

16005

北京大学成人教育理科类计算机教材

程序设计方法

陈士龙 顾小凤 李风 编著

N0025251-2
TP311.1

北京大学出版社

北京大学成人教育
理科类计算机教材

程序设计方法

陈士龙 顾小凤 李 风 编著

北京 大学 出版社
北 京

内 容 简 介

本书是北京大学成人教育理科各专业和部分文科专业的计算机教材,由主讲教师按照教学大纲编写。内容包括:程序设计概述,算法的基础知识和实现方法,模块化程序设计方法和应用技巧,以及C语言程序设计方法和编程技巧等。

本书充分考虑到成人教育的特点,简明扼要地讲述基础知识,用通俗易懂的语言描述程序设计方法,并用大量实例分析编程技巧,还在每章后面安排了练习题和上机题,便于读者自学。

本书可作为成人教育、各类大中专、职业高中的教材,也可作为程序设计人员和自学者的参考书。

图书在版编目(CIP)数据

程序设计方法/陈士龙等编著. -北京:北京大学出版社,1996.12

ISBN 7-301-03113-0

I. 程… II. 陈… III. 程序设计-方法-成人教育:高等教育-教材 IV. TP311.1

书 名: 程序设计方法

著作责任者: 陈士龙 顾小凤 李 风

责任编辑: 杨锡林

标准书号: ISBN 7-301-03113-0/TP·0296

出版者: 北京大学出版社

地 址: 北京市海淀区中关村北京大学校内 100871

电 话: 出版部 62752015 发行部 62559712 编辑部 62752032

排 印 者: 北京经纬印刷厂印刷

发 行 者: 北京大学出版社

经 销 者: 新华书店

787×1092毫米 16开本 11.875印张 300千字

1996年12月第一版 1996年12月第一次印刷

定 价: 17.00元

前 言

用计算机代替人脑工作的问题,实质上是把人工求解问题转化为可在计算机上实现的算法语言程序。计算机执行这样的程序,获得问题的解,或指出问题无解。不管是用计算机做科学计算,还是用计算机做企事业管理,或者是用计算机绘图,都是如此。

求解问题,首先要分析清楚问题本身,描述它的数据结构,书写问题的求解步骤,这是算法设计阶段。算法应是确定的(无歧义性),在有限次的有效执行以后即终止。然后,选择某种算法语言(如C语言),表达(描述)上述算法,这是程序设计阶段。做程序设计时,要掌握结构化程序设计的基本方法,熟悉一般的编程技巧,并且运用这些“原则”做实际的编程练习。

因此,程序设计方法是一门应用技术,实践性很强,不能像读一本小说那样学习程序设计方法。学习这门课的要点是:在算法设计中,学会在求解问题时用语言(自然语言、数学语言,或伪码)来描述算法;在程序设计中,学会将算法描述的问题转化为结构化程序,并在计算机上调试和执行它;熟悉一般(或较复杂)应用问题的常规算法,并且多动手进行应用程序设计练习。

本书是按上述思想和原则编著的,并且用C语言程序实施算法。通过这门课的学习,对初学算法程序的设计者来说,可掌握如何编制一个正确的程序的设计方法;对已有一些程序设计基础的读者来说,可学会如何编制一个较好的程序的设计方法;并为从事各种应用程序设计打下基础。

本书是北京大学成人教育理科各专业和部分文科专业的计算机教材。在编写过程中,充分考虑到成人教育的特点,简明扼要地讲述基础知识,用通俗易懂的语言描述程序设计方法,并用大量的例题分析编程技巧,便于读者自学。在每章后,安排了练习题、作业题和上机题,其内容既涉及概念又训练应用,使读者通过实际练习去掌握所学的知识,并将这些知识用到自己的工作中去,编写出符合要求的应用程序。

