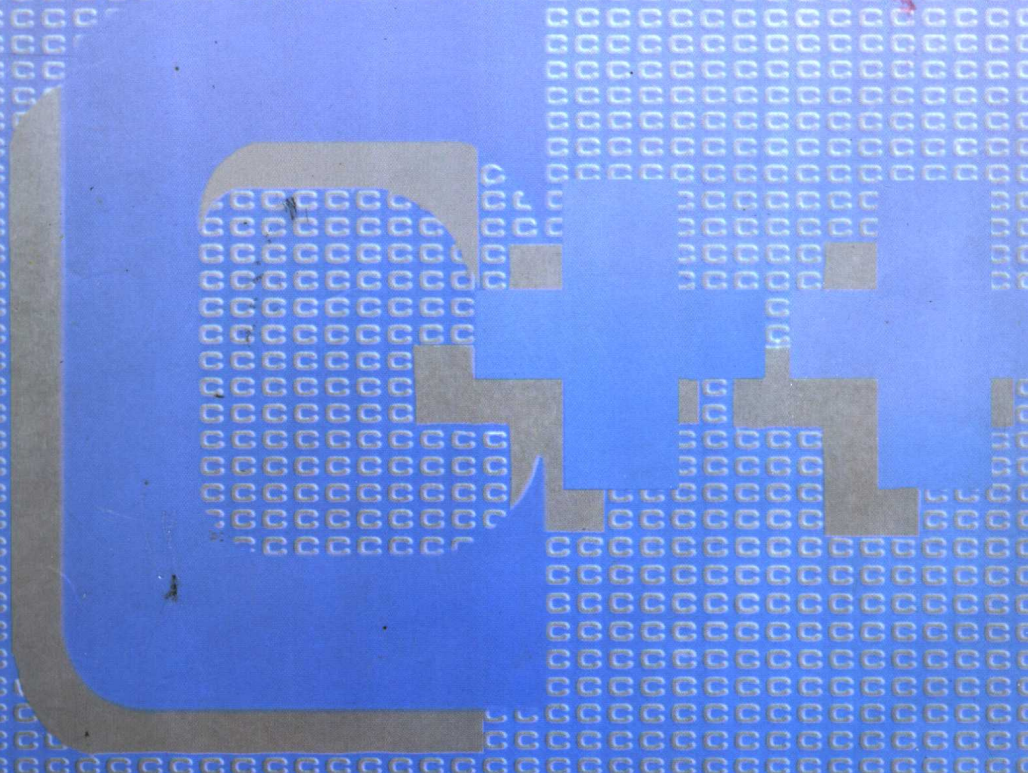
高等学校电子信息类教材

计算机软件技术基础
——C++/C程序设计
（第二版）

周佩德　柏　毅　编著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL:http://www.phei.com.cn

高等学校电子信息类教材

计算机软件技术基础——C++/C程序设计(第二版)

周佩德　柏　毅　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内容简介

如果你准备学习 C 语言,不如开始学习 C++,因为 C++ 是一个更好的 C。C++ 语言在 C 语言的基础上增加了类的概念,支持面向对象的程序设计。

本书以程序设计的初学者为对象,系统讨论 C++ 语言的程序设计的方法,同时结合介绍关于算法和数据结构等方面的基本知识。全书每一章都按由浅入深的原则含有大量例题,章末的习题供读者加深对基本概念的理解和上机实习。

本书是一本用 C++ 语言进行基础程序设计语言教学的教科书,亦可作为计算机软件技术基础课程的教材。本书采用将 C++ 语言和 C 语言对照介绍的方法,所以也适用于用 C 语言进行基础程序设计语言的教学要求。

书　　名:计算机软件技术基础——C++/C 程序设计(第二版)
编 著 者:周佩德　柏　毅
责任编辑:陈晓明
排版制作:电子工业出版社计算机排版室
印 刷 者:北京大中印刷厂
出版发行:电子工业出版社　URL:http://www.phei.com.cn
北京市海淀区万寿路 173 信箱　邮编 100036
经　　销:各地新华书店经销
开　　本:787×1092 1/16　印张:20 字数:506 千字
版　　次:1998 年 9 月第 2 版　1998 年 9 月第 1 次印刷
书　　号:ISBN 7-5053-4861-2
G·389
定　　价:26.00 元

