第 7 章　算法初步

7.1　算法与程序框图

7.1.1　算法的概念

算法不仅是数学及其应用的重要组成部分，也是计算机科学的重要基础．在现代社会里，计算机已经成为人们日常生活和工作不可缺少的工具．听音乐、看电影、玩游戏、打字、画卡通画、处理数据，计算机几乎渗透到了人们生活的所有领域．那么，计算机是怎样工作的呢？要想弄清楚这个问题，算法的学习是一个开始．同时，算法有利于发展有条理的思考与表达的能力，提高逻辑思维能力．

在以前的学习中，虽然没有出现算法这个名词，但实际上在数学教学中已经渗透了大量的算法思想，如四则运算的过程、求解方程的步骤等，完成这些工作都需要一系列程序化的步骤，这就是算法的思想．下面举例说明．

例 1：写出你在家里烧开水过程的一个算法．

解：第一步：把水注入电锅．

第二步：打开电源把水烧开．

第三步：把烧开的水注入热水瓶．

（以上算法是解决某一问题的程序或步骤）

例 2：给出求 1+2+3+4+5 的一个算法．

解：算法 1：按照逐一相加的程序进行．

第一步：计算 1+2，得到 3．

第二步：将第一步中的运算结果 3 与 3 相加，得到 6．

第三步：将第二步中的运算结果 6 与 4 相加，得到 10．

第四步：将第三步中的运算结果 10 与 5 相加，得到 15．

算法 2：可以运用公式 $1+2+3+\cdots+n=\dfrac{n(n+1)}{2}$ 直接计算．

第一步：取 $n=5$．

第二步：计算 $\dfrac{n(n+1)}{2}$．

第三步：输出运算结果．

（说明算法不唯一）

例 3：解二元一次方程组的步骤.

（可推广到解一般的二元一次方程组，说明算法的普遍性）

例 4：用“待定系数法”求圆的方程的大致步骤.

第一步：根据题意，选择标准方程或一般方程.

第二步：根据条件列出关于 a、b、r 或 D、E、F 的方程组.

第三步：解出 a、b、r 或 D、E、F，代入标准方程或一般方程.

算法的概念

通过对以上几个问题的分析，我们对算法有了一个初步的了解．在解决某些问题时，需要设计出一系列可操作或可计算的步骤，广义地说，算法就是做某一件事的步骤或程序．菜谱是做菜肴的算法，洗衣机的使用说明书是操作洗衣机的算法，歌谱是一首歌曲的算法．在数学中，现代意义上的“算法”通常是指可以用计算机来解决的某一类问题的程序或步骤，这些程序或步骤必须是明确和有效的，而且能够在有限步之内完成.

例题分析

例 1：任意给定一个大于 1 的整数 n，试设计一个程序或步骤对 n 是否为质数做出判定.

算法分析：根据质数的定义，很容易设计出下面的步骤：

第一步：判断 n 是否等于 2，若 n=2，则 n 是质数；若 n>2，则执行第二步.

第二步：依次从 2 至 n-1 检验是不是 n 的因数，即整除 n 的数，若有这样的数，则 n 不是质数；若没有这样的数，则 n 是质数.

这是判断一个大于 1 的整数 n 是否为质数的最基本算法.

例 2：用二分法设计一个求方程 $x^2-2=0$ 的近似根的算法.

算法分析：回顾二分法解方程的过程，并假设所求近似根与准确解的差的绝对值不超过 0.005，则不难设计出以下步骤：

第一步：令 $f(x)=x^2-2$．因为 $f(1)<0$，$f(2)>0$，所以设 $x_1=1$，$x_2=2$.

第二步：令 $m=(x_1+x_2)/2$，判断 $f(m)$ 是否为 0，若是，则 m 为所求；若否，则继续判断 $f(x_1)\cdot f(m)$ 大于 0 还是小于 0.

第三步：若 $f(x_1)\cdot f(m)>0$，则令 $x_1=m$；否则，令 $x_2=m$.

第四步：判断 $|x_1-x_2|<0.005$ 是否成立．若是，则 x_1、x_2 之间的任意取值均为满足条件的近似根；若否，则返回第二步.

小结：算法具有以下特性：有穷性、确定性、顺序性、不唯一性、普遍性.

例 3：写出解二元一次方程组 $\begin{cases} x-2y=-1 & ① \\ 2x+y=1 & ② \end{cases}$

解：第一步：②-①×2 得 5y=3．③

第二步：解③得 y=3/5.

第三步：将 y=3/5 代入①，得 x=1/5.

