V瓶中。

流程图也不能画成图1.9那样。

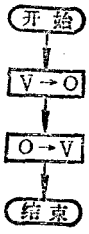

图 1.9

例1.2.5 有一批正数a_1，a_2，…，a_n，找出其中最小的一个数。

解决这个问题的思路是：像打擂台一样，先让第一个人在台上，由第二个人和他比，谁赢谁就留在台上；然后第三个人和台上的人比，谁赢谁就留在台上。如此继续下去，直到比完为止。

今用一个变量MIN代表最小数，先让MIN＝a_1；然后将MIN依次和a_2，…，a_n比较，若a_i＜MIN就让a_i→MIN；再让MIN与a_{i+1}比较，……，直到i＝n为止。

这个算法也可以描述为以下形式：

① ai→MIN，1→i

② i+1→i

③ if a_i＜MIN then a_i→MIN

④ if i≥n 打印输出MIN,end;else go to ②

流程图如图1.10示。

通过以上几个例子可以初步了解到怎样用流程图表示算法。应当说明，流程图是画给人看的（而不是输入给计算机的），只要逻辑正确，自己和别人能看懂就可以了。一般要由上而下地按执行顺序画下来。矩形框有一个入口，一个出口；菱形框有一个入口，二个出口。要注意

带箭头的流程线指向什么地方，千万不能画错，否则逻辑关系就错了。例如图1.10的第一个矩形框“a_1→MIN，1→i”，也可以画成两个矩形框，分别为“a_1→MIN”和“1→i”，作用是相同的。因为这两个框是顺序执行的。但是不能把图1.10中第一、二个矩形框合并为一个框（见图1.11），因为那样的话，算法的内容就变了。请读者自己分析比较图1.10和图1.11执行结果有何不同？按图1.11的流程能否得到最小的数？从此例可以看出流程线的重要性。流程线不能忘画箭头，否则不知道从什么地方到什么地方，先执行那个框，后执行哪个框。

框内的文字符号要简单明确、一目了然，不能有二义性。

图中的“a_1→MIN”还可以用其它多种方法表示，如“使MIN值为a_1”，“将a_1送到MIN中”，“MIN = a_1”或“MIN←a_1”等，都可以，作用相同。

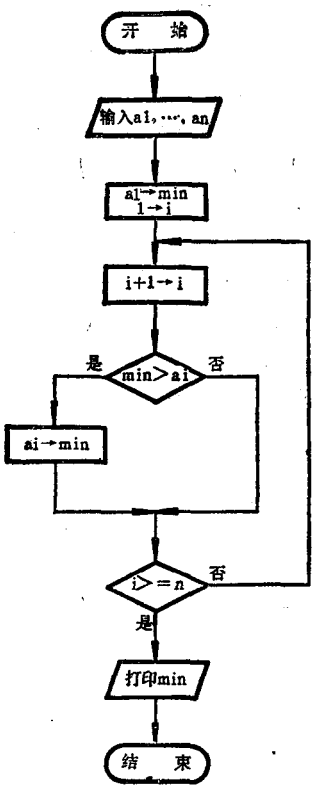

图 1.10

1.2.3 三种基本结构

用流程图表示算法方便、直观。画图方法比较自由灵活，但是它存在着一个致命的弊病：由于流程图要用带箭头的流程线指明执行的先后顺序，对流程线的使用也无任何限制，即允许从某一处用流程线将流程转向流程图中的任何一个框的入口处，这样看起来方便，但如果随心所欲不加任何限制地在流程图中用流程线使流程无规则地转来转去，会使流程图变得象乱麻一样，使人难以理解。或者说，这种算法结构没有规律，清晰度差，使人难以阅读。图1.12是这种算法结构的示意图，其中每一根横线代表一个框的功能。由于这种算法结构像“意大利面条”一样（象一碗面条一样互相缠绕在一起难以找出头和尾），故被称之为BS型(Bowl of Spaghetti，面条型），也有人称其为耗子窝（rat's nest）型。

显然，这种BS型结构不利于阅读和交流，也不便于修改，从而使算法的可靠性和可维护性难以保证。为了提高算法的质量，必须限制箭头的滥用。人们设想，如果能规定几种结构的基本单元，由这些基本结构单元顺序地组成一个算法结构，整个算法的结构就是自上而下地串起来的

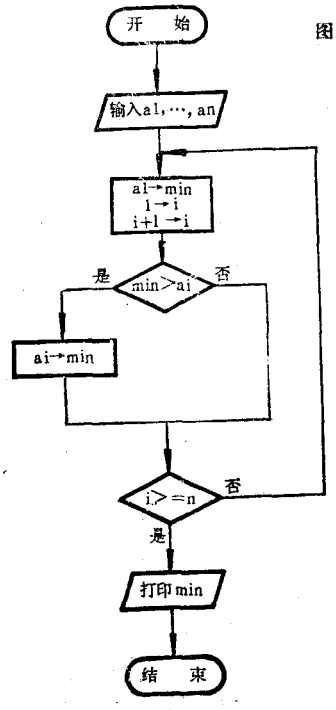

图 1.11