前　言

二十一世纪只有咫尺之遥，新世纪信息化社会的图景已开始展现在我们面前。培养所有大学生、特别是工科大学生在计算机方面的开发能力是大学教育必须完成的任务。根据国家教委对计算机基础教育的要求，在学生掌握了计算机文化基础的知识后，需要接受计算机技术基础的训练，其中程序设计是软件技术基础的重要内容。计算机程序设计课程的教学目的是使学生掌握计算机程序设计的概念和方法，对只具有计算机文化基础的大学生，该课程的教学必然需要结合一种程序设计语言进行。由于 C 语言和其他程序设计语言相比更具有广泛的实用性，C 语言也具有计算机高级语言的基本特征，所以许多学校在计算机程序设计的教学中正逐步用 C 语言取代 Pascal 等语言作为基础教学语言。在 90 年代提出面向对象的概念以后，C 语言发展为 C++ 语言，面向过程的程序设计方法已过渡到面向对象的程序设计。本书根据这个软件技术发展的背景、推荐直接用 C++ 语言开始进行计算机技术基础的教学。考虑到学生总体的计算机基础，本课程定位在 C++ 的基础部分，即 C++ 中的 C，然后以一章的篇幅介绍面向对象的基本概念和面向对象的程序设计方法。同时本书力图能将程序设计的教学和基本的软件知识的教学有机结合起来，在系统讨论程序设计的方法的同时，结合介绍关于算法、数据结构和软件工程等方面的基本知识。迅速使受教育者既能具备软件开发的实际能力，又能掌握计算机软件领域的一些必要的基本知识。

本课程的先修课程是计算机文化基础。通过计算机文化基础的学习，学生应当掌握计算机系统的基础知识，包括计算机组成、工作原理以及计算机操作系统和文字处理等方面知识，并具备一定的计算机操作能力。课程参考学时为 48 学时到 60 学时，同时要安排不少于 24 小时的上机实习时间，全书每一章都按由浅入深的原则组织了大量例题，章末的习题供读者回顾基本概念以及上机实习。课程结束后最好安排一个大型的课题实习，将软件工程方面的教学结合设计任务进行。不同系科根据对计算机语言的要求并视学生的接受程度可对教材中的材料进行适当增减，例如精简关于面向对象方面的内容和数据结构方面的内容等。在教学内容的改革中应当结合进行教学方法和教学手段的改革，推荐采用联机投影仪或联网教学等方法，以加大课堂教学的信息量并改进教学效果。考虑到国家（或省市）计算机二级等级考试的现状，以及用 C 语言开发的实际需要，本书在讨论 C++ 语言的同时也介绍了 C 语言的相应内容，所以学生可以顺利通过国家（或省市）的计算机二级（C 语言）等级考试。

作者周佩德教授和柏毅副教授，近年来承担了东南大学强化班和部分系的计算机基础类课程的教学任务，现将教学中采用的教材《计算机软件技术基础》（电子工业出版社 1996 年版）经过归纳整理重新出版。在本书的修订过程中，柏毅老师做了大量的工作，并承担了第 9、10 章的编写任务。东南大学有若干系选择 C++（或 C）语言作为计算机教学的基础语言，他们的工作对作者有很多启迪。96 强化班的吴丹同学测试了本书修订版的全部例题。

本书由东南大学计算机科学与工程系邢汉承教授审阅。作者对关心和支持本书出版的所有同志表示衷心的感谢。同时作者借此机会对原书中的错误向相关同志表示深深的歉意。对于修订版的错误和不足之处，仍请各位同仁以及广大读者批评指正。

编著者

1998 年 4 月

目　录

第1章 预备知识

计算机是信息处理的工具，计算机的出现将人类的思维活动推向一个更高的阶段。思维活动可以利用语言来形式化，而语言层次可以离开意识层次相对独立地活动。计算机语言作为人和计算机之间进行意识交流的工具，人通过计算机语言将意识活动交给计算机进行独立的加工，产生进一步的思维活动，所以可以认为计算机是人类思维的工具，计算机思维是一种物化的思维，是人脑思维的进一步延伸。

