第一章　程序设计基础知识

- 了解计算机的工作原理；
- 掌握计算机中各种数制的转换方法；
- 掌握如何用 N-S 流程图表示算法。

§1.1　计算机的工作原理

一、计算机的指令系统

大家知道，计算机中的存储器是由千千万万的电子线路单元组成，每个单元有两个稳定的工作状态（例如二极管或三极管的截止和导通，磁性元件的消磁和充磁等），分别以 0 和 1 表示，因此电子计算机存储的信息是以二进制形式存储的。人们要用计算机处理信息，就要给计算机规定一些最基本的操作，并用 0和 1表示这些操作，这就构成一条一条的指令。在设计的时候，就给它规定了一套指令，称之为指令系统（即 Instruction set）。不同型号的计算机，指令系统也不相同。

一条指令由操作码（Opcode）和操作数（Operand）两部分构成，例如在 Z80 中有这样一条指令：

11000110	00000110
操作码	操作数

操作码 11000110 表示加法操作，操作数是 00000110。这条指令的功能是把操作数 00000110 与计算机的累加器中的数相加，相加的和仍放在累加器中，例如先在累加器中放一个数 00000101,执行这条指令的过程如图 1.1 所示。这条指令用十六进制表示为：C6　06。

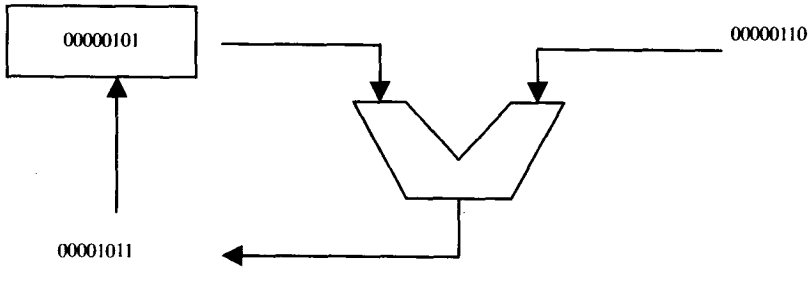


图 1.1　执行过程

二、计算机的解题过程

计算机算题要由人事先告诉它算题的方法和步骤，一步一步地去执行。如果人们设计的步骤是正确的，计算机就能算出正确的结果；如果设计的步骤不正确，计算机就不能算出正确的结果，甚至没有结果。

以简单的 5+6 加法为例，它的解题步骤如下：

1. 把数字 5 和 6 送到计算机的内存中存放起来，存储单元都要有一个编号，称为地址。例如 43 号地址存放数 5，写为(43)←5，同样，(44)←6;
2. 把数 5 取出来，送到累加器；
3. 把数 6 取出来，与累加器中的数相加，结果放在累加器中；
4. 把累加器结果送回到内存的 45 号地址存放起来，即(45)←11;
5. 把结果输出到打印机或显示器上；
6. 结束。

这些解题步骤的集合，我们称之为程序，第一步是数据的输入，第五步是数据的输出，2~4 是计算机内部的处理，用某种机器指令写出 2~4 这一过程，有如下形式：

存储地址	机器指令
⋮	
00010000	00111011
00010001	00000000
00010010	00001011
00010011	00100001
00010100	00000000
00010101	00001100
00010110	10000110
00010111	00110010
00011000	00000000
00011001	00101101
⋮	
00101011	00000101
00101100	00000110
00101101	00001011

这些由机器指令构成的有序集合，称为机器语言程序。计算机的工作就是按规定的顺序执行程序。人们使用计算机就要为它编制程序，我们称为程序设计。用机器语言编写程序很不直观，初学者看到这个程序就不知其所以然了。不必着急，看不懂没关系，这只是让你对机器语言有点感性认识。对于计算机来说，只有这样的机器语言，才能执行。

三、存储程序原理

有了指令和程序的概念，就可以进一步了解计算机的工作原理，计算机是基于存储程序的方法工作的。

首先，把程序和数据通过输入设备送入内存。一般的内存都是划分为很多存储单元，

每个存储单元都有地址编号，这样按一定顺序把程序和数据存起来，而且还把内存分为若干区域，比如有专门存放程序的程序区和专门存放数据的数据区。

其次，执行程序，必须从第一条指令开始，以后一条一条的执行。一般的情况下按存放地址号的顺序，由小到大依此执行，当遇到条件转移的指令时，才改变执行的顺序。每执行一条指令，都要经过三个步骤。第一步，把指令从内存中送往译码器，称为取指；第二步，译码器把指令分解成操作码和操作数，产生相应的各种控制信号送往各电器部件；第三步，执行相应的操作。这一过程是由电子线路来控制的，从而实现自动连续的工作。

这一原理是计算机结构设计的基础，它是美籍匈牙利数学家冯·诺依曼（Von Neumann）1946 年提出并论证的，因此常把这一类型的计算机称为冯·诺依曼计算机，迄今为止各种计算机仍属这一类型。

§1.2 计算机中数的表示与编码

一、二进制数

在日常生活中，我们最熟悉的是逢十进位的十进制数。它有两个基本特征：

1. 数位从右至左，每位代表的数值是按 10 的指数增加的，如个、十、百、千…，这个数值，我们常称为位权值。例如 2184 这个数可以用多项式表示为

$$2184=2\times10^3+1\times10^2+8\times10^1+4\times10^0$$

千位 百位 十位 个位

可见，十进制数的位权值是以 10 为底的幂，因此任何一个十进制数都可以用一个多项式表示：

$$(N)_{10}=a_n\cdot10^n+a_{n-1}\cdot10^{n-1}+\cdots+a_1\cdot10^1+a_0\cdot10^0+a_{-1}\cdot10^{-1}+\cdots+a_{-m}\cdot10^{-m}$$