学生做一做：对于一般的二元一次方程组来说，上述步骤应该怎样进一步完善？

老师评一评：本题的算法是由加减消元法求解的，这个算法也适合一般的二元一次方程组的解法．下面写出求方程组 $\begin{cases} A_1x+B_1y+C_1=0 \\ A_2x+B_2y+C_2=0 \end{cases}$（$A_1B_2-A_2B_1\neq 0$）的解的算法．

第一步：②×A_1-①×A_2，得 $(A_1B_2-A_2B_1)y+A_1C_2-A_2C_1=0$．③

第二步：解③得 $y=\dfrac{A_2C_1-A_2C_2}{A_1B_2-A_2B_1}$．

第三步：将 $y=\dfrac{A_2C_1-A_2C_2}{A_1B_2-A_2B_1}$ 代入①，得 $x=\dfrac{-B_2C_1+B_1C_2}{A_1B_2-A_2B_1}$．

此时我们得到了二元一次方程组的求解公式，利用此公式可以得到另一个算法．

第一步：取 $A_1=1$，$B_1=-2$，$C_1=1$，$A_2=2$，$B_2=1$，$C_2=-1$．

第二步：计算 $x=\dfrac{-B_2C_1+B_1C_2}{A_1B_2-A_2B_1}$ 与 $y=\dfrac{A_2C_1-A_2C_2}{A_1B_2-A_2B_1}$．

第三步：输出运算结果．

可见利用上述算法更加有利于上机执行与操作．

例 4：写出一个求有限整数列中的最大值的算法．

解：算法如下：

第一步：假定序列中的第一个整数为“最大值”．

第二步：将序列中的下一个整数值与“最大值”比较，如果它大于此“最大值”，这时就假定“最大值”是这个整数．

第三步：如果序列中还有其他整数，重复第二步．

第四步：在序列中一直到没有可比的数为止，这时假定的“最大值”就是这个序列中的最大值．

学生做一做：写出对任意 3 个整数 a、b、c 求出最大值的算法．

老师评一评：在例 2 中我们是用自然语言来描述算法的，下面我们用数学语言来描述本题的算法．

第一步：max=a

第二步：如果 b>max，则 max=b．

第三步：如果 c>max，则 max=c．

第四步：max 就是 a、b、c 中的最大值．

例 5：写出求 1+2+3+4+5+6 的一个算法．

分析：可以按逐一相加的程序进行，也可以利用公式 1+2+…+n=$\dfrac{n(n+1)}{2}$ 进行，也可以根据加法运算律简化运算过程．

解：算法 1：

第一步：计算 1+2 得到 3.

第二步：将第一步中的运算结果 3 与 3 相加得到 6.

第三步：将第二步中的运算结果 6 与 4 相加得到 10.

第四步：将第三步中的运算结果 10 与 5 相加得到 15.

第五步：将第四步中的运算结果 15 与 6 相加得到 21.

算法 2：

第一步：取 n=6.

第二步：计算 $\frac{n(n+1)}{2}$.

第三步：输出运算结果.

算法 3：

第一步：将原式变形为(1+6)+(2+5)+(3+4)=3×7.

第二步：计算 3×7.

第三步：输出运算结果.

小结：算法 1 是最原始的方法，最为繁琐，步骤较多，当加数较大时，如 1+2+3+...+10000，再用这种方法是行不通的；算法 2 与算法 3 都是比较简单的算法，但比较而言，算法 2 最为简单，且易于在计算机上执行操作.

学生做一做：求 1×3×5×7×9×11 的值，写出其算法.

老师评一评：

算法 1：

第一步：先求 1×3，得到结果 3.

第二步：将第一步所得结果 3 再乘以 5，得到结果 15.

第三步：再将 15 乘以 7，得到结果 105.

第四步：再将 105 乘以 9，得到 945.

第五步：再将 945 乘以 11，得到 10395，即是最后结果.

算法 2：用 P 表示被乘数，i 表示乘数.

第一步：使 P=1.

第二步：使 i=3.

第三步：使 P=P×i.

第四步：使 i=i+2.

第五步：若 $i\leqslant 11$，则返回到第三步继续执行，否则算法结束.

小结：由于计算机是高速计算的自动机器，能实现循环的语句. 因此，上述算法 2 不仅是正确的，而且是在计算机上能够实现得较好的算法. 在上面的算法中，第三步、第四步、第五步构成一个完整的循环，这里需要说明的是，每经过一次循环之后，变量 P、i 的值都发生

了变化，并且循环一次之后都要在第五步对 i 的值进行检验，一旦发现 i 的值大于 11 时，立即停止循环，同时输出最后一个 P 的值，对于循环结构的详细情况，我们将在以后的学习中介绍.

课堂练习

1．写出解不等式 x^2-2x-3<0 的一个算法.

解：第一步：x^2-2x-3=0 的两个根是 x_1=3，x_2=-1.

第二步：由 x^2-2x-3<0 可知不等式的解集为{x | -1<x<3}.

评注：该题的解法具有一般性，下面给出解形如 $ax^2+bx+c>0$ 的不等式的步骤（为方便，我们设 $a>0$）如下：

第一步：计算 $\Delta = b^2 - 4ac$.

第二步：若 $\Delta>0$，求出方程两根 $x_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ （设 $x_1>x_2$），则不等式解集为{x | $x>x_1$ 或 $x<x_2$}.

第三步：若 $\Delta = 0$，则不等式解集为$\{x \mid x \in \mathrm{R}$ 且 $x \neq -\dfrac{b}{2a}\}$.

第四步：若 $\Delta<0$，则不等式的解集为 R.