本书第一章至第九章由陈士龙编写,第十、十一章由李风编写,顾小凤教授审阅全书并提出修改意见,最后由陈士龙统稿。

程序设计方法是一个重要问题,由于水平所限,加之时间仓促,难免有错误和疏漏,恳请读者批评指正。

作 者

1996年9月于北京大学

目 录

第一章 程序设计方法概述	(1)
1.1 一个程序设计的例子.....	(1)
1.2 应用软件设计中的三个主要问题.....	(1)
1.3 算法语言是使用计算机的主要工具.....	(3)
1.4 什么是程序设计方法?好的程序标准是什么?.....	(4)
1.5 构筑结构化程序的基本结构.....	(6)
练习题一.....	(7)
作业题一.....	(7)
上机题一.....	(8)
第二章 算法的基本概念	(9)
2.1 什么是算法.....	(9)
2.2 算法的描述.....	(10)
2.3 算法的实现.....	(11)
2.4 几个实例的算法分析和算法的实现.....	(12)
练习题二.....	(15)
作业题二.....	(16)
上机题二.....	(16)
第三章 顺序结构的实现	(17)
练习题三.....	(21)
作业题三.....	(21)
上机题三.....	(22)
第四章 选择结构的实现	(23)
4.1 条件的描述和表示法.....	(25)
4.2 用 IF 语句实现在两种情况中选择一个.....	(25)
4.3 用 IF 语句嵌套实现多层条件描述和选择.....	(26)
4.4 用 SWITCH 语句实现在多种情况中选择一个.....	(27)
4.5 选择结构程序设计综合练习.....	(29)
练习题四.....	(33)
作业题四.....	(33)
上机题四.....	(34)
第五章 循环结构的实现	(36)
5.1 循环结构的基本形式.....	(36)
5.2 寻找问题内在规律和构造循环.....	(40)
5.3 循环的有终性和异常出口处理.....	(41)
5.4 用多重循环结构实现排列组合.....	(42)
5.5 顺序结构、选择结构和循环结构的联合使用.....	(45)
练习题五.....	(52)

作业题五	(52)
上机题五	(53)
第六章 模块化的实现	(54)
6.1 模块化程序设计的特点	(54)
6.2 模块程序之间的数据传递	(59)
6.3 模块化程序设计技巧	(62)
6.4 模块程序设计实例	(63)
练习题六	(70)
作业题六	(71)
上机题六	(72)
第七章 数据的查询、删除、插入和替换算法	(73)
7.1 数据查询算法	(73)
7.2 数据的删除、插入和替换算法	(75)
7.3 文本的行删除和行插入	(77)
练习题七	(79)
作业题七	(79)
上机题七	(80)
第八章 数据的排序算法	(81)
8.1 选择法	(81)
8.2 冒泡法	(82)
8.3 shell 排序法	(83)
8.4 插入法排序	(85)
练习题八	(86)
作业题八	(87)
上机题八	(88)
第九章 迭代和递归算法	(89)
9.1 迭代算法	(89)
9.2 递归算法	(90)
9.3 迭代过程和递归过程之间的转换	(92)
练习题九	(93)
作业题九	(93)
上机题九	(94)
第十章 链表运算算法	(95)
10.1 链表建立算法	(95)
10.2 链表遍历算法	(97)
10.3 链表中结点删除算法	(98)
10.4 链表中结点插入算法	(99)
10.5 环形链表	(101)
练习题十	(104)
作业题十	(105)
上机题十	(106)
第十一章 C 语言程序设计基本知识	(107)

11.1	特点	(107)
11.2	数据类型	(110)
11.3	表达式	(116)
11.4	语句	(123)
11.5	预处理	(134)
11.6	数组	(138)
11.7	结构	(143)
11.8	联合	(147)
11.9	指针	(148)
11.10	函数	(153)
11.11	文件	(160)
	练习题十一	(165)
	作业题十一	(169)
	上机题十一	(169)
附录		(170)
附录 A	ASCII 码字符表	(170)
附录 B	Turbo C V2.0 的基本操作	(171)
附录 C	Turbo C V2.0 的基本库函数	(173)
附录 D	使用 Turbo C 设计文本窗口	(175)