整数部分 小数部分

2. 每一位数上要有 10 个不同的符号：0、1、2、3、4、5、6、7、8、9。在多项式中的 a_n、a_{n-1}、…可以是这 10 个中的任意一个。

我们用 10 个不同形状的字符表示 10 个数，但是在计算机的硬件中，还不可能找到 10 种不同形状的物理量来表示 10 个数。因此，数制必须修改。我们知道，计算机是基于电子元件的特性来工作的，这些电子线路处理的是数字信号，它们表现为电位的高与低，电路的接通与断开等两种状态，我们用“0”、“1”来描述。这样就可以用 0、1 两个状态表示数，以 2 为底的幂作位权值，建立逢 2 进位的二进制数。例如：

$$(1101)_2=1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0$$

表 1.1 列出从 0~15 的各种进制数的表示。

任何一个二进制数都可以用以下多项式表示：

$$(N)_2=a_n\cdot2^n+a_{n-1}\cdot2^{n-1}+\cdots+a_1\cdot2^1+a_0\cdot2^0+a_{-1}\cdot2^{-1}+\cdots+a_{-m}\cdot2^{-m}$$

整数部分 小数部分

式中 a_n、a_{n-1}、…是 0 或 1。

表 1.1　各种数制对应表

数	十进制	二进制	八进制	十六进制	数	十进制	二进制	八进制	十六进制
零	0	0	0	0	八	8	1000	10	8
一	1	1	1	1	九	9	1001	11	9
二	2	10	2	2	十	10	1010	12	A
三	3	11	3	3	十一	11	1011	13	B
四	4	100	4	4	十二	12	1100	14	C
五	5	101	5	5	十三	13	1101	15	D
六	6	110	6	6	十四	14	1110	16	E
七	7	111	7	7	十五	15	1111	17	F

二、二进制与十进制间的转换

1．二进制数转换为十进制数

按位权值展开，求多项式之和，就能得到十进制数。

为了区别不同的数制，我们把数值用圆括号括起来，在右下角用小号字体的 2、10 等来注明，如 $(110101)_2$ 表示是二进制数，$(216)_{10}$ 表示是十进制数。

①整数的转换

例如：$(110101)_2=1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0$

$=32+16+0+4+0+1$

$=(53)_{10}$

②小数的转换

例如：$(0.101)_2=1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

$=0.5+0+0.125$

$=(0.625)_{10}$

③既有整数部分又有小数部分的数转换时分别按整数和小数转换，再用小数点连起来，

例如：$(110101.101)_2=(110101)_2+(0.101)_2$

$=(53)_{10}+(0.625)_{10}$

$=(53.625)_{10}$

2．十进制转换为二进制数

①整数的转换　把十进制整数连续除以 2，记录其余数，就得到二进制数，这个方法简称为除 2 取余法。

例如：将 59 转换为二进制数

```
2 | 59        余数
  2 | 29      1 ……最低位 a0
    2 | 14    1 ……a1
      2 | 7   0 ……a2
        2 | 3 1 ……a3
          2 | 1 1 ……a4
            0   1 ……最高位 a5
```

按由高位到低位写下来就得到：

$$(59)_{10}=(111011)_2$$

② 小数的转换　把十进制小数连续乘以 2，取出其积的整数，就得到二进制数，这个

方法简称为乘 2 取整法。

例如：乘积的整数部分

	↓	0.5625 × 2
a_{-1} 最高位	→1←	1.1250 × 2
a_{-2}	0←	0.2500 × 2
a_{-3}	0←	0.5000 × 2
a_{-4} 最低位	→1←	1.0000

于是得到 $(0.5625)_{10}=(0.1001)_2$

小数转换中，不是都能得到小数部分乘积为零的结果，这时，只能按精度要求取若干位作为它的近似值。

③既有整数部分又有小数部分的数，转换时要把整数部分和小数部分分别转换，再把两部分加起来，就是一个二进制数。

例如：$(59.3)_{10}=(59)_{10}+(0.3)_{10}$

上例已求出 $(59)_{10}=(111011)_2$

$(0.3)_{10}\approx(0.01001)_2$

所以 $(59.3)_{10}\approx(111011.01001)_2$

三、八进制数和十六进制数

计算机只能识别二进制数，当数值愈大，二进制数表示的位数就愈多，例如：

$(11011011011.001)_2=(1755.125)_{10}$

这个二进制数整数有 11 位，小数有 3 位(可以表示上千的十进制数),在读数和书写时，都极不方便。因此，在编写程序时又常常使用八进制数和十六进制数。

和十进制、二进制的规则相仿，八进制数采用 8 个符号(即 0、1、2、3、4、5、6、7)，逢 8 进位。十六进制数采用 16 个符号（即 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F)，逢 16 进位。

例如：$(16)_8=1\times8^1+6\times8^0=(14)_{10}$

$(25)_{16}=2\times16^1+5\times16^0=(37)_{10}$

以下介绍各种数制之间的转换方法：

1．八进制、十六进制转换为十进制　方法与上面二进制与十进制的转换方法相类似。如上例所示。

2．八进制数转换成二进制数　把每位八进制数用所对应的 3 位二进制数表示，就转换为它的二进制数。

例如：

(3	2	$6)_8$	
↓	↓	↓	
(011)	(010)	(110)	

即　　$(326)_8=(11010110)_2$

3．十六进制数转换成二进制数　把每位十六进制数用所对应的 4 位二进制数表示，就转换为它的二进制数。

例如：

$$(7 \quad A \quad 3)_{16}$$
$$\downarrow \quad \downarrow \quad \downarrow$$
$$(0111) \quad (1010) \quad (0011)$$