2．求过 $P(a_1,b_1)$、$Q(a_2,b_2)$两点的直线斜率有如下的算法：

第一步：取 $x_1= a_1$，$y_1= b_1$，$x_2= a_2$，$y_1= b_2$.

第二步：若 $x_1= x_2$.

第三步：输出斜率不存在.

第四步：若 $x_1 \neq x_2$.

第五步：计算 $k = \dfrac{y_2 - y_1}{x_2 - x_1}$.

第六步：输出结果.

3．写出求过两点 M(-2,-1)、N(2,3)的直线与坐标轴围成面积的一个算法.

解：

第一步：取 $x_1 = -2$，$y_1 = -1$，$x_2 = 2$，$y_2 = 3$.

第二步：计算 $\dfrac{y - y_1}{y_2 - y_1} = \dfrac{x - x_1}{x_2 - x_1}$.

第三步：在第二步结果中令 x=0 得到 y 的值 m，得直线与 y 轴交点(0,m).

第四步：在第二步结果中令 y=0 得到 x 的值 n，得直线与 x 轴交点(n,0).

第五步：计算 $S = \dfrac{1}{2}|m||n|$.

第六步：输出运算结果.

7.1.2 程序框图

1. 程序框图的概念

算法可以用自然语言来描述，但为了使算法的程序或步骤表达得更为直观，我们更经常地用图形方式来表示它，即程序框图. 程序框图又称流程图，是一种用规定的图形、指向线及文字说明来准确、直观地表示算法的图形.

2. 构成程序框图的图形符号及其作用

（1）起止框  ：是任何流程图都不可缺少的，它表明程序的开始和结束，所以一个完整的流程图的首末两端必须是起止框.

（2）输入、输出框  ：表示数据的输入或结果的输出，它可用在算法中的任何需要输入、输出的位置.

（3）处理框  ：是用来赋值、执行计算语句、传送运算结果的图形符号.

（4）判断框  ：一般有一个入口和两个出口，有时也有多个出口，它是唯一具有两个或两个以上出口的符号，在只有两个出口的情形中，通常都分成“是”与“否”（也可用“Y”与“N”）两个分支.

在学习这部分知识的时候，要掌握各个图形的形状、作用及使用规则，画程序框图的规则如下：

（1）使用标准的图形符号，如图 7-1 所示.

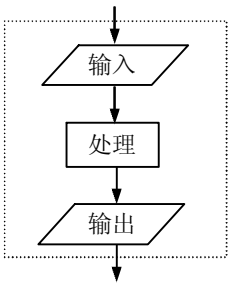

图 7-1

（2）框图一般按从上到下、从左到右的方向画.

（3）除判断框外，大多数流程图符号只有一个进入点和一个退出点. 判断框是具有超过一个退出点的唯一符号.

（4）判断框分两大类，一类是“是”与“否”的两分支判断，而且有且仅有两个结果；另一类是多分支判断，有几种不同的结果.

（5）在图形符号内描述的语言要非常简练清楚.

3. 顺序结构

顺序结构描述的是最简单的算法结构，语句与语句之间、框与框之间是按从上到下的顺序进行的.

例 1：交换两个变量 A 和 B 的值，并输出交换前后的值.

解：算法如下：

第一步：输入 A、B 的值.

第二步：把 A 的值赋给 x.

第三步：把 B 的值赋给 A.

第四步：把 x 的值赋给 B.

第五步：输出 A、B 的值.

程序框图如图 7-2 所示.

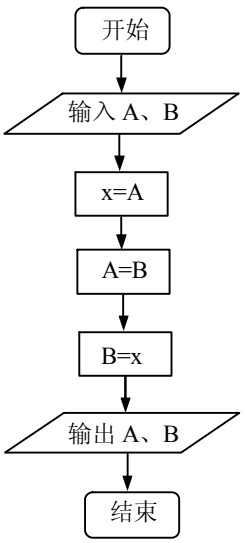

图 7-2

4. 条件结构

一些简单的算法可以用顺序结构来表示，但是这种结构无法对描述对象进行逻辑判断并根据判断结果进行不同的处理. 因此，需要有另一种逻辑结构来处理这类问题，这种结构叫做条件结构，它是根据指定条件选择执行不同指令的控制结构，如图 7-3 所示.

例 2：有三个整数 a、b、c，由键盘输入，输出其中最大的数.

解：算法 1：

第一步：输入 a、b、c.

第二步：若 $a>b$ 且 $a>c$，则输出 a；否则，执行第三步.

第三步：若$b>c$，则输出b；否则，输出c.

算法 2：

第一步：输入a、b、c.

第二步：若$a>b$，则$t=a$；否则，$t=b$.

第三步：若$t>c$，则输出t；否则，输出c.

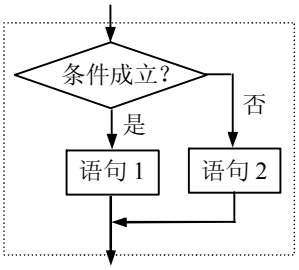

图 7-3

例 3：已知$f(x)=x^2-2x-3$，求$f(3)+f(-5)$的值.

设计出解决该问题的一个算法，并画出程序框图.