第一章 程序设计方法概述

不管用计算机求解大问题还是小问题,都要把所求解的问题转化成计算机能识别的语言程序,即进行程序设计。如何进行程序设计,如何编制一个正确的或较好的程序,这就要讨论程序设计的原则和方法问题。

1.1 一个程序设计的例子

已知 s_1, s_2, \dots, s_n 是一组成绩数列,将它们从低分到高分排序。
分析该问题后知道,用计算机求解它,要涉及到下面三个问题:

- (1) 数据存放,用数组较合适。这是数据结构问题。
- (2) 数据排序,例如用选择法。这是数据排序算法问题。
- (3) 程序实现,主要是循环结构程序。这是程序结构问题。

用 C 语言表达上面三个问题的程序为:

```
void sorting(s,n)
    float s[ ];
    int n;
    { int i,j,k;
      float temp;
      for(i=1; i<=n-1; i++)
      {
          k=i;
          for(j=i+1; j<=n; j++)
              if(s[k]>s[j]) k=j;
          temp=s[i];
          s[i]=s[k];
          s[k]=temp;
      }
    }
```

上面提到的三个问题,在一般程序设计中具有“普遍意义”。任何问题的计算机处理,先要分析被加工的数据,确定它的数据结构;在数据表示形式确定后,再解决计算该数据的算法步骤;最后解决用什么样的程序结构实现算法。

1.2 应用软件设计中的三个主要问题

通过程序设计方法的学习和实际练习,就能动手进行一般应用软件的设计。应用软件设计

中有三个主要问题:功能设计,算法设计,结构设计。这三者之间既有联系又不能相互代替。

1.2.1 功能设计

依据“解决什么样的问题,完成什么样的功能”,提出“面向计算机的、含意明确而无歧义性”的说明书。

功能需求,一般是由用户提出的。这些需求未必合理和利于在计算机上实现。设计者分析用户的需求后,应使之“面向计算机且含意确切”。也就是说便于在计算机上实现。

例如,用计算机做教务管理,要求具备“学生学籍管理,学生成绩管理,课程安排”等功能。我们就要分析它的全局和细节,将各部分功能具体化,使之便于在计算机上实现它。一个“面向计算机的、含意明确而无歧义性”的说明书,将是算法设计和程序设计的依据。

1.2.2 算法设计

算法设计在程序设计中占有特别重要的位置。如果对被求解的问题的算法模糊,则不可能“拼凑”出求解它的正确程序。

算法设计是提出实现软件功能的合适算法。算法应是正确有效的,并且用自然语言或伪码描述算法。

比如1.1节中,对一组学生成绩数据进行排序,在学生成绩管理中就涉及到这个问题,我们用选择排序算法。

算法为:

- (1) 选取具有最小值的数组元素 $s[k]$ 。
- (2) 将 $s[1]$ 和 $s[k]$ 交换。

然后,对剩下的 $n-1$ 个元素, $n-2$ 个元素, ..., 重复此运算,直到剩下最后一项 $s[n]$ 。

算法细化为:

```
for (i=1; i<=n-1; i++)
```

```
    1-1 求  $s[i], s[i+1], \dots, s[n]$  中的最小值  $s[k]$ ;
```

```
    1-2 把  $s[i]$  和  $s[k]$  互换;
```

其中 第1-1步可进一步细化为:

k 是当前最小值元素序号

```
for (j=i+1; j<=n; j++)
```

```
    if ( $s[k] > s[j]$ )  $k=j$ ;
```