即　　$(7A3)_{16}=(11110100011)_2$

4．二进制数转换为八进制数　从最低位开始，向左每 3 位分为一组，不够 3 位的用 0 补足 3 位，将每组的二进制数按对应的八进制数写出，就转换成了八进制数；二进制小数，则从小数点开始向右每 3 位划为一组，不够 3 位，用 0 补足 3 位，写出其对应的八进制数。

例如：　　$(11101001110.1101)_2$

分组为　　（011）　（101）　（001）　（110）．（110）　（100）

对应的八进制数为：　3　5　1　6　.　6　4

即　　$(11101001110.1101)_2=(3516.64)_8$

5．二进制数转换为十六进制数　从最低位开始，向左每 4 位分为一组，不够 4 位的用 0 补足 4 位，将每组的二进制数按对应的十六进制数写出，就转换成了十六进制数；二进制小数，则从小数点开始向右每 4 位划为一组，不够 4 位，用 0 补足 4 位，写出其对应的十六进制数。

例如：　　$(1011011.01101)_2$

分组为　　（0101）　（1011）　（0110）　（1000）

对应的十六进制数为：　5　B　6　8

$$(1011011.01101)_2=(5B.68)_{16}$$

四、编码

我们用 0、1 来表示二进制数，是数的一种编码表示。编码的方法在我们生活中也是常用的，如通信中的邮政编码、电话号码、电报中的莫尔斯码、身份证代码、学号代码、汉字的国标码、区位码等等。

在计算机中只给数的表示进行了编码还不够，要处理数的符号是正还是负怎么办？处理英文字符、加减乘除以至汉字又怎么办？都得用 0、1 来给它们编码。现在最通用的一种给字符编码的是 ASCII 代码（即 American Standard Code for Information Interchange　的缩写），例如英文字母 A，用 8 位二进制数 01000001 表示，换成十六进制表示为$(41)_{16}$，各种字符的 ASCII 代码，列在附录 I 中。

总之，计算机只能识别 0、1 两个符号。和计算机沟通信息，就要以这两个最简单的符号为基础，对各种信息形式进行变换，把计算机不能直接识别的信息变换为计算机能识别的信息。

§1.3 计算机的机器语言和高级语言

一、机器语言

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。人要和机器交换信息就要解决一个“语言”的问题。计算机并不懂人类的语言（无论是中文或英文），例如，我们写 A+B=C，机器不能接受，它只能识别 0 和 1 两种状态。如光电输入机中纸带有孔的地方，它代表 1，无孔的地方，代表 0，由 0 和 1 组成各种排列组合，通过线路转变成电信号，让计算机执行各种不同的操作。

这种直接用 0 和 1 组成的机器指令编写的程序，就是机器语言源程序。对计算机来说，这是它唯一能直接“听”得懂的语言。所以，常常称之为面向机器的语言。但是，对使用计算机的人来说，这是十分难懂的语言，它难读、难记、难写，容易出错，不同机型又不通用。显然人和机器之间的通信存在巨大的鸿沟，只有填补上这个鸿沟，使用的人愈是方便容易，机器又能懂得，计算机才能发挥更大的作用。为此，人们研究了一种汇编语言。

二、汇编语言

把用二进制数表示的指令，用一些符号来表示，如用表示操作的英文缩写来代替指令代码，用十六进制数表示数字。在§1.1 节我们用机器语言编写的 5+6 这一过程，就可写为如下形式：

```
LD      A, (2BH)
LD      HL,(2CH)
ADD     A,(HL)
LD      (2DH),A
```

第一条 LD 即 load 的缩写，表示“取数”的操作，A 表示累加器，（2BH）括号内的十六进制数是内存地址。它的含义是把存放在内存第 43 号地址的数（已存放有数“5”）取出来，放到累加器 A 中。

第二条 LD 仍为取数，HL 表示一个暂时存放数据的寄存器名，它的含义是把内存地址号 44（表示为十六进制数 2CH），放在寄存器 HL 中。

第三条 ADD，是“加”的意思，操作数中指出把 A 中和 HL 所指示的 44 号地址中存放的数相加（即将 5 和 6 相加，结果为 11），并把结果放在 A 中。

第四条 LD，送数，把计算结果从 A 中送到内存地址为 45（表示为十六进制数 2DH）的存储单元中存放。

这种用符号代替后的指令，就叫汇编语言，又称符号语言，像 LD、ADD 等这类符号称为指令符号或助记符。用汇编语言编写的程序，称为汇编语言源程序，常简称为汇编语言程序。

这种语言，相对机器语言就容易读、容易记、容易写了。但是，机器却不能识别。因此，计算机是无法直接执行的。一个不懂汉语的外国人到中国来要和中国人直接交谈，那是无法进行对话的，所以，只好求助于翻译。在计算机中，也同样采取这种方法，人们编写程

序用汇编语言，然后请一位翻译，把汇编语言程序翻译成机器能懂得的机器语言程序，这个翻译过程，叫做“汇编”。汇编后产生的机器代码称为目标程序。翻译可以由人手工完成，但做起来既繁琐单调，又容易出错。实际上，我们是让计算机来做。因而就研制出了担任翻译的程序，取名叫汇编程序。汇编过程如图 1.2 所示。

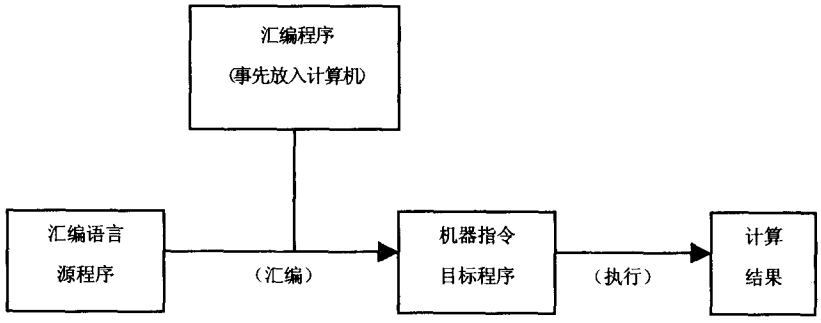


图 1.2 汇编过程

汇编语言使程序设计工作前进了一大步，但是仍然存在很多缺点，第一、不便于我们求解问题过程的描述，如一个数学公式，汇编语言的表达形式与人们的习惯表达形式差距很大；第二、它仍是面向机器语言，不同机型，汇编语言也不一样，因此用它编制的程序，没有通用性。为了克服这些不足之处，人们进一步研制出了高级语言。