解：算法如下：

第一步：$x=3$.

第二步：$y_1=x^2-2x-3$.

第三步：$x=-5$.

第四步：$y_2=x^2-2x-3$.

第五步：$y=y_1+y_2$.

第六步：输出y.

例 4：设计一个求任意数的绝对值的算法，并画出程序框图.

解：第一步：输入任意实数x.

第二步：若$x\geqslant 0$，则$y=x$；否则$y=-x$.

第三步：输出y.

5. 循环结构

例 5：设计一个计算 1+2+…+100 的值的算法.

解：算法 1：按照逐一相加的程序进行.

第一步：计算 1+2，得到 3.

第二步：将第一步中的运算结果 3 与 3 相加，得到 6.

第三步：将第二步中的运算结果 6 与 4 相加，得到 10.

……

第九十九步：将第九十八步中的运算结果 4950 与 100 相加，得到 5050.

简化描述：

第一步：sum=0.

第二步：sum=sum+1.

第三步：sum=sum+2.

第四步：sum=sum+3.

……

第一百步：sum=sum+99.

第一百零一步：sum=sum+100.

第一百零二步：输出 sum.

进一步简化：

第一步：sum=0，i=1.

第二步：依次 i 从 1 到 100 反复做 sum=sum+i.

第三步：输出 sum.

在一些算法中，经常会出现从某处开始，按照一定条件，反复执行某一处理步骤的情况，这就是循环结构，反复执行的处理步骤为循环体，显然循环结构中一定包含条件结构，如图 7-4 所示.

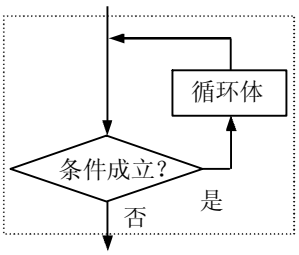

图 7-4

循环体：反复执行的处理步骤称为循环体.

计数变量：在循环结构中，通常都有一个起到循环计数作用的变量，这个变量的取值一般都含在执行或终止循环体的条件中.

当型循环：在每次执行循环体前对控制循环条件进行判断，当条件满足时执行循环体，不满足则停止.

直到型循环：在执行了一次循环体之后，对控制循环体进行判断，当条件不满足时执行循环体，满足则停止.

当型循环与直到型循环的区别：①当型循环可以不执行循环体，直到型循环至少执行一次循环体；②当型循环先判断后执行，直到型循环先执行后判断；③对同一算法来说，当型循环和直到型循环的条件互为反条件.

小结：画循环结构程序框图前：①确定循环变量和初始条件；②确定算法中反复执行的

部分，即循环体；③确定循环的转向位置；④确定循环的终止条件.

6. 条件结构与循环结构的区别与联系

区别：条件结构通过判断分支，只是执行一次；循环结构通过条件判断可以反复执行.

联系：循环结构是通过条件结构来实现的.

7.2 基本算法语句

在现代社会里，计算机已经成为人们日常生活和工作不可缺少的工具，如听 MP3、看电影、玩游戏、打字排版、画卡通画、处理数据等，那么计算机是怎样工作的呢？

计算机完成任何一项任务都需要算法，但是我们用自然语言或程序框图描述的算法，计算机是无法“看得懂，听得见”的. 因此还需要将算法用计算机能够理解的程序设计语言（Programming Language）翻译成计算机程序.

7.2.1 输入语句、输出语句和赋值语句

我们知道，顺序结构是任何一个算法都离不开的基本结构. 输入语句、输出语句和赋值语句基本上对应于算法中的顺序结构. 如图 7-5 所示，计算机从上而下按照语句排列的顺序执行这些语句.

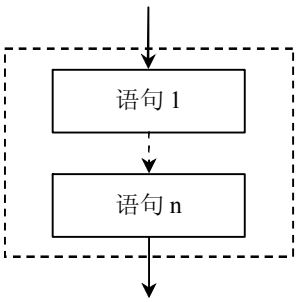

图 7-5

输入语句和输出语句分别用来实现算法的输入信息、输出结果的功能. 如下面的例子：用描点法作函数 $y=x^3+3x^2-24x+30$ 的图像时，需要求出自变量与函数的一组对应值. 编写程序，分别计算当 x =-5、-4、-3、-2、-1、0、1、2、3、4、5 时的函数值.

```
INPUT "x=";x
 y=x^3+3*x^2−24*x+30
PRINT  x
PRINT y
END
```

【提问】在这个程序中，你们觉得哪些是输入语句、输出语句和赋值语句呢？

1．输入语句

在该程序中，第 1 行中的 INPUT 语句就是输入语句．这个语句的一般格式为：

```
INPUT "提示内容";变量
```

其中，“提示内容”一般是提示用户输入什么样的信息．如每次运行上述程序时，依次输入-5、-4、-3、-2、-1、0、1、2、3、4、5，计算机每次都把新输入的值赋给变量 x，并按 x 新获得的值执行下面的语句．

INPUT 语句不但可以给单个变量赋值，还可以给多个变量赋值，其格式为：

```
INPUT "提示内容 1,提示内容 2,提示内容 3,…";变量 1,变量 2,变量 3,…
```