算法的再进一步细化,将容易转化为上面的 C 语言程序。

注: 算法的特性、作用、如何书写算法等问题,详见第二章。

1.2.3 结构设计

结构设计就是选择适当的数据结构和程序结构实现算法。这和建筑一栋楼房时要进行周密的结构设计相似。

算法设计和结构设计之间有联系,但不能代替。

例如1.1节中,对一组数据进行排序时,主要使用的是数组和循环结构程序。如果不使用数组,而是采用 n 个简单变量,则很难对它们进行排序。如果不用循环结构程序,则程序将要写

得很长和难读。

注：“数据结构”问题，详见数据结构课。本书在进行常规问题程序设计时，将结合问题本身，讨论用什么样的数据结构较合适。

现在，结构设计已被提高到应有的位置上了。一个程序有较好的结构，则易读、易调试、易维护和方便功能扩充，反之不然。

构筑一个结构化的程序，到底应熟悉哪些基本结构，请看后面1.5节。

1.3 算法语言是使用计算机的主要工具

人们通过语言(例如汉语和英语)进行通信，计算机通过语言(机器语言、高级语言)进行运算，这两种语言之间没有直接关系。那么，让计算机代替人要做的工作，便只能用计算机认识的语言描述要做的事，主要是算法语言，如C语言。

同一件事，用“汉语”和“英语”都能表达出来，这说明它们代表的语义一样。但是，汉语和英语是有区别的，句子结构和语法不一样。我们学习计算机语言时，就要采用这样的思想方法：既把自然语言与计算机语言联系起来，又要注意它们在语法和语义上的不同。

例如，计算 $s = 1+2+3+\dots+100$ 。

用自然语言描述它的算法时，我们这样写：

- (1) 计数器 $n=1$ ，累加和 $s=0$ 。
- (2) 累加 $s=s+n$ ，计数 $n=n+1$ 。
- (3) 若 $n>100$ 转第4步，否则转第2步。
- (4) 输出结果 s 。

这样的求解问题的过程，人容易看懂它，计算机却不认识。用计算机认识的C语言表达上述要做的事，程序如下：

```
main()  
{ int n=1,s=0;  
  do {  
    s=s+n;  
    n=n+1;  
  }  
  while(n <= 100);  
  printf("result=%d\n", s);  
}
```

把C语言写的这个源程序，交给计算机处理，执行后输出结果为：

```
result = 5050
```

用计算机求解任何问题，都要做下面几项基本工作：

1. 数据的定义

对求解的问题的数据做属性说明。例如，年龄用整型，电话号码用长整型，姓名用字符串，……，等等。

2. 数据的输入

提供输入计算数据的手段。

3. 数据的运算

提供对运算数据的操作,数据的运算主要是表达式的计算,保存运算的结果值时用赋值语句。

4. 数据的输出

提供获得计算结果的手段。

数据处理过程中,条件的描述,重复动作的控制,独立功能子程序的设计,……等,详见本书第三章—第六章。

程序设计方法课,将大量地用自然语言(或伪码)书写求解算法,然后用计算机 C 语言表达算法。读者要熟练地掌握 C 语言程序设计的基本知识(详见第十一章)。

1.4 什么是程序设计方法?好的程序标准是什么?

1.4.1 什么是程序设计方法

程序设计方法指的是按一定规则书写程序结构,研究的是程序设计的有关原则和方法。科学家已证明,实践已验证,“结构化程序设计方法”是一种较好的程序设计方法,软件设计中,广泛采用这种程序设计方法——自顶向下、逐步求精、模块化。

我们用一个实例,看看结构化程序设计方法的主要思想方法:验证哥德巴赫猜想“任一个大于等于4的偶数,都可以表示成两个素数之和”。

1. 先作整体分析

- (1) 任给一个大于等于4的偶数 n 。
- (2) 在 $[2, n/2]$ 中间找一个较小的素数 a 。
- (3) 取 $b = n - a$, 验证 b 是否为素数。若 b 为素数, 转第4步; 否则转第2步。
- (4) 若 a 和 b 都是素数, 输出 $n = a + b$ 。

