C/C++程序设计基础

李凤云　刘凤华　编著

人 民 邮 电 出 版 社

图书在版编目(CIP)数据

C/C++程序设计基础/李凤云，刘凤华编著．—北京：人民邮电出版社，2003.1
（高职高专现代信息技术系列教材）
ISBN 7-115-10908-7

I．C…　II．①李…②刘…　III．C 语言—程序设计—高等学校：技术学校—教材
IV．TP312

中国版本图书馆 CIP 数据核字(2002)第 100829 号

内 容 提 要

本书是编者在广泛参考有关资料的基础上，结合自己多年的教学经验和应用 C 语言的体会，为满足 C 语言初学者的需要而编写的。全书共分 16 章，内容包括程序设计的基础知识、C 语言的基本知识、C++初步、Turbo C 环境下的程序调试方法等。其中，C 语言的基本知识部分在编写时兼顾了全国计算机等级考试的要求。

本书思路新颖，以问题引出概念，在例题中讲解语法；注重实用，在讲解语法的同时强调其实际用途，在例题分析中运用程序设计方法。各章配有详细的上机实践练习和程序调试分析，便于读者深入理解语法和培养程序设计能力，也便于读者自学。书中例题丰富，且均在 Turbo C 下调试通过。

本书为普通高校高职高专的程序设计入门课教材，也可作为全国计算机等级考试的辅导教材，也适合广大程序设计初学者作为学习 C 语言的自学读本。

C/C++程序设计基础

◆ 编　　著　李凤云　刘凤华
责任编辑　潘春燕
执行编辑　王健波

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www. ptpress.com.cn
读者热线：010-67180876
北京汉魂图文设计有限公司制作
印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：19.5
字数：470 千字　　2002 年 3 月第 1 版
印数：1 – 0 000 册　　2002 年 3 月北京第 1 次印刷

ISBN 7-115-10908-7/TP・3227

定价 25.00 元

关于本书

近年来，我国高等院校不但在计算机专业开设 C 语言课程，而且在越来越多的非计算机专业中也开设了 C 语言课程。在全国计算机软件专业技术资格和水平考试的程序员级考试中，程序设计部分也由多种程序设计语言选择逐步改为单一的 C 语言试卷。同时，随着计算机技术的发展，大量面向对象的程序设计语言逐渐成为用户的首选工具，使得 C 语言课由原来作为程序设计能力的提高、强化课程，逐步变为程序设计的入门语言课。最近两年，高校迅速扩招，“ 大合堂 ” 上课的现象越来越普遍，由此导致 C 语言上机实践课中每个学生很难得到及时的辅导，直接影响了教学效果。面对这种情况，原来大量的 C 语言教材表现出一些不足。首先，大部分是针对学过一门以上的计算机语言，或者具有一定计算机基础知识的读者，因此教材的重点放在了 C 语言的语法本身，淡化了程序设计方法的介绍，不利于培养学生的程序设计能力。其次，大部分教材的讲解模式为概念、语法、例题，在实际讲授中发现，这种思路不利于学生接受新知识。再次，原有教材普遍缺少必要的、紧扣所学内容的上机练习题，缺少上机调试的方法指导，不利于学生在实践课中快速地调试程序，掌握所学内容，也不利于课下自学。本书正是为解决这些问题而编写的。

本书具有如下特点。

1．思路新颖、注重实用。本书以实际问题引出概念，在例题中讲解语法及注意问题，便于初学者接受。同时强调实用性，在讲解语法中给出应用建议，使读者在掌握语法的同时明确它的实际用途。

2．注重程序设计能力的培养。在介绍 C 语言基本知识的同时，结合实例着重介绍程序设计方法，使读者逐步建立起程序结构的概念，掌握程序设计的一般思路和方法，培养学生独立解决问题的能力。