三、高级语言

它是用更接近人的自然语言和数学表达式的一种语言，由表达不同意义的“关键字”和“表达式”按照一定的语法语义规则组成，完全不依赖机器的指令系统。这样的高级语言为人们提供了很大的方便，编制出来的程序易读易记，也便于修改、调试，大大提高了编制程序的效率，也大大提高了程序的通用性，便于推广交流，从而极大的推动了计算机的普及应用。

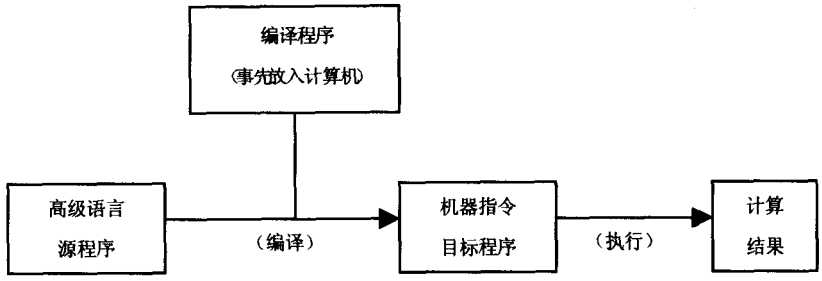
例如使用高级语言 BASIC，要想得到 $3\times8\times SIN(\pi/2)$ 的结果，只需写出 print 3*8*sin(3.14159/2)即可，计算机将计算并输出结果。

我们看到高级语言离人们的理解愈加接近了，但离计算机的理解就愈远了。计算机是不能直接理解那些英语单词、数学表达式的。所以，为了填补人机之间的鸿沟，还是得求助于翻译。这种“翻译”，通常有两种做法，即编译方式和解释方式。

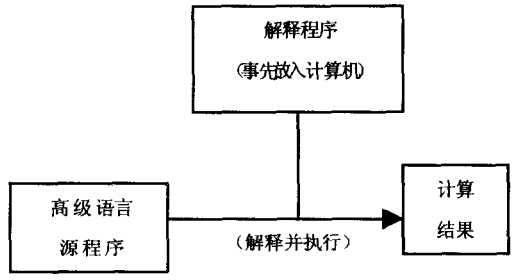
编译方式是：事先编好一个称为编译程序的机器指令程序，并放在计算机中，把用高级语言编写的源程序输入计算机，编译程序便把源程序整个翻译成用机器指令表示的目标程序，然后执行该目标程序，得到计算结果。如图 1.3a)所示。

解释方式是：事先编好一个称为解释程序的机器指令程序，并放在计算机中，把用高级语言编写的源程序输入计算机，它并不像编译方式那样把源程序整个翻译成用机器指令表示的目标程序，而是逐句地翻译，译出一句立即执行，即边解释边执行。见图 1.3b)。这种方式比编译方式费机器时间，但少占计算机的内存。

FORTRAN、ALGOL、COBOL 等高级语言采用编译执行方式，而大多数 BASIC 高级语言采用解释执行方式。



a)编译过程



b)解释过程

图 1.3

由于编译（解释）程序代替了人工把高级语言源程序翻译为机器指令程序，大大节约了使用者的工作量。自从有了高级语言后，一般的科技人员和大、中学生以及职工，都能很快地学会使用计算机，可以完全不顾什么机器指令，也可以不必深入懂得计算机的内部结构和工作原理，就能方便地使用计算机进行各种科学计算或事务处理等。因此有人说，高级语言的出现是计算机发展中“最惊人的成就”。

目前，世界上已有一百多种高级语言，最流行的有几十种，例如：

1. FORTRAN（Formula Translator 的缩写）语言，它是世界上最早出现的高级语言，从 1954 年问世以来，经过几次大的发展，功能有很大的增强。现在流行的是 FORTRAN 77 版本。它特别适于科学、工程计算。
2. COBOL(Common Business Language 的缩写)语言,适于非数值计算的商业、管理领域。
3. PASCAL 语言是最早出现的结构化语言，适合于计算机教育。
4. PL/1 语言是一种大型语言，功能强，数值计算和数据处理均适用。
5. Ada 语言是一种工程化的大型语言，适合于大型软件工程。
6. C 语言是近年来广泛推广的结构化语言，适于编写系统软件。
7. BASIC 语言是一种简单会话式语言，在世界上应用最广泛。

§1.4 算法与程序设计

学完前几节后，我们知道了计算机所做的工作就是按指定的步骤执行一系列的操作，以完成某一特定的任务。因此，要计算机为我们做事，就必须事先为它设计出这一系列的操

作步骤，并用计算机语言写成程序，这就是程序设计。解决问题的方法和步骤，称之为算法，它是程序设计的关键，因此，在学习用 C 语言进行程序设计之前，了解一点算法的知识是必要的，它是后续章节学习的基础。

一、什么是算法

计算一个算术式，要先乘除后加减，这个规则就是算法。使用算盘，珠算口诀就是算法。在日常生活中做任何一件事情，都是按照一定规则，一步一步进行，比如乐队演奏、厨师炒菜，都有各自的操作步骤，这就是算法。在工厂中生产一部机器，先把零件按一道道工序进行加工，然后，又把各种零件按一定法则组装成一部完整机器，它们的工艺流程就是算法；在农村中种庄稼有耕地、播种、育苗、施肥、中耕、收割等各个环节，这些栽培技术也是算法。总之，把任何这些数值计算的或非数值计算的过程中的方法和步骤，都称之为算法。