在计算机语言层次中，人与计算机的意识活动的交流是通过程序设计这个环节来完成的。1976 年 N. Wirth 出版了一本名为《Algorithms + Data Structure = Programs》的著作，明确提出算法和数据结构是程序的两个要素，也就是说程序设计主要包括两方面的内容：行为特性的设计和结构特性的设计。行为特性的设计是指完整地描述问题求解的全过程，并精确地定义每个解题步骤，这一过程即是算法的设计；而结构特性的设计是指在问题求解的过程中，计算机所处理的数据以及数据之间联系的表示方法。

本章预备知识包括对算法简单的讨论、对逻辑代数基础知识的回顾、以及结合程序设计语言的发展介绍程序设计思想的沿革，最后通过一个简单的 C++（及等价的 C 程序）程序来说明程序设计的基本步骤。

1.1 算 法

1.1.1 算法的概念

粗略地说，算法是解决问题的流程安排。解决任何一个问题，小到冲制一杯速溶咖啡，大至联合国召开一次安理会会议，都必须有一个工作步骤的安排，例如冲咖啡时必须先准备好杯子，再准备开水，开水冲入咖啡后对不同的客人还会有是否加糖及是否加牛奶等不同的要求。工作过程的描述就是一个算法，对这个原始算法的描述采用了自然语言的方法。

在欧几里德的《几何原本》里，阐述了求两个数的最大公因子的过程，也称欧几里德算法。

欧几里德算法：给定两个正整数 m 和 n，求它们的最大公因子，即同时能够整除 m 和 n 的最大正整数。

Step1. 以 n 除 m，并令 r 为所得余数。（显然 n > r >= 0）。

Step2. 若 r = 0，算法结束，n 即 为 m 和 n 的最大公因子。

Step3. 置 m 为 n、n 为 r（记为 m ← n，n ← r），返回 Step1。

在描述欧几里德算法的时侯，我们采用了一种比自然语言更抽象一些的方法，或者称为半自然语言的方法。同时欧几里德算法的描述与冲制速溶咖啡的算法相比，具有完全的确定性，即每一步骤有确定的含义。在欧几里德算法的描述中引入了一些变量，如 m、n、r，引入了关系运算符 =、>=，还引入了赋值记号 ←。在程序设计中，变量是个很重要的概念，程序执行中每个变量将被分配在计算机内存中的一个确定存储区域内，根据变量名在程序中的上下文，一个变量名可能代表的是内存中所分配的存储空间，也可能代表的是在这个空间中所存储的数

据，因此变量名有变量的存储空间和在该空间中变量的数据这两方面的含义。赋值是指将赋值号“←”右边变量的存储单元中的值送到左边的变量所代表的存储空间中去。

还需要注意的是，在算法中不仅各步骤间的顺序是重要的，在每步中的动作次序同样也是重要的。例如在欧几里德算法的 Step3 中，“置 m←n，n←r”绝不能写成“置 n←r，m←n”，因为这样做的结果是在 m、n 和 r 中最后都存储了同一个值 r，即都变成了 Step1 除法运算的余数。

由于算法是对实际运算的抽象表述，同时算法描述的往往是比冲制咖啡复杂得多的运算过程，所以一个算法的每一步的含义（或者说是作用）往往不是一目了然的。学习复杂算法的最好方法莫过于将该算法的实例代入到算法中去，按算法的步骤去执行算法，这种由人来执行算法的方法称为“人工模拟”。人工模拟也是开发和检验算法的一种重要手段，通过人工模拟能了解算法每一步的实际含义。算法中一般都包含了“如果…则…”的步骤，规定对不同实例的不同处理途径，所以为深刻弄懂一个算法，一般还需要设计多个实例来反复体会算法中各种不同处理路径的作用。