说明：①输入语句的作用是实现算法的输入信息功能；②“提示内容”提示用户输入什么样的信息，用双引号括起来；③提示内容与变量之间用分号“;”隔开，若输入多个变量，变量与变量之间用逗号“,”隔开，如“INPUT "a=,b=,c=";a,b,c”；④变量是指程序在运行时其值是可以变化的量；如③中的 a、b、c 都是变量，通俗地把一个变量比喻成一个盒子，盒子内可以存放数据，可随时更新盒子内的数据；⑤如③中当依次输入了 1、2、3 程序在运行时把输入的值依次赋给 a、b、c，即 a=1，b=2，c=3．

例如，输入一个学生数学、语文、英语三门课的成绩：

```
INPUT "Maths,Chines,English"; a,b,c
```

输入任意整数 n：

```
INPUT "n="; n
```

2．输出语句

在该程序中，第 3 行和第 4 行中的 PRINT 语句是输出语句．它的一般格式为：

```
PRINT "提示内容";表达式
```

说明：①输出语句的作用是实现算法的输出结果的功能，可以在计算机的屏幕上输出常量、变量的值和系统信息；②“提示内容”提示用户输出什么样的信息，用双引号括起来；③提示内容与表达式之间用分号“;”隔开；④要输出表达式中的字符，需要用双引号""，如：PRINT "提示内容：";"a+2"，这时屏幕上将显示：提示内容：a+2．

例如下面的语句可以输出斐波那契数列：

```
PRINT  "The Fibonacci Progression is:";
1  1  2  3  5  8  13  21  34  55 "…"
```

此时屏幕上显示：

```
The Fibonacci Progression is：1  1  2  3  5  8  13  21  34  55 …
```

3．赋值语句

是用来表明赋给某一个变量一个具体的确定值的语句．

除了输入语句，在该程序第 2 行中的赋值语句也可以给变量提供初值．它的一般格式为：

```
变量=表达式
```

说明：①赋值语句的作用是将表达式所代表的值赋给变量；②赋值语句中的“=”叫做赋值号，它和数学中的等号不完全一样．赋值号的左右两边不能对换，赋值语句是将赋值号右边的表达式的值赋给赋值号左边的变量，如 a=b 表示用 b 的值代替变量 a 原先的值；③格式中右边的“表达式”可以是一个数据、常量和算式，如果“表达式”是一个算式时，赋值语句的作用是先计算出“=”右边表达式的值，然后将该值赋给“=”左边的变量，如 a=1，b=2，c=a+b 是指先计算 a+b 的值 3 再赋给 c，而不是将 a+b 赋给 c．

4．输入语句、输出语句和赋值语句之间的区别

（1）输入语句和赋值语句的区别：输入语句是外部直接给程序中的变量赋值；赋值语句是程序内部运行时给变量赋值，先计算右边的表达式，得到的值赋给左边的变量．

（2）输入语句和输出语句的区别：输入语句是外部直接给程序中的变量赋值；输出语句是程序运行的结果输出到外部，先计算表达式，得到结果输出．

例 1：编写程序，计算一个学生数学、语文、英语三门课的平均成绩．

分析：先写出算法，画出程序框图（如图 7-6 所示），再进行编程．

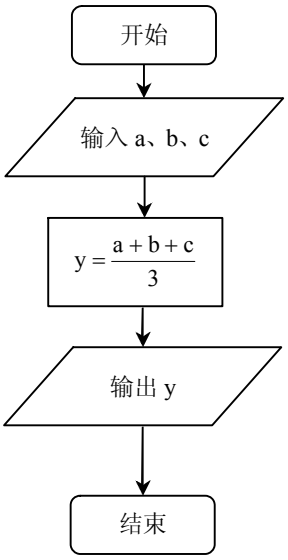

图 7-6

```
INPUT "数学=";a
INPUT "语文=";b
INPUT "英语=";c
 y=(a+b+c)/3
PRINT  "The average=";y
END
```

例 2：编写一个程序，要求输入一个圆的半径便能输出该圆的周长和面积（π 取 3.14）．

分析：设圆的半径为 R，则圆的周长为 $C=2\pi R$，面积为 $S=\pi R^2$，可以利用顺序结构中的 INPUT 语句、PRINT 语句和赋值语句设计程序.

```
INPUT "半径为 R=";R
C=2*3.14*R
S=3.14*R^2
PRINT "该圆的周长为:";C
PRINT "该圆的面积为:";S
END
```

7.2.2　条件语句和循环语句

试求自然数 1+2+3+…+99+100 的和.

显然大家都能准确地口算出它的答案：5050．而能不能将这项计算工作交给计算机来完成呢？而要编程，以我们前面所学的输入语句、输出语句和赋值语句还不能满足“我们日益增长的物质需要”，因此，还需要进一步学习基本算法语句中的另外两种：条件语句和循环语句.

1．条件语句

算法中的条件结构是由条件语句来表达的，是处理条件分支逻辑结构的算法语句．它的一般格式为 IF-THEN-ELSE 格式.