计算机解决问题的方法和步骤，就是计算机的算法。计算机用于解决数值计算，如科学计算中的数值积分、解线性方程等的计算方法，就是数值计算的算法；用于解决非数值计算，如用于管理、文字处理、图象图形等的排序、分类、查找，就是非数值计算的算法。

下面举几个简单例子，以说明计算机算法。

[例 1.1] 将两个变量 X和 Y 的值互换。设 X=5，Y=10。

问题分析：两人交换座位，只要各自去坐对方的座位就行了，这是直接交换。一瓶酒和一瓶醋互换，就不能直接从一个瓶子往另一个瓶子倒。必须借助于一个空瓶子，先把酒倒入空瓶，再把醋倒入已倒空的酒瓶，最后把酒倒入已倒空的醋瓶，这样才能实现酒和醋的交换，这是间接交换。

计算机中交换两个变量的值不能用两个变量直接交换的方法，而必须采用间接交换的方法。因此，设一中间变量为 Z。

算法表示如下：

S1 将 Y 值存入中间变量 Z： Y→Z

S2 将 X 值存入变量 Y 中： X→Y

S3 将中间变量 Z 的值存入 X 中：Z→X

这里 S1、S2、…即 Step1、Step2…的简写，用以标识执行的步骤，以后的例中均用此符号，不再说明。

用一个示意图说明此算法，如图 1.4 所示。

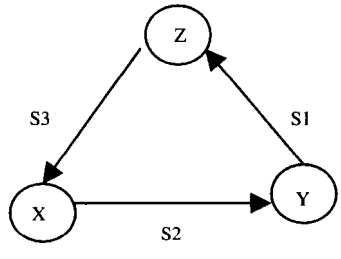


图 1.4 示意图

[例 1.2] 求 5 个自然数的和 $\sum_{n=1}^{5} n$（即 S=1+2+3+4+5）。

问题分析：该题有两个特点：(1)重复执行加法，每加一个数，总和的值都在变；(2）加数是一个有规则的等差数列，第 1 项是 1，以后每加一次，加数就增加 1,因此,可以用以下算法实现。

算法 1：

S1 设一存放累加和的变量 S，初值置 0，即 0→S；设一计数变量 N,初值置 0，即 0→N。

S2 计算和 S+N→S，并把计数变量增值 N+1→N。

S3 判断：当 N≤5 时，返回第 2 步 S2，再次求和；当 N>5 时，顺序执行下一步 S4。

S4 输出结果，S 为所求之和。

算法 2：

S1 0→S,1→N（同算法 1)。

S2 判断:当 N≤5 时，顺序执行下一步 S3;当 N>5 时,跳过 S3、S4,执行第 5步 S5

S3 S+N→S，N+1→N(同算法 1)。

S4 返回第 2 步 S2。

S5 输出结果，S 为所求之和。

在这个算法里，出现了重复求和的操作，称为循环，这种循环必须用条件加以限制，让它进行有限次数就停止，否则，成为死循环。上述算法 1 的 S3 是条件判断,求和的操作是先执行，后判断；算法 2 的 S2 是条件判断，求和的操作是先判断，后执行。

[例 1.3] 计算 N!（即求 1×2×3×4×5×6×…×N ）的值。

问题分析：该题也有两个特点：（1）重复执行乘法的操作，每乘一次，积都在变；（2乘数是一个有规则的等差级数，后项是前项加 1，因此，可用以下算法实现。

S1 指定一个具体的 N 值，如 5→N。

S2 设一累乘变量 M，初值置 1；设一计数变量 P，初值置 1。

S3 求积 M×P→M，计数变量 P 增值 P+1→P。

S4 判断：当 P<N 时，返回第 3 步 S3；否则，往下执行 S5。

S5 输出结果，M 为所求之积。

[例 1.4] 求两个自然数 M 和 N 的最大公约数。

问题分析：最大公约数就是能同时整除 M 和 N 的最大正整数，这是一个古老的算术问题，这里介绍欧几里得（Euclid）算法。

S1 输入两个自然数。

S2 求余数：M/N,得到余数 R（0<R<N)。

S3 置换：N→M, R→N。

S4 判断：当 R≠0，返回第 2 步 S2；当 R=0,顺序执行第 5 步 S5。

S5 输出结果，M 为所求最大公约数。

综合以上算法示例看出，算法是解题方法的精确描述。算法并不给出问题的精确的解，只是说明怎样才能得到解。每一个算法都是由一系列的操作组成的。这些操作包括加、减、乘、除、判断、置数等，按顺序、分支、重复等结构组成。所以研究算法的目的就是研究怎样把各种类型的问题的求解过程分解成一些基本的操作。

算法写好之后，要检查其正确性和完整性，再根据它编写出用某种高级语言表示的程序。程序设计的关键就在于设计出一个好的算法。所以，算法是程序设计的核心。

二、算法的特性

从上面的例子中，可以概括出算法的 5 个特性：

（1）算法中执行的步骤总是有限次数的，不能无休止地执行下去，称之为有穷性。例如计算圆周率 π 的值，可用如下公式：

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

这个多项式的项数是无穷的，因此，它是一个计算方法，不是算法。我们要计算 π 的值，只能取有限个项数。例如，取精确到第 5 位，那么，这个计算就是有限次的，因而才能称得上算法。