3．加强实践环节。除第 1 章和第 16 章外，其它各章后面都附有上机练习，包括紧扣本章内容的练习题和本章常见的错误分析两部分；第 16 章介绍 C 语言集成环境的使用方法，以及常见错误介绍。这两者相结合，便于读者上机实践，也便于课下自学，更有利于读者深入理解本章的内容。

4．扩充了 C++的初步知识，便于读者以此为基础进一步学习面向对象的程序设计语言，如 C++、VC++等。

5．语法介绍简明扼要，条理清楚，例题丰富，难度控制在初学者能接受的范围内，对于哪些内容可以了解、哪些必须掌握、哪些是较深入的应用等都给出了明确的说明。程序例题尽量简单，尽可能少的牵扯硬件，以适合非计算机专业的读者和初学者使用。

本书由李凤云、刘凤华、周应兵、王凤瑛编写，其中第 1、2、3、4、5、6、10 章由李凤云编写，第 7、8、11、13、14、15 章由刘凤华编写，第 9、12 章由周应兵编写，第 16 章由王凤瑛编写。全书由顾骏梁教授和周应兵副教授主审，李凤云负责统稿和定稿。他们都是各校计算机专业的骨干老师，有多年教授 C 语言的经验。在本书的编写过程中，得到了沈祥玖教授的热心支持，在此表示感谢。

由于时间仓促，编者的水平有限，书中难免有不当之处，敬请读者不吝指正。

编者电子邮件地址：fy_li@sina.com。

编　者

02002 .10

目　　录

第 1 章　程序设计基础

我们知道，计算机能自动进行信息处理，实际上是计算机执行特定程序的结果。例如在图书管理系统中，我们可以借助计算机查找图书、借书和还书，这是通过执行该系统中的查找图书程序、借书程序和还书程序来实现的。我们学习程序设计的目的，就是要学会编写程序，解决实际问题。对于一个具体的任务，怎样编写它的程序？怎样才能编写出高质量的程序呢？这是每一位程序设计的初学者急于了解的问题。

本章围绕以上问题，介绍程序设计的有关概念、程序设计的基本过程、程序设计的关键——算法设计、结构化程序设计方法。

学习程序设计，是一项艰苦的工作，需要循序渐进的方法、持之以恒的决心。本章的内容贯穿学习程序设计的整个过程，因此，本章旨在使读者了解程序设计的总体思路，重要的是在以后的学习中，不断地将本章的思想付诸于实践，真正掌握程序设计的方法。

1.1　程序设计概述

1.1.1　程序与程序设计语言

程序是为解决某一问题而编写的语句序列。通俗地说，将解决一个实际问题的具体操作步骤用某种计算机语言描述出来，就形成了程序。

计算机语言是计算机能够识别的语言，它与我们熟悉的汉语、英语一样，都有一套固定的符号和语法规则。利用计算机解决问题，必须用计算机语言告诉计算机“做什么”及“怎样做”，这个过程就是程序设计，因此计算机语言又叫程序设计语言。

程序设计语言是人们根据描述问题的需要而设计的。如今，人们已经设计出了许多种程序设计语言，如汇编语言、BASIC、FoxPro、C、C++、Visual C++、Visual Basic、SQL、Java 等上百种，但其中只有极少部分得到了比较广泛的应用。对程序设计语言的分类可以从不同的角度进行，如面向机器的程序设计语言、面向过程的程序设计语言、面向对象的程序设计语言等。其中，最常见的分类方法是根据程序设计语言与计算机硬件的联系程度将其分为三类，即机器语言、汇编语言和高级语言。前两类依赖于计算机硬件，有时统称为低级语言；而高级语言与计算机硬件基本无关。因此可以说程序设计语言经历了由低级向高级发展的过程。

机器语言：是一种用计算机能直接理解和执行的“0”和“1”表示各种操作的程序设计语言。机器语言的优点是计算机能够直接识别，执行速度快；其缺点是难记忆、难书写、编程困难、可读性差且容易出错。机器语言是面向机器的语言，因机器而异，可移植性极差。