当计算机执行上述语句时，首先对 IF 后的条件进行判断，如果条件符合，就执行 THEN 后的语句 1，否则执行 ELSE 后的语句 2．其对应的程序框图如图 7-7 所示.

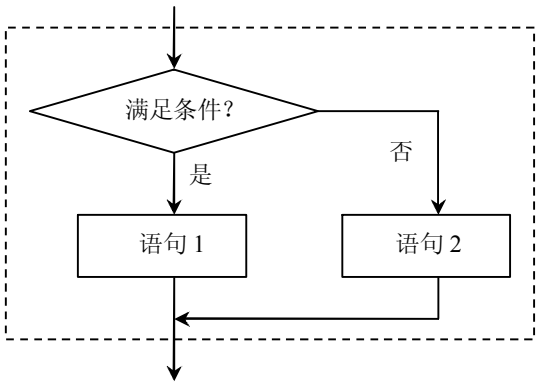

图 7-7

在某些情况下，也可以只使用 IF-THEN 语句，即 IF-THEN 格式.

```
IF 条件　THEN
  语句
END IF
```

计算机执行这种形式的条件语句时，也是首先对 IF 后的条件进行判断，如果条件符合，就执行 THEN 后的语句；如果条件不符合，则直接结束该条件语句，转而执行其他语句．其对应的程序框图如图 7-8 所示．

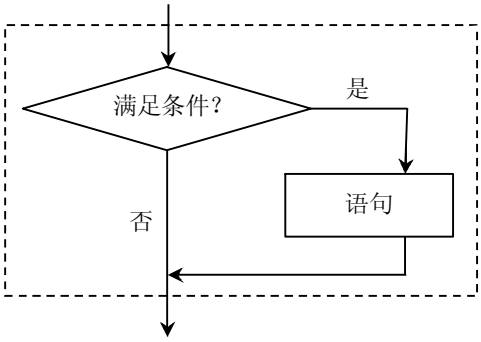

图 7-8

```
INPUT  "Please input a,b,c =";a,b,c
    d=b*b-4*a*c
    p=-b/(2*a)
    q=SQR(ABS(d))/(2*a)
IF   d>=0   THEN
     x1=p+q
     x2=p-q
IF   x1=x2   THEN
    PRINT "One real root:";x1
ELSE
    PRINT "Two real roots:x1";x1,"and x2";x2
END IF
ELSE
    PRINT   "No real root!"
END IF
END
```

条件语句的作用：在程序执行过程中，根据判断是否满足约定的条件而决定需要转换到何处去．需要计算机按条件进行分析、比较、判断，并按判断后的不同情况进行不同的处理．

例 1：编写程序，输入一元二次方程 $ax^2+bx+c=0$ 的系数，输出它的实数根．

分析：先把解决问题的思路用程序框图表示出来，然后再根据程序框图给出的算法步骤逐步把算法用对应的程序语句表达出来．

算法分析：我们知道，若判别式 $\Delta=b^2-4ac>0$，原方程有两个不相等的实数根：$x_1=\dfrac{-b+\sqrt{\Delta}}{2a}$、$x_2=\dfrac{-b-\sqrt{\Delta}}{2a}$；若 $\Delta=0$，原方程有两个相等的实数根：$x_1=x_2=-\dfrac{b}{2a}$；若 $\Delta<0$，

原方程没有实数根．也就是说，在求解方程之前，需要首先判断判别式的符号．因此，这个过程可以用算法中的条件结构来实现．

又因为方程的两个根有相同的部分，为了避免重复计算，可以在计算 x_1 和 x_2 之前先计算 $p=-\frac{b}{2a}$，$q=\frac{\sqrt{|\Delta|}}{2a}$．

例 2：编写程序，使得任意输入的 3 个整数按从大到小的顺序输出．

算法分析：用 a、b、c 表示输入的 3 个整数；为了节约变量，把它们重新排列后，仍用 a、b、c 表示，并使 $a \geqslant b \geqslant c$．具体操作步骤如下：

第一步：输入 3 个整数 a、b、c．

第二步：将 a 与 b 比较，并把小者赋给 b，大者赋给 a．

第三步：将 a 与 c 比较，并把小者赋给 c，大者赋给 a，此时 a 已是三者中最大的．

第四步：将 b 与 c 比较，并把小者赋给 c，大者赋给 b，此时 a、b、c 已按从大到小的顺序排列好．

第五步：按顺序输出 a、b、c．

补例：铁路部门托运行李的收费方法如下：y 是收费额（单位：元），x 是行李重量（单位：kg），当 $0<x \leqslant 20$ 时，按 0.35 元/kg 收费；当 $x>20$kg 时，20kg 的部分按 0.35 元/kg 收费；超出 20kg 的部分，则按 0.65 元/kg 收费．请根据上述收费方法编写程序．

分析：首先由题意得：$y=\begin{cases}0.35x & 0<x \leqslant 20\\ 0.35\times 20+0.65(x-20) & x>20\end{cases}$，该函数是个分段函数，需要对行李重量作出判断，因此这个过程可以用算法中的条件结构来实现．

```
INPUT "请输入旅客行李的重量（kg）x=";x
IF   x>0   AND   x<=20   THEN
    y=0.35*x
ELSE
    y=0.35*20+0.65*(x-20)
END IF
    PRINT "该旅客行李托运费为：";y
END
```