（2）算法中的每一步操作的内容和顺序必须含义确切，称之为确切性。不能有二义性。

（3）算法中的每一步操作都必须是可执行的，称之为有效性。

（4）要有数据输入。算法中操作的对象是数据，因此，应在操作前提供数据。

（5）要有结果输出，算法的目的是用来解决一个给定的问题，因此，它应向人们提供产生的结果，否则，就没有意义了。

了解算法的这些特性，对我们在程序设计中构造一个好的算法，是有重要指导意义的。

§1.5 用流程图表示算法

描述算法有多种不同的工具，采取不同描述算法的工具对算法的质量有很大的影响。如上节中例 1 至例 4 是用自然语言——汉语描述的算法。使用自然语言描述算法的最大优点在于人们比较习惯，容易接受。但也确实存在着很多缺点，一是容易产生二义性，例如，中文“走火”二字，既可表达子弹从枪膛射出，又可表示电线上漏电，也可表示说话漏了嘴；二是比较冗长；三是在算法中如果有分支或转移时，用文字表示就显得不够直观；四是计算机不便于处理，所以自然语言不适合描述算法。在计算机中常用流程图、结构化流程图、计算机程序设计语言等描述工具来描述算法。

流程图，亦称框图，它是用一些几何框图、流程线和文字说明表示各种类型的操作。流程图中的基本图形、图形意义和长度比例都有国家颁布的标准（GB ISO5807—85），如图 1.5 所示。

处理框亦称矩形框，有一个入口，一个出口。判断框亦称菱形框，有一个入口，两个出口。在框内写上简明的文字或符号表示具体的操作，用带箭头的流向线表示操作的先后顺序。

流程图是人们交流算法设计的一种工具，不是输入给计算机的。只要逻辑正确，人们都能看得懂就可以了，一般是由上而下按执行顺序画下来。用流程图表示例 1～例 4 的算法，如图 1.6～1.9 所示。