汇编语言：是一种用助记符号来表示各个基本操作的程序设计语言。如用 ADD 表示加法操作，用 SUB

表示减法操作。汇编语言也是一种面向机器的语言，但计算机不能直接执行汇编语言程序，必须经过汇编程序翻译成机器语言程序后才能在计算机上执行。目前，许多系统软件的核心部分仍采用汇编语言编制。

高级语言：是一种用接近自然语言和数学语言的语法、符号描述基本操作的程序设计语言。它符合人们叙述问题的习惯，因此简单易学。如上面所列的程序设计语言中，除汇编语言外，均是高级语言。通常把用高级语言编写的程序称为“源程序”，而把由二进制的“0”、“1”代码构成的程序称为“目标程序”。用高级语言编写的程序计算机不能直接执行，需用专门的翻译程序将其转换成机器语言目标程序后才能执行。如用 C 语言编写的程序，必须先经 C 编译系统翻译成目标程序，再连接成可执行文件后才能执行。

1.1.2 程序设计的一般过程

概括地说，程序设计就是分析问题、编写程序、调试程序的过程。要设计出一个好的程序，首先必须了解利用计算机解决实际问题的过程，其次必须掌握程序设计的基本技术，最后要熟练掌握一门程序设计语言。

简略地描述用计算机解决问题的基本过程如图 1.1 所示。

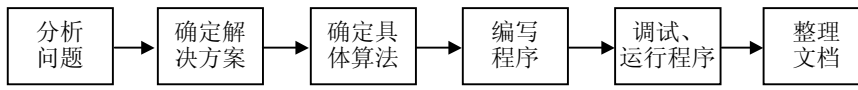


图 1.1

具体地说，程序设计的基本过程一般由分析所求解的问题，抽象数学模型，设计合适的算法，编写程序，调试通过直至得到正确结果等几个阶段。其设计步骤如下。

（1）分析要解决的问题，明确任务。

接到任务后，首先对要处理的对象进行调查分析，明确要实现的功能。然后详细地分析要处理的原始数据有哪些、从哪里来，要进行怎样的加工处理，处理的结果送到哪里？例如，要编一段计算学生的平均成绩的程序。分析该问题，明确它有三项功能：输入学生成绩、求平均成绩、输出结果；要处理的原始数据为：学生的成绩；要进行的处理为：求平均结果为平均值，送屏幕显示。

（2）分析问题，建立数学模型，并选择合适的解决方案。

对要解决的问题进行分析，找出它们的运算操作和变化规律，然后进行归纳，并用抽象的数学语言描述出来。也就是说，将具体问题抽象为数学问题。例如，在计算平均成绩的任务中，分析求平均成绩的处理过程，可以将数据的具体含义去掉，抽象为：计算一批数据的平均值。当同一问题有多个解决方案时，可根据计算的速度、求值的精度等方面，选择适合计算机解决问题的方法。

（3）确定数据结构和算法。

方案确定后，要考虑程序中要处理的数据的组织形式即数据结构；并针对选定的数据结构设计相应的算法。然后根据已确定的算法，画出流程图（或其他描述方法）。这样能使程序结构清晰，减少编写程序的错误。也就是说，算法是编程的依据。

（4）编写程序。

把用流程图描述的算法用计算机语言（如 C 语言）描述出来，就是编写程序。这一步应注意的是，要选择一种合适的语言来适应实际算法和所处的计算机环境，并要正确地使用语言，准确地描述算法。