2. 进一步细化

- (1) 定义一个函数 $\text{prime}(x)$, 对给定的 x , 若 x 是素数返回 true 值; 否则返回 false 值。
- (2) 定义一个函数 $\text{goldbach}(n)$, 对给定的偶数 n , 它反复调用 prime , 求素数 a 和 b , 使得 $n = a + b$ 。

3. 模块化

```
int prime(x)
    int x;
    { int i, flag;
      /* 假设 x 是素数 */
      flag=1;
      /* 查看 [2, x-1] 中间是否有整除 x 的因数 i, 若有则否定 x 是素数 */
      for(i=2; i<=x-1; i++)
        if(x % i == 0){
            flag=0;
```

```

        break;
    }
    /* 返回真假值 */
    return(flag);
}
void goldbach(n)
int n;
{ int a,b,prime();
  /* 在[2,n/2]中间找一个较小的素数 a,且检查 b=n-a */
  for(a=2; a<=n/2; a++)
  if(prime(a))
  { /* a 是素数 */
    b=n-a;
    if(prime(b)){
      /* b 是素数,输出 n=a+b */
      printf("%d = %d + %d\n", n,a,b);
    }
  }
}
main()
{ int n;
  void goldbach();
  /* 输入一个大于等于4的偶数 */
  scanf("%d", &n);
  /* 调用函数 goldbach */
  goldbach(n);
}

```

上面程序执行时,若给 n 赋值为8,则输出结果为:

8 = 3 + 5

这样设计程序,容易被别人看懂,也容易对它修改和维护,灵活性较好。例如,对100之内的偶数,验证哥德巴赫猜想时,只需要主函数作如下变动:

```

for(n=4; n<=100; n=n+2)
  goldbach(n);

```

1.4.2 好的程序标准是什么

- (1) 按使用说明书的要求正确运行。
- (2) 调试代价小,即容易调试。
- (3) 程序维护代价小。即容易被别人看懂,也容易对它修改和维护。
- (4) 灵活性较好。实践证明,模块化、结构化的程序灵活性较好。

(5) 程序开发代价小。

(6) 程序效率高,尤其是程序中频繁使用的部分。例如,验证哥德巴赫猜想程序中,求 $n=a+b$ 时,反复调用函数 $\text{prime}(x)$ 就是这样。

1.5 构筑结构化程序的基本结构

计算机早期主要用于自然科学领域中的数值计算,后来在管理科学领域中的非数值计算的比重越来越大,直到今天的多媒体,使计算机应用范围无所不有。长期的实践经验,获得了一个重要的结论:应采用结构化程序设计。这种程序设计“有章可循”,它们像建筑楼房一样,可由一些“基本结构”,按照一定的逻辑规则有序地组织在一起,去解决任何较复杂的计算问题。这些基本结构,都有自己的独特功能,计算机高级语言配备相应的语句,支持表达求解问题的不同侧面,彼此间相辅相成。这些基本结构是:

1. 顺序结构

顺序结构的特点是:计算机执行这种程序,按书写的先后次序,自上而下地逐条执行,中间没有条件判定分叉和重复过程。见图1.1。

注:顺序结构程序设计,详见第三章。

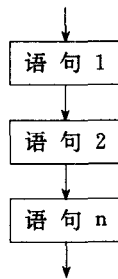


图 1.1 顺序结构程序执行

2. 选择结构

选择结构的特点是:程序执行的控制出现了“分叉”,要根据不同情况选择其中之一执行,即“具体问题具体分析”。见图1.2。

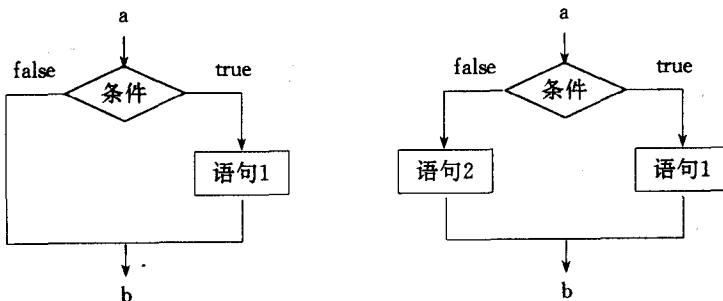


图 1.2 选择结构程序执行

注:选择结构程序的设计,详见第四章。

3. 循环结构

较复杂问题的求解,往往是要做大量的重复计算,即设计循环结构程序。循环结构由循环条件和循环体组成。循环执行过程中,循环体中的语句没变,参加运算的数据变了,循环体在有限次运行后终止。见图1.3。

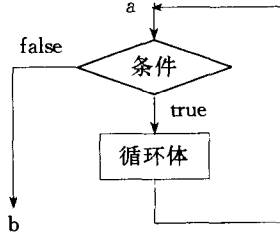


图1.3 循环结构程序执行

注:循环结构程序的设计,详见第五章。

一个大问题的求解,则化成若干个小问题,这就要进行模块化程序设计。模块化程序设计详见第六章。

这一章提出了程序设计方法中的一些“原则”问题,举的例子较少。这好比数学中的“公理”和“定理”。在后面的章节中,将反复地应用这些“原则”,去解决程序设计中的实际问题,就像使用“公理”和“定理”解数学题目那样。

练习题一

1. 编制计算机解题程序时,为什么先要掌握程序设计方法?
2. 应用软件设计中三个主要问题是什么?它们之间的关系是什么?
3. 什么是程序设计方法?好的程序标准是什么?
4. 结构化程序设计方法的特点是什么?“自顶向下,逐步求精,模块化”的意义?
5. 结构化程序构筑的基本结构是什么?
6. 用“自顶向下,逐步求精”的结构化程序设计方法,分析下面问题的求解过程:

(A) 求一元二次方程 $Ax^2 + Bx + C = 0$ 的根。

(B) 求两个正整数的最大公约数。

提示:参考验证哥德巴赫猜想问题的分析方法。

作业题一

一、判断题

1. “自顶向下,逐步求精”的意义,就是从头到尾看一遍,将大问题变成小问题来做。
2. 设计计算机解题程序,就是先找一下有关语句,然后将语句组织到一起。
3. 应用软件设计中三个主要问题是“提出问题、分析问题、解决问题。”
4. 一个好的程序就是给它计算数据时能做对。
5. 结构化程序构筑的基本结构就是“赋值语句、条件语句、循环语句。”

二、填空题

1. 应用软件设计中的三个主要问题是() () ()。
2. 所谓程序设计方法是指()。好的或较好的程序的标准是() () () () () () ()。
3. 结构化程序设计方法的基本要点是()。
4. 构筑结构化程序的基本结构是() () ()。

上机题一

1. 求一元二次方程 $Ax^2 + Bx + C = 0$ 的根。
2. 求两个正整数的最大公约数。

注：本章验证哥德巴赫猜想程序中，用“汉字”写程序注释，方便读者自学。这样的程序是可以在计算机中处理的，只要在启动 Turbo-C 编辑系统之前，先启动某个汉字系统，如 UC DOS V5.0。全书的其它章节也有这样的问题，不再重述。

第二章 算法的基本概念

第一章介绍了程序设计方法的基本知识,初步了解了算法设计在程序设计中的地位。其中谈到,把实际计算问题交给计算机求解时,首先要描述问题的求解算法——算法设计,然后再用计算机语言表达算法——程序设计。本章讨论算法的有关基本知识:什么是算法、算法的作用、算法的特性、算法的描述(书写)。以后章节的学习中将反复应用算法。