对欧几里德算法，第一个实例是：

m = 12 和 n = 4

Step1. m/n = 12/4 = 3，余数为 0

Step2. 由于余数 r = 0，算法结束，4 为 12 和 4 的最大公因子。

由于这个实例没有“走”到 Step3，所以由这个实例不能全面了解欧几里德算法。第二个实例是：

m = 48 和 n = 14

Step1. m/n = 48/14 = 3，余数为 6（r=6）

Step2. 余数 r 不为零，进入下一步

Step3. 置 m←n、n←r，即 m = 14、n = 6，返回 Step1。

Step1. m/n = 14/6 = 2，余数为 2（r=2）

Step2. 余数 r 不为零，进入下一步

Step3. 置 m←n、n←r，即 m = 6、n = 2，返回 Step1。

Step1. m/n = 6/2 = 3，余数为 0

Step2. 由于余数 r = 0，算法结束，n 的现值 2 即为 48 和 14 的最大公因子。

对欧几里德算法的跟踪应当可以告一段落了，但再深入地想一想，欧几里德算法的上述表述是否尚有可改进之处呢？下一个实例是：

m = 14 和 n = 48

Step1. m/n = 14/48 = 0，余数为 14

Step2. 由于余数 r = 14，进入 Step3

Step3. 置 m←n、n←r，即 m = 48、n = 14，返回 Step1。

我们可以发现，当 m 小于 n 时，经过 Step1、Step2 和 Step3 后，仅将 m 和 n 互换了一次。所以，可以为欧几里德算法增加 Step0：

Step0. 如果 m<n，则 m ↔ n，（用符号↔表示交换两个变量中的值）

这样尽管欧几里德算法由三步变为四步，但对 m<n 的实例，实际减少了算法的执行时间。而按 m 和 n 之间大小关系的出现几率考虑，m 小于 n 和 m 大于 n 这两种情况出现的或然率各为一半。算法步的数目未必能直接反映算法实际执行所需的时间。

至此可以给算法一个更精确的定义：一个算法是一个有穷规则的集合，这个规则规定了解

决一个特定问题的运算序列。作为一个算法必须具备以下五个特性。

1. 有穷性

一个算法必须总是在有穷步之后结束。

欧几里德算法满足这个条件。因为在 Step1 以后，r 的值肯定小于 n，所以如果 r 不等于零，则在经过 Step3 将 r 的值赋给 n 以后，重新进行 Step1 时 n 的值已经减小，正整数的递减序列必然最后终止为 0。所以对任何给定的 m 和 n 的值，Step1 只会执行有穷次。对足够大的 m 和 n，可能算法执行的次数相当多，但即使达到天文数字，仍然是有穷的。

如果一个计算不具有有穷性但具有算法的其他特性，则可称其为计算方法。例如对无穷调和级数求和的计算就不是一个算法而是一个计算方法，在确定了项数或确定了对精度的要求以后，调和级数求和才能满足有穷性的要求，也就是说，这时对调和级数求和的计算才可能成为一个算法。

2. 确定性

算法的每一个步骤都必须有确定的定义。

欧几里德算法的每一步都是确定的。例如在 Step1 中，除法的算术运算法则保证了两个正整数相除的方法是确定的，而结果的商和余数也都是确定的。

按照确定性的要求，冲制一杯速溶咖啡不是一个严格意义上的算法，因为其中若干步骤不具备确定性，例如“加一些牛奶并轻轻搅拌”，“一些”是多少，何谓“轻轻”等等都是不确定的说明。

3. 能行性

算法的能行性是指算法中有待实现的每一个步骤都必须是基本的。

欧几里德算法涉及到的运算包括整数的表示、整数的除法、整数是否为零的判断以及赋值等，这些运算都是基本的，也即是能行的。

非能行的一个例子是：“如果 2 是使方程 $x^n+y^n=z^n$ 有正整数解 x、y、z 的 n 中最大的整数，则执行算法步骤 x”，这一步骤就不是基本的能行的。因为 17 世纪数学家费马曾提出命题：当 n 是大于 2 的自然数时，没有自然数组 x、y、z 能满足 $x^n+y^n=z^n$。但这个命题尚未得到证明。

4. 一个算法有 0 个或若干个输入，算法的输入作为算法执行的初始数据

欧几里德算法需要两个正整数 m 和 n 作为初始数据。

5. 一个算法有一个或多个输出，作为算法执行的结果

欧几里德算法的结果是正整数 m 和 n 的最大公约数。

1.1.2 算法的表示和分类

算法的作用在于记录及交流人类解决问题的思想，同时算法也是作为编制计算机程序的前导步骤。前面在讨论欧几里德算法时，已经有了变量的概念和赋值的概念，并介绍了算法的一种类似于自然语言的表示方法。为使算法的描述更简捷和清晰，人们研究和创造了多种表示算法的方法，本书将以伪码表示法和流程图表示法作为描述算法的工具。伪码是一种用以表示算法的代码，伪码采用了类似于程序设计语言的语句表示方法，但伪码不是任何一种程序设计语言，不涉及程序设计的细节。流程图是一种图语言，用流程图可以直观地了解算法的结构。