2. *循环语句*

算法中的循环结构是由循环语句来实现的．对应于程序框图中的两种循环结构，一般程序设计语言中也有当型（WHILE 型）和直到型（UNTIL 型）两种语句结构，即 WHILE 语句和 UNTIL 语句．

（1）WHILE 语句．

一般格式如下：

```
WHILE   条件
```

```
  循环体
WEND
```

其中循环体是由计算机反复执行的一组语句构成的．WHILE 后面的“条件”是用于控制计算机执行循环体或跳出循环体的．

当计算机遇到 WHILE 语句时，先判断条件的真假，如果条件符合，就执行 WHILE 与 WEND 之间的循环体；然后再检查上述条件，如果条件仍符合，再次执行循环体，这个过程反复进行，直到某一次条件不符合为止．这时，计算机将不执行循环体，直接跳到 WEND 语句后，接着执行 WEND 之后的语句．因此，当型循环有时也称为“前测试型”循环．其对应的程序结构框图如图 7-9 所示．

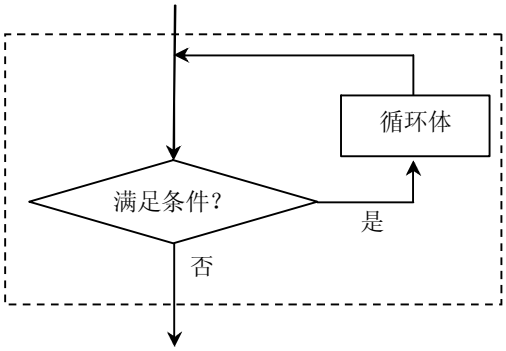

图 7-9

（2）UNTIL 语句．

一般格式如下：

```
DO
  循环体
LOOP  UNTIL  条件
```

其对应的程序结构框图如图 7-10 所示．

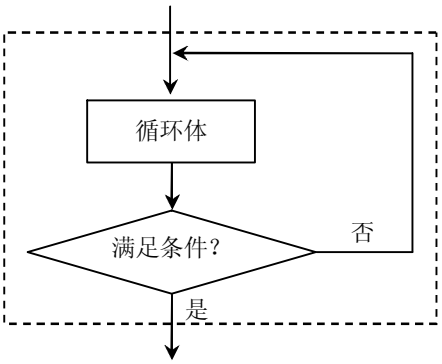

图 7-10

【思考】直到型循环又称为“后测试型”循环，参照其直到型循环结构对应的程序框图，说说计算机是按怎样顺序执行 UNTIL 语句的（让学生模仿执行 WHILE 语句的表述）？

从 UNTIL 型循环结构分析，计算机执行该语句时，先执行一次循环体，然后进行条件的判断，如果条件不满足，继续返回执行循环体，然后再进行条件的判断，这个过程反复进行，直到某一次条件满足时，不再执行循环体，跳到 LOOP UNTIL 语句后执行其他语句，是先执行循环体后进行条件判断的循环语句.

【提问】通过对照，大家觉得 WHILE 型语句与 UNTIL 型语句之间有什么区别呢（让学生表达自己的感受）？

区别：在 WHILE 语句中，是当条件满足时执行循环体；而在 UNTIL 语句中，是当条件不满足时执行循环体.

例 3：编写程序，计算自然数 1+2+3+…+99+100 的和.

分析：这是一个累加问题．我们可以用 WHILE 型语句，也可以用 UNTIL 型语句．由此看来，解决问题的方法不是唯一的，当然程序的设计也是有多种的，只是程序简单与复杂的问题.

WHILE 型：

```
i=1
sum=0
WHLIE   i<=100
  sum=sum+i
  i=i+1
WEND
PRINT   sum
END
```

UNTIL 型：

```
i=1
sum=0
DO
   sum=sum+i
   i=i+1
LOOP UNTIL   i>100
PRINT   sum
END
```

例 4：将程序框图 7-4 转化为程序语句.

分析：仔细观察，该程序框图中既有条件结构，又有循环结构.

```
INPUT  "n=";n
flag=1
IF   n>2   THEN
  d=2
  WHILE   d<=n-1   AND   flag=1
       IF   n MOD d=0   THEN
            flag=0
       ELSE
            d=d+1
       END IF
  WEND
ELSE
  IF   flag=1   THEN
```

```
        PRINT n;"是质数。"
    ELSE
        PRINT n;"不是质数。"
    END IF
END IF
END
```

思考：上述判定质数的算法是否还能有所改进（让学生课后思考）？

补例：某纺织厂 1997 年的生产总值为 300 万元，如果年生产增产率为 5%，计算最早在哪一年生产总值超过 400 万元.

分析：从 1997 年底开始，经过 x 年后生产总值为 $300\times(1+5\%)^x$，可将 1997 年生产总值赋给变量 a，然后对其进行累乘，用 n 作为计数变量进行循环，直到 a 的值超过 400 万元为止.