简单地说,讨论求解问题的算法,犹如日常去完成一个任务,当盲目无举措时事半功倍,当有办法和措施周密时事倍功半。

2.1 什么是算法

2.1.1 什么是算法和算法的作用

我们给出一个比较粗略的、直观的定义:对某个特定的问题,提出求解它的公式和步骤。算法实现以后,得到问题的解,或指出问题无解。

[例2.1] 求正整数 n, m 的最大公约数。

人工求解时,先将它们分解为质因数的乘积,然后找出“公共因数中的最大者”。比如, $n=6, m=8$, 有 $n=2 \times 3$ 和 $m=2 \times 2 \times 2$, 所以最大公约数是2。

这样的思维过程,目前计算机还不能做。因此,要提出面向计算机的求解算法,我们用欧几里得辗转相减算法:

- (1) 输入 n, m 的值。
- (2) 若 $n > m$, 则 $n = n - m$ 。
- (3) 若 $m > n$, 则 $m = m - n$ 。
- (4) 若 $n = m$, 则算法结束, 转第5步; 否则, 重复上述过程, 转回第2步。
- (5) 输出 n (或 m) 中的值, 它是所求的最大公约数。

用 C 语言表达上述算法的程序片段是:

```
scanf("%d%d", &n, &m);
while(n != m)
{
    if(n > m) n = n - m;
    if(m > n) m = m - n;
}
printf("result = %d\n", n);
```

2.1.2 两大类算法

1. 数值算法

例如, 求定积分 $\int_a^b f(x) dx$, 求方程 $Ax^2 + Bx + C = 0$ 的根, 矩阵运算等。这些问题, 都有经

典的算法,许多计算机上装有算法程序包。这与科学计算应用计算机时间长有密切关系。

2. 非数值算法

例如,对一组数据进行排序,查找其中的某个数,在其中插入一个新数等。非数值计算问题,在管理科学中应用十分广泛,其中许多问题没有现成的求解公式,是人们关心的热点,也是计算机应用领域广的一种表现。

2.1.3 算法特性

1. 有穷性

包含有限的操作步骤,执行有限次操作之后终止。在例2.1中,操作步骤是五步,终止条件是 $n=m$ 。“无限次地执行操作”,不是一个正确的算法。

2. 确定性

每一步是确定的,不能“模棱两可”。例如,若将例2.1中的第2步和第3步改成为:

若 $n>m$ 则 $n=n-m$; 否则 $m=m-n$ 。

那就不一定对。算法的某一步运算结果不确定,不可能产生正确的计算结果。

3. 有效性

每一步应当有效地执行,并产生有效和确定的结果。例如,16位(bit)微机中,两个实型量(如, $x=0.0000000123$, $y=0.0000000321$)作相等比较时,计算机则视为 true,事实上 $x \neq y$ 。用 $|x-y| < \epsilon$ 较合适。

4. 零个或多个输入

例2.1中对 n 和 m 输入数据。零个输入的情况也有,如求100之内的素数时,可以这样写:

```
for(n=2; n<=100; n++)
```

```
if(prime(n))输出素数 n;
```

5. 一个或多个输出

输出是打印算法的解,否则“一无所获”。

上面所讲的算法的特性,约束我们去正确地书写算法,使之正确无误地执行,达到求解问题的预期效果。

初学“算法设计”的读者,在书写实际问题求解算法时,往往“忘记”了算法特性,使得所写的算法“漏洞”较多,当然不会有好的算法程序实现。这一点要特别注意。

2.2 算法的描述

算法的描述,可以用自然语言(如汉语和数学语言)表达,也可以用伪码表示,或者自然语言和伪码联合使用。

2.2.1 用自然语言表达

用我们熟悉的自然语言(汉语或英语)书写算法,直观且容易掌握。如例2.1,求正整数 n , m 的最大公约数的欧几里得辗转相减算法,就是使用的自然语言书写算法。

自然语言表达算法,与计算机的具体的高级语言形式差距较大,需要读者认真地进行转换。反复练习,则“熟能生巧”。