对程序设计方法的理论研究及程序设计实践指出，程序的基本流程可以用三种基本结构来表示，第一种是顺序结构，第二种是选择结构，第三种是循环结构。对于一个算法也可以认为，任何一个算法，无论其多么简单或多么复杂，都可由这三种基本结构组合构造而成。

下面分别讨论顺序结构、选择结构和循环结构的含义，以及其伪码表示法和流程图表示法。讨论将结合算法的举例进行。

1. 顺序结构

在伪码表示法中，每一个步骤称为一条语句。顺序结构中，各语句是按照在算法中排列的先后次序执行的。

【例 1.1】求两个数的绝对值之和。

```
算法 1：
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*   输入：任意两个数                                                          *
*   输出：输入的两个数的绝对值之和                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  num1 ← 第一个输入的数
  num2 ← 第二个输入的数
  num1 ← num1 的绝对值
  num2 ← num2 的绝对值
  sum ← num1 + num2
  输出 sum 的值
```

流程图表示中所用的符号符合国家标准局制定的国家标准(见图 1.1)。

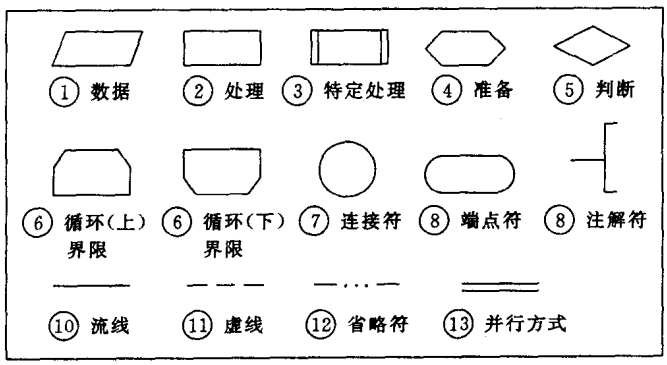


图 1.1　标准流程图的国标符号

算法 1 用流程图表示为图 1.2。

2. 选择结构

选择结构根据某种条件选择性地执行算法的某一部分。在选择结构中判断所说明的条件是否成立，如果条件成立执行某个语句或语句序列，否则执行另一个语句或语句序列。

选择结构用伪码表示为

```
IF <条件>
   <语句序列 1>
[ELSE
   <语句序列 2>]
```

无论是执行了语句序列 1，还是执行了语句序列 2，都代表选择结构执行完成。上列表示还说明，如果我们在条件不成立时不要求做任何动作，可以省略 ELSE 那部分。

选择结构的流程图表示见图 1.3。

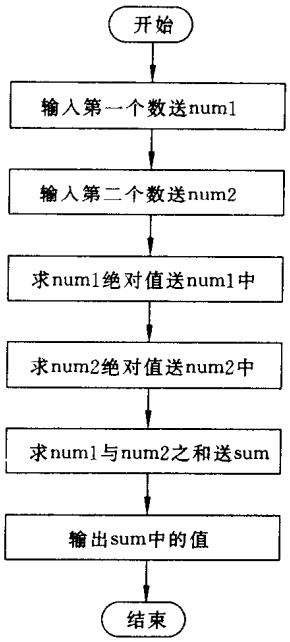


图 1.2　算法 1 的流程图

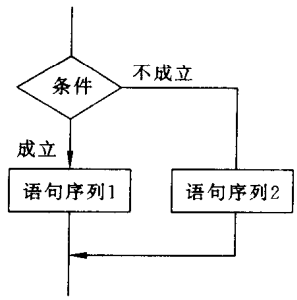


图 1.3　选择结构的流程图

【例 1.2】输入三个数，求其中最大的数。

算法 2：

```
* * * * * * * * * * * * * * * * * * * * * *
*   输入:任意三个数                         *
*   输出:输入的三个数中的最大的数           *
* * * * * * * * * * * * * * * * * * * * * *
  num1 ← 第一个输入的数
  num2 ← 第二个输入的数
  num3 ← 第三个输入的数
  IF (num1 < num2)
    IF (num2 < num3)
      max ← num3
    ELSE
      max ← num2
  ELSE
    IF (num1 < num3)
      max ← num3
    ELSE
      max ← num1
输出 max 的值
```