解：程序框图如图 7-11 所示.

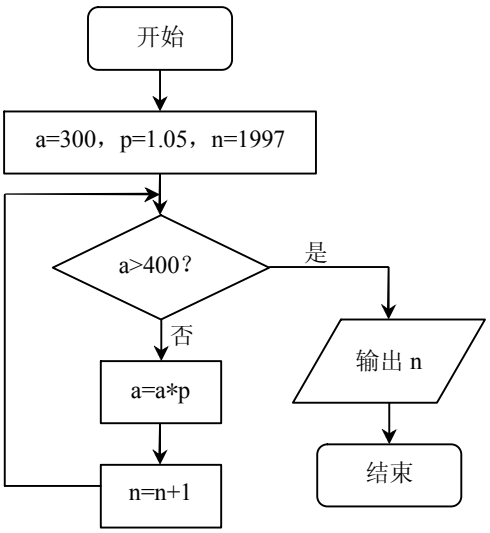

图 7-11

```
a=300
p=1.05
n=1997
DO
   a=a*p
   n=n+1
LOOP UNTIL a>400
PRINT n
END
```

7.3　算法案例

在初中，我们已经学过求最大公约数的知识，你能求出 18 与 30 的公约数吗？我们以前都是利用找公约数的方法来求最大公约数，如果公约数比较大而且根据我们的观察又不能得到一些公约数，我们又应该怎样求它们的最大公约数呢？比如求 8251 与 6105 的最大公约数.

1. 辗转相除法

例 1：求两个正数 8251 和 6105 的最大公约数.

分析：8251 与 6105 两数都比较大，而且没有明显的公约数，如能把它们都变小一点，根据已有的知识即可求出最大公约数.

解：8251=6105×1+2146

显然 8251 的最大公约数也必是 2146 的约数，同样 6105 与 2146 的公约数也必是 8251 的约数，所以 8251 与 6105 的最大公约数也是 6105 与 2146 的最大公约数.

6105=2146×2+1813

2146=1813×1+333

1813=333×5+148

333=148×2+37

148=37×4+0

则 37 为 8251 与 6105 的最大公约数.

以上我们求最大公约数的方法就是辗转相除法，也叫欧几里德算法，它是由欧几里德在公元前 300 年左右首先提出的. 利用辗转相除法求最大公约数的步骤如下：

第一步：用较大的数 m 除以较小的数 n 得到一个商 q_0 和一个余数 r_0.

第二步：若 $r_0=0$，则 n 为 m 和 n 的最大公约数；若 $r_0\neq 0$，则用除数 n 除以余数 r_0 得到一个商 q_1 和一个余数 r_1.

第三步：若 $r_1=0$，则 r_1 为 m 和 n 的最大公约数；若 $r_1\neq 0$，则用除数 r_0 除以余数 r_1 得到一个商 q_2 和一个余数 r_2.

……

依次计算直至 $r_n=0$，此时所得到的 r_{n-1} 即为所求的最大公约数.

练习：利用辗转相除法求两数 4081 与 20723 的最大公约数（答案：53）.

2. 更相减损术

我国早期也有解决求最大公约数问题的算法，就是更相减损术.

更相减损术求最大公约数的步骤为：可半者半之，不可半者，副置分母・子之数，以少减多，更相减损，求其等也，以等数约之.

翻译出来为：

第一步：任意给出两个正数，判断它们是否都是偶数：若是，用 2 约简；若不是，执行

第二步.

第二步：以较大的数减去较小的数，接着把较小的数与所得的差比较，并以大数减小数. 继续这个操作，直到所得的数相等为止，则这个数（等数）就是所求的最大公约数.

例 2：用更相减损术求 98 与 63 的最大公约数.

解：由于 63 不是偶数，把 98 和 63 以大数减小数并辗转相减，即：

98-63=35

63-35=28

35-28=7

28-7=21

21-7=14

14-7=7

所以，98 与 63 的最大公约数是 7.

练习：用更相减损术求两个正数 84 与 72 的最大公约数（答案：12）.

3. 比较辗转相除法与更相减损术的区别

（1）都是求最大公约数的方法，计算上辗转相除法以除法为主，更相减损术以减法为主，计算次数上辗转相除法计算次数相对较少，特别当两个数字大小区别较大时计算次数的区别较明显.

（2）从结果体现形式来看，辗转相除法体现结果是以相除余数为 0 则得到，而更相减损术则以减数与差相等而得到.

4. 辗转相除法计算的程序框图及程序

利用辗转相除法与更相减损术的计算算法，我们可以设计出程序框图以及 BASIC 程序来在计算机上实现辗转相除法与更相减损术求最大公约数，下面由同学们设计相应框图，相互之间检查框图与程序的正确性，并在计算机上验证自己的结果.

辗转相除法的程序框图如图 7-12 所示.

程序如下：

```
INPUT "m=";m
INPUT "n=";n
IF   m<n THEN   x=m
  m=n
  n=x
END IF
r=m MOD n
WHILE   r<>0
  r=m MOD n
  m=n
  n=r
WEND
PRINT   m
END
```