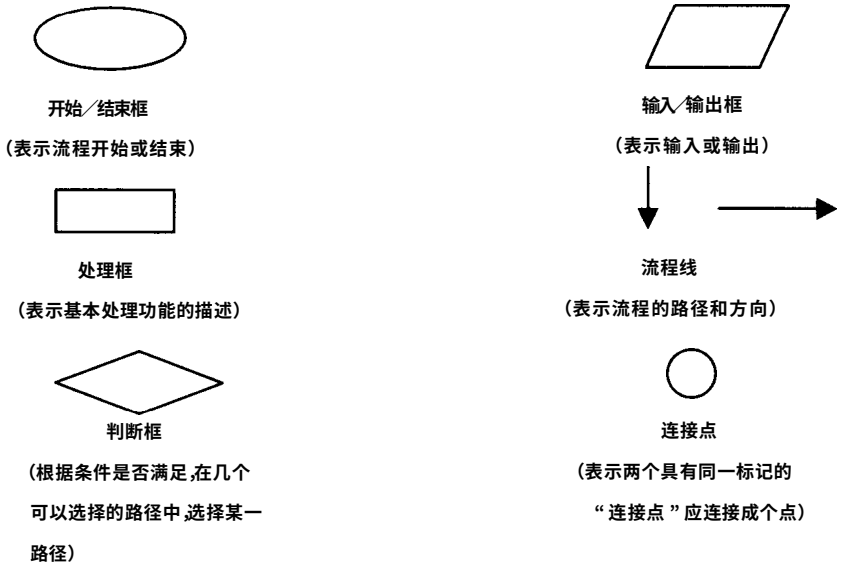


图 1.5 流程图中的符号

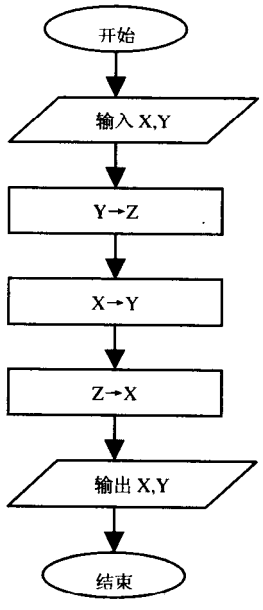


图 1.6 交换变量流程图

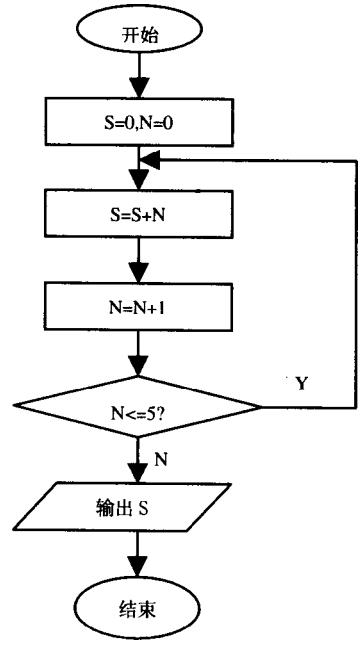


图 1.7 求 5 个自然数和的流程图

从上面的 4 个例子看出，流程图的优点是形象直观，清晰明了，所以它很早就被广泛采用在各种高级语言的程序设计中，常常称之为传统的流程图。

但是它也有不足之处，对于较复杂的问题，占面积大，而且由于使用流程线，使流程任意转移，容易使人弄不清流程的思路。

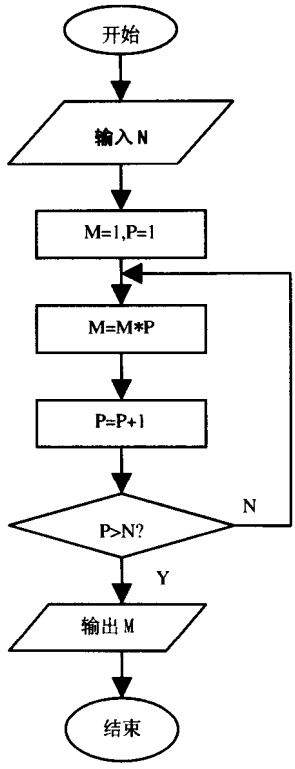


图 1.8　求 N!流程图

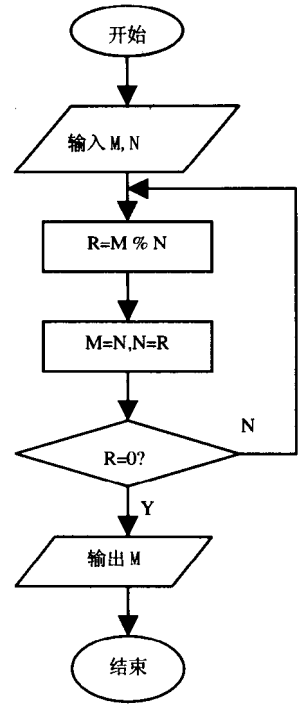


图 1.9　求最大公约数流程图

§ 1.6　用 N—S 结构化流程图表示算法

一、什么是结构化程序

随着计算机的发展，编制的程序越来越复杂。一个复杂程序多达数千万条语句，而且程序的流向也很复杂，常常用无条件转向语句去实现复杂的逻辑判断功能。因而造成质量差，可靠性很难保证，程序也不易阅读，维护困难，60 年代末期，国际上出现了所谓“软件危机”。

为了解决这一问题，就出现了结构化程序设计，它的基本思想是象玩积木游戏那样，只要有几种简单类型的结构，可以构成任意复杂的程序。这样可以使程序设计规范化，便于用工程的方法来进行软件生产。基于这样的思想，1966 年意大利的 Bobra 和 Jacopini 提出了三种基本结构，由这三种基本结构组成的程序，就是结构化程序（Structured Program）。

二、三种基本结构

1．顺序结构

程序流向是沿着一个方向进行的，有一个入口（A），有一个出口(B)，如同 1.10 所示。顺序结构是最简单的一种结构。先执行 A 块然后再执行 B 块，A 块和 B 块分别代表某些操

作。

2. 选择结构

程序的流程发生分支，根据一定的条件选择其中之一，这就是选择结构，它也有一个入口（A），一个出口（B）。简单的选择结构是只有两个分支的情况，如图 1.11 所示。根据

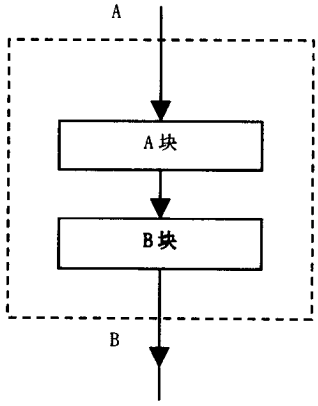


图 1.10 顺序结构

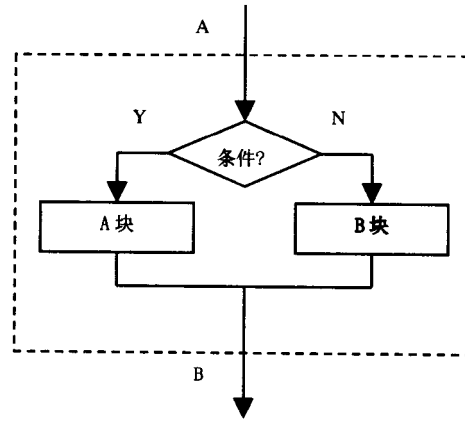


图 1.11 选择结构

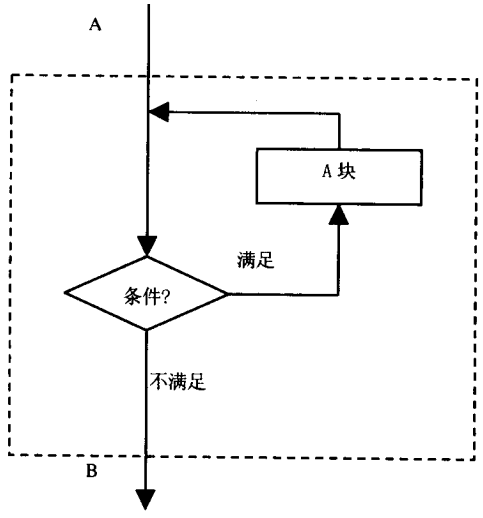
给定的条件是否满足决定执行 A 块或 B 块。

3. 循环结构（亦称重复结构）

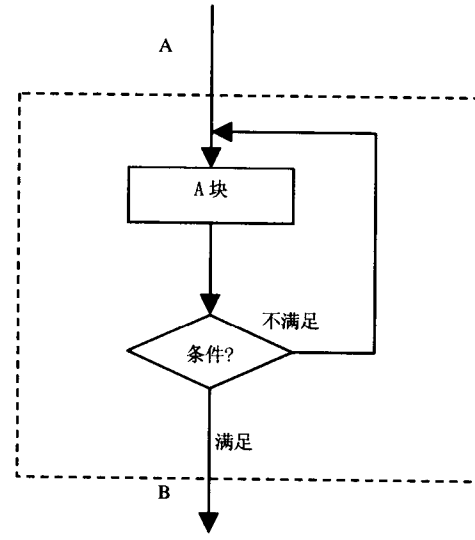
程序流程是有限次重复执行某一块程序，再退出循环。循环结构有两种类型：

（1）当型（WHILE）循环结构，先判断条件是否满足，若满足时就执行循环体，若条件不满足时就不执行循环体，并转到出口。如图 1.12a)所示。

（2）直到型（UNTIL）循环结构，它是先执行循环体，后判断。当条件不满足时继续执行循环体，条件满足时，停止执行，并转到出口。如图 1.12b)所示。



a）当型循环结构



b）直到型循环结构

图 1.12 循环结构

循环结构也有一个入口（A），一个出口（B），它应当使重复处理为有限次，不能无限

制的循环下去。

这种结构化程序具有以下特点：

（1）只有一个入口和一个出口。

（2）无死语句，即没有永远都执行不到的语句。

（3）无死循环，即无永远执行不完的循环。

三、结构化流程图（N—S图）

前面的图 1.10 至图 1.12，用框图表示了结构化程序的三种基本结构。看起来还清楚，但情况复杂时，图形中的流程线多，仍不便于阅读。所以美国学者 I.Nassi和 B.Schneiderman 于 1973 年提出了一种新的绘制流程图的方法。由于他二人的名字以 N和 S 开头，故把这种流程图称为 N—S 图。这种结构化流程图，完全去掉了在描述中引起混乱的带箭头的流程线，全部算法由三种基本结构的框表示。