算法 2 用流程图表示为图 1.4。

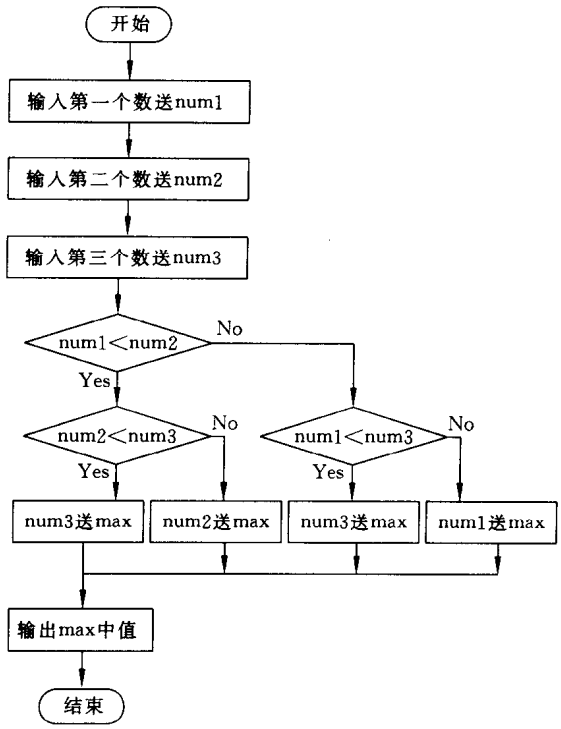


图 1.4　算法 2 流程图

3. 循环结构

循环结构根据某种条件重复性地执行算法所规定的某一部分，重复执行的语句或语句序列称为循环体。在进入循环前要判断所说明的条件是否成立，只有在条件成立时才执行循环体，每次循环体执行后再判断条件是否成立，条件成立则再一次执行循环体，如此循环重复直至条件不成立为止，从而达到重复性地执行算法所规定的某一部分的目的。

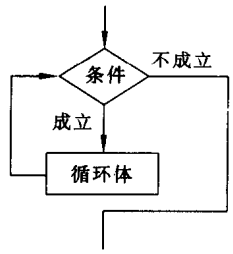


图 1.5　循环结构的流程图

循环结构用伪码表示为

WHILE ＜条件＞

　　＜循环体＞

循环结构的流程图表示见图 1.5。

【例 1.3】设计一个算法，统计输入的一组数据中非负整数的个数。

算法 3：

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*   输入：有限个数的一组输入数据                                        *
*   输出：这组输入数据中的非负整数的个数                                *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
1     number ← 输入数据的个数
2     count ← 0       // 非负整数的个数，先预置为 0
3     WHILE (number ! = 0)
4         num ← 输入的下一个数据
```