（5）调试程序。

将源程序送入计算机，进行排错、试运行，调试的结果是得到一个能正确运行的程序。通常，调试程序至少占整个程序设计工作量的一半。

（6）整理资料，交付使用。

程序调试通过后，应将解决问题整个过程的有关资料进行整理，编写程序使用说明书。

以上是一个完整的程序设计过程，对于初学者而言，由于要解决的问题都比较简单，因此（1）、（2）、（3）可看作一步，即分析问题、设计算法。

1.1.3　学习程序设计的目的

从程序设计的一般过程可以看出，要成为一名程序设计人员，应具备以下几方面的知识：数据结构、算法、程序设计方法、程序设计语言。因此，学习程序设计的目的不只是学习一种特定的语言，而是结合某种语言学习进行程序设计的一般方法。脱离具体的程序设计语言来学习程序设计是不现实的，但学习程序设计语言只是为了设计程序，因此在学习程序设计语言时千万不能只局限于掌握语言本身的语法。而且，现在流行的高级语言多种多样，各种语言也在不断发展、更新，因而我们通过 C 语言的学习，应掌握程序设计的一般思路，掌握简单的数据结构和常用算法，掌握学习程序设计语言的方法，达到举一反三的目的。

1.2　算　　法

通过上一节的介绍，可以看出，在程序设计中，分析问题是解决问题的前提，算法设计是程序设计的关键，程序只不过是算法的另一种表现形式（即用程序设计语言描述的算法）而已。

因此，本节着重讨论算法及其描述。

1.2.1　算法的概念

算法是对解决某一特定问题的操作步骤的具体描述。广义地说，算法就是为解决一个问题而采取的方法和步骤。例如，期末考试前的复习计划，就是“复习算法”；到医院看病，先挂号，后诊断、检查，再取药等，是“看病算法”。而计算机中的算法就是为计算机解决问题设计的有明确意义的运算步骤的有限集合。世界著名的计算机科学家 Wirth（沃思）提出了一个用来表达程序设计实质的著名公式：

程序=算法+数据结构

就是说：“程序是在数据的特定的组织方式的基础上，对抽象算法的具体描述”。作为程序设计人员，在设计算法前，必须认真考虑和设计数据的组织方式，即数据结构；然后针对具体的数据结构设计相应的操作步骤，即算法。

例 1.1　计算任意长方形的面积。

分析：要实现的功能有：输入长和宽，计算面积，输出结果。先定义数据结构：程序中要处理的数据有 3 个，即长、宽、面积，设分别用实型变量 *length*、*width*、*area* 表示。则算法如下：

① 输入长和宽分别存入 *length* 和 *width*；

② 计算面积：*length*×*width* 送入 *area*；

③ 输出结果 *area*。

例 1.2　计算 *s*1+*s*2+*s*3+*s*4+*s*5（其中 *si* 表示第 *i* 个数据）。

分析：要实现的功能是“加法运算”，要输入的数据是 *s*1~*s*5，对其进行的运算是“加”，结果是“累加和”，并输出结果。

算法分析：

（1）手工计算步骤

① 输入 *s*1、*s*2，求 *s*l 与 *s*2 的和；得到两个数之和。

② 输入 *s*3，将上一步的和与 *s*3 相加；得到 3 个数之和。

③ 输入 *s*4，将上一步的和与 *s*4 相加；得到 4 个数之和。

④ 输入 *s*5，将上一步的和与 *s*5 相加；得到 5 个数之和。

⑤ 输出最后的和。

从手工计算过程可知：其运算规则类似于用算盘计算，每次仅求两数之和，其中一个加数为上一步所得的结果，另一加数为新输入的一项，重复这个过程，直到加完最后一项为止。

问题：假如要处理的数据有 100 个或者更多，这样做岂不是太繁琐或不现实吗？如果我们用计算机来解决此类问题，就容易多了。

（2）计算机中的算法

先定义数据结构：设变量 *s* 表示累计和，初值为 0；变量 *x* 表示每次要处理的数据（*s*1~*s*5 中的一个），变量 *i* 表示要处理第几个数（其取值范围可以是 1~5），设初值为 1。则算法设计如下：