三种基本结构框的画法规定如下：

1. 顺序结构（Sequence Structure）

它由若干个前后衔接的矩形块顺序组成。一个顺序结构，如图 1.13，先执行 A 块，然后执行 B块。各块中的内容表示一条或若干条需要顺序执行的操作。

2. 选择结构（Choice Structure）

见图 1.14，在此结构内有两个分支，它表示当给定的条件满足时执行 A 块的操作，条件不满足时，执行 B 块的操作。

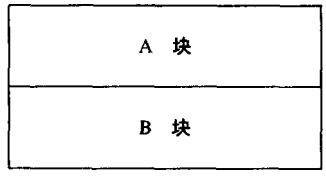


图 1.13 顺序结构

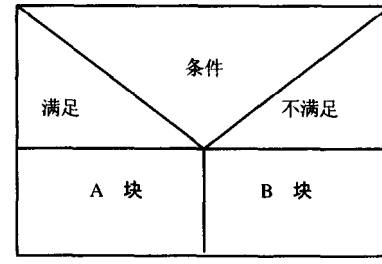
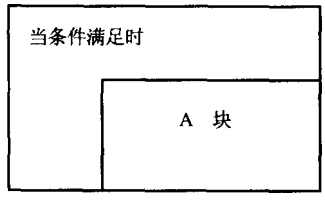


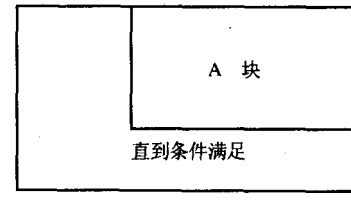
图 1.14 选择结构

3. 循环结构（Repetition Structure）

（1）当型（WHILE）循环结构，如图 1.15a)。先判断条件是否满足，若满足就执行 A 块（循环体），然后再返回判断条件是否满足，如满足再执行 A 块，如此循环下去，直到条件不满足为止。



a)当型循环结构



b)直到型循环结构

图 1.15 循环结构

（2）直到型（UNTIL）循环结构，如图 1.15b)所示。先执行 A 块（循环体），然后判断条件是否满足，若不满足则返回再执行 A 块，若满足则不再继续执行循环体。

用 N—S 图表示例 1 至例 4 中的算法，见图 1.16 至图 1.19。

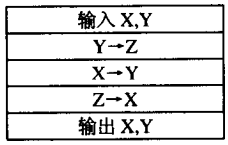


图 1.16 交换变量 N—S 图

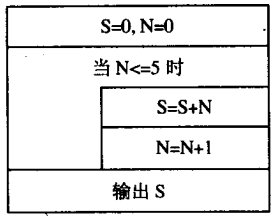


图 1.17 求 5 个自然数和 N—S 图

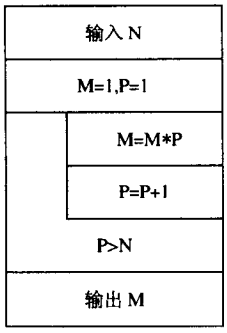


图 1.18 于求 N！N—S 图

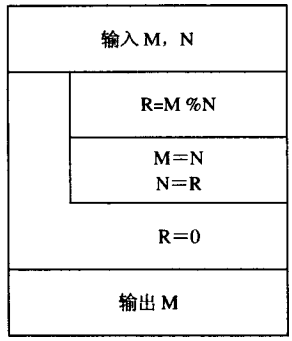


图 1.19 求最大公约数 N—S 图

将图 1.16 至图 1.19 的 N—S 图与图 1.10 至图 1.11 的传统流程图相比较，明显看出：N—S 流程图是由基本结构单元组成的，各基本结构单元之间是顺序执行关系，即从上到下，一个结构一个结构地顺序执行下来。这样的程序结构，对于任何复杂的问题，都可以很方便地用以上三种基本结构顺序地构成。因而它描述的算法是结构化的，这是 N—S 图的最大优点。

用 N—S 图表示算法，思路清晰，阅读起来直观、明确、容易理解，大大方便了结构化程序的设计，并能有效地提高算法设计的质量和效率。对初学者来说，使用 N—S 图还能培养良好的程序设计风格，因此本书在表示算法时，主要采用了这种 N—S 图。

四、结构化程序设计步骤

在学习编写程序之前，对结构化程序设计的全过程有个全面的了解，从中可以知道用计算机语言编写程序在全过程中的地位，这对培养和提高程序设计的能力很有好处。

完成一个正确的程序设计任务，一般可分为以下几个步骤进行，如图 1.20 所示。

1. 提出和分析问题　即弄清提出任务的性质和具体要求。例如，提供什么数据，得到什么结果，打印什么格式，允许多大误差，都要确定。若没有详细而确切的了解，匆忙动手编程序，就会出现许多错误，造成无谓的返工或损失。

2. 构造模型　即把工程中或工作中实际的物理过程，经过简化，构成物理模型，然后，用数学语言来描述它，这称为建立数学模型。