```
5        IF (num >= 0)
6            count ← count + 1
7        number ← number - 1
8    输出 count 的值
```

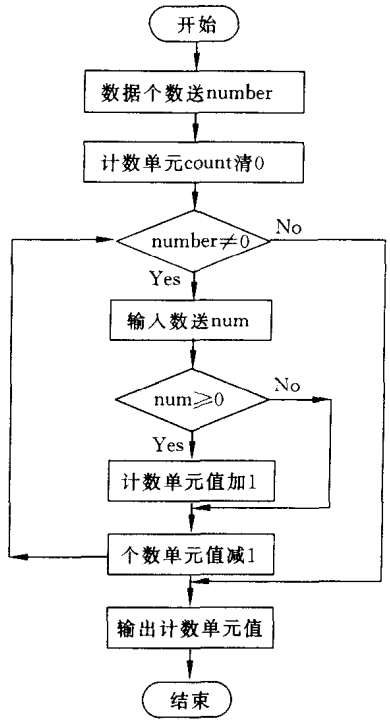


图 1.6　算法 3 的流程图

算法 3 和算法 2 不同，算法 2 中预先确定了输入数据的个数，而算法 3 并没有限定输入数据的个数，也就是说该算法应能灵活地处理不同个数的输入数据。所以算法 3 定义了一个变量 number，该变量作为算法所要求的第一个输入量，代表算法当前这次执行所处理的输入数据的个数，从第二个输入量开始才是算法所处理的输入数据。在算法执行中，每处理了一个输入数据，将 number 的值减 1，一直到所有的输入数据处理完成为止。

在算法 3 中用“//”的记号引入了对算法某个步骤的注释。为提高算法(以及程序)的可读性，在适当的地方加以注释是十分必要的，应有意识地逐步培养这种良好的习惯。

算法 3 用流程图表示为图 1.6。

和前面几个算法相比，这个算法显然要复杂一些。为了学习这个算法让我们用人工模拟的方法，将该算法的实例代入到算法中去，按算法的步骤自己来人工执行一次。为跟踪算法的动态执行过程，需要将算法中的可执行语句及包括判断的语句按顺序编号，同时为记录算法当前执行的及下一步将执行的语句，以及记录算法中变量值的变化过程，需要设计一个算法的动态跟踪表来记录算法的跟踪过程。跟踪表的第一列是执行步的序号，第二列和第三列分别是这一步所执行的语句编号及下一步将执行的语句编号，第四列是当前这一步的注记，以后的各列是算法中所用的各个变量。算法每执行一步，跟踪表就增加一行，记录在这一步执行的语句标号，指示下一步将执行的语句，同时记录这一步对算法中变量值的作用结果，随着算法的逐步执行，跟踪表逐行增加，直至过程结束。算法 3 的跟踪表见表 1.1，在人工模拟前这张表是一张空表，现在表中已经是人工模拟后跟踪的结果。

表 1.1　算法 3 的跟踪表

步	当前语句	下一语句	注　　记	number	count	num
1	1	2	处理 5 个数	5		
2	2	3	计数变量赋初值		0	
3	3	4				
4	4	5	输入第一个数			6
5	5	6				
6	6	7			1	
7	7	3	循环变量减 1	4		
8	3	4				
9	4	5	输入第二个数			−17

续表

步	当前语句	下一语句	注　　记	number	count	num
10	5	6				
11	6	7			2	
12	7	3	循环变量减 1	3		
13	3	4				
14	4	5	输入第三个数			0
15	5	6				
16	6	7			3	
17	7	3	循环变量减 1	2		
18	3	4				
19	4	5	输入第四个数			3
20	5	6				
21	6	7				
22	7	3	循环变量减 1	1		
23	3	4				
24	4	5	输入第五个数			−10
25	5	6				
26	6	7				
27	7	3	循环变量减 1	0		
28	3	8	循环结束			
29	8		输出非负数的个数			

1.1.3　算法的基本分类

算法是解决问题的方法，不同的领域有各自的算法。如果根据问题的领域来分，算法可分为数值问题算法和非数值问题算法两大类。数值问题算法是指解决经典数学问题的算法，例如解方程的算法、解方程组的算法、以及积分算法微分算法等等。随着计算机在数值计算方面应用和研究的发展，求各种数值问题的近似解的算法获得迅速的发展和应用。数值问题算法是数学研究的一个重要方面，本书将在第 8 章给出几个数值问题算法的例子。计算机科学往往对非数值问题的算法给予了更多的重视，计算机非数值应用领域可能是个更宽广的领域，在各个不同的学科中有不同的非数值问题的算法，例如数据处理中的排序算法和查找算法，图形图象处理中变形算法和加工算法等。算法的研究基本上是依赖于具体研究领域的。

尽管不同的领域有各自的算法，但从算法所采用的方法或思路上看，最基本的大致可分成以下几种：直接法、枚举法、递推法、递归法、回溯法、数字模拟等等。下面先简单介绍直接法、枚举法、递推法和递归法的基本思想，其他一些算法将随着语言学习的深入逐步展开。

1. 直接法

直接法是指根据问题所给的条件，直接通过计算来得到解答。本章的算法 1 是直接法的一个例子，通过计算直接得到两个数的绝对值之和。