① 赋初值 0 送入 *s*；

② 赋初值 1 送入 *i*；

③ 输入第 *i* 个数 *x*；

④ 累加：*s*+*x* 送入 *s*；

⑤ 计数增值：*i*+1 送入 i；

⑥ 若 *i*≤5，表示数据还未处理完，返回③继续重复③、④、⑤；否则，计算结束。

⑦ 输出结果 *s*。

可见，用计算机中表示的算法可以更简洁地描述繁琐的手工操作。

算法具有如下性质。

（1）有穷性：它是一个有穷动作序列，且每一步操作在有限时间内完成。如此保证程序能正常结束。

（2）确定性：每一步操作都是确定的，不能“模棱两可”。

（3）有效性：每一步操作应当有效地执行，并产生确定的结果。

（4）有 0 个或多个输入。有时要处理的数据是由程序产生的，因此可以没有输入。

（5）有一个或多个输出，即至少有一个输出结果。

上述所讲的算法特性，是为了约束人们正确地设计算法，使之正确无误地执行，达到求解问题的预期目的。同时，算法还应有直观、清晰、易懂的表示形式，以利于维护和修改。

1.2.2 算法的描述

描述算法的方法有多种，常用的有自然语言、结构化流程图、N–S 图、伪代码、PAD 图等。建议初学者掌握流程图和自然语言表示法，其他的先作了解，有一定基础后再慢慢熟悉。

1．用自然语言表示算法

例 1.1 和例 1.2 就是用自然语言描述的算法，这样描述的算法通俗易懂，直观且容易掌握，但容易出现“歧义性”（即对同一句话，不同的人可能有不同的理解），且表达的算法与计算机的具体高级语言形式差距较大，通常在很简单的问题中使用。

2．用流程图表示算法

流程图是用几种图形、箭头线和文字说明来表示算法的框图。用流程图的优点是直观形象、易于理解，

能将设计者的思路清楚地表达出来，便于以后检查修改和编程。

流程图中规定使用的符号如图 1.2 所示。

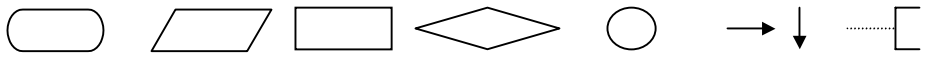


图 1.2

经过长期的实践，人们将传统的流程图逐步改进，提出了结构化的流程图，即一个流程图由 3 种基本结构（顺序、选择、循环结构）组成，这 3 种基本结构可以相互嵌套，组合成复杂的算法。由结构化的流程图写出的程序也是结构化的程序。结构化程序的 3 种基本结构如图 1.3 所示。

图 1.3 中，虚线框表示一个基本结构，A、B 可以是一个基本操作或一个基本结构。P 表示条件，它的值有两个：成立和不成立。

图 1.3 中，列出了以下基本结构：

（a）表示顺序结构，按流程线的指向，先执行 A，后执行 B。

（b）表示选择结构，执行该结构时，先判断条件 P，若条件成立，则执行 A；否则，执行 B。也就是说，根据条件 P 成立与否，每次从 A 或 B 中选择一个执行。

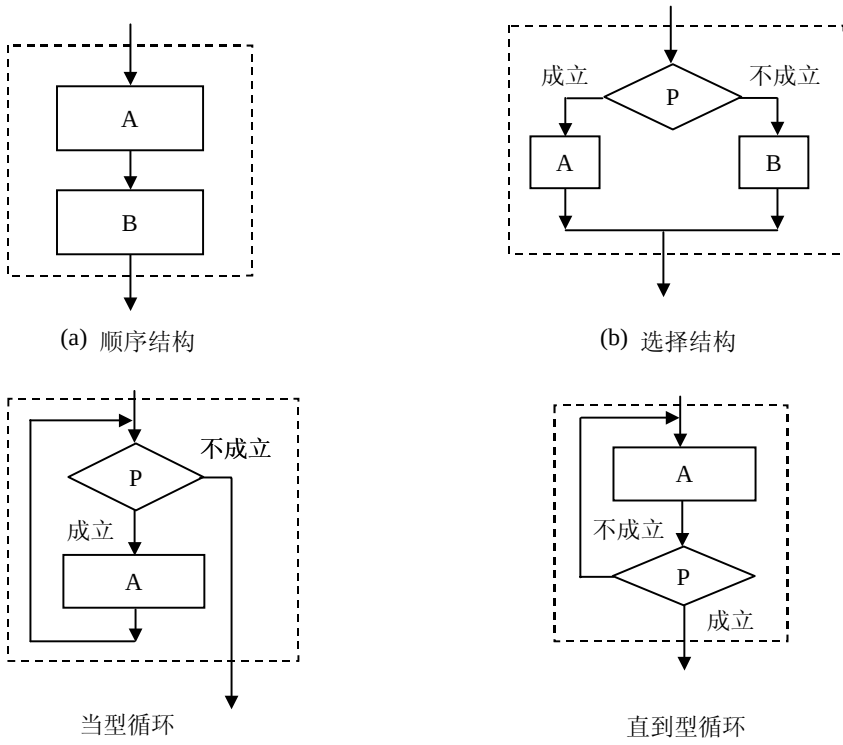
（c）表示当型循环结构，它的功能是：先判断条件 P，若成立，则执行 A，然后判断条件 P 是否成立，若成立，再执行 A，如此反复，直到当条件 P 不成立时结束循环。

（d）表示直到型循环结构，它的功能是先执行 A，然后判断条件 P，若条件不成立，返回重复执行 A，并判断条件 P，直到条件成立时结束循环。

可见，带有条件判断的问题用选择结构实现，需要重复某些操作的问题用循环结构实现。

例 1.3　将例 1.1 的算法用流程图表示如图 1.4 所示。

例 1.4　判断任意整数是奇数还是偶数。



(a) 顺序结构　(b) 选择结构

当型循环　直到型循环

(c) 循环结构

图 1.3

分析：设 *n* 表示要处理的整数，则首先输入 *n*，然后判断条件“*n*%2 等于 0”（%是求余运算符）是否成立，若条件成立，则输出“*n* 是偶数”；否则输出“*n* 是奇数”。用流程图表示的算法如图 1.5 所示。

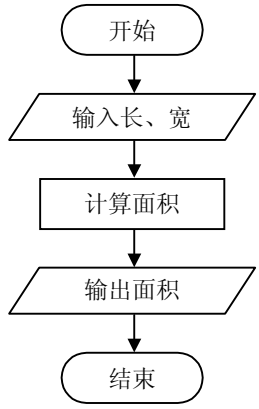


图 1.4

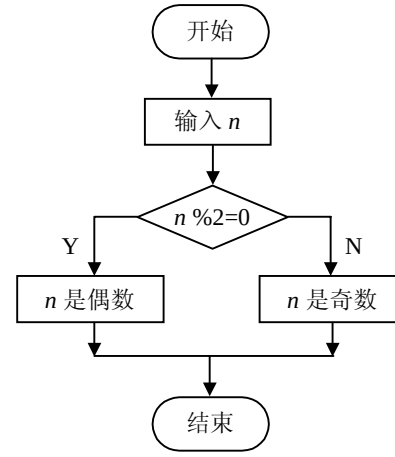


图 1.5

例 1.5　将例 1.2 的算法用流程图表示如图 1.6 所示。

3．用 N–S 图表示算法

N–S 图是以美国学者 I.Nassi 和 B.Shneiderman 的名字命名的。它的基本组成单位是一个矩形框，该框表示一个操作或一个基本结构。图中没有流程线，按矩形框自上到下的顺序执行。用 N–S 图表示的 3 种基本结构如图 1.7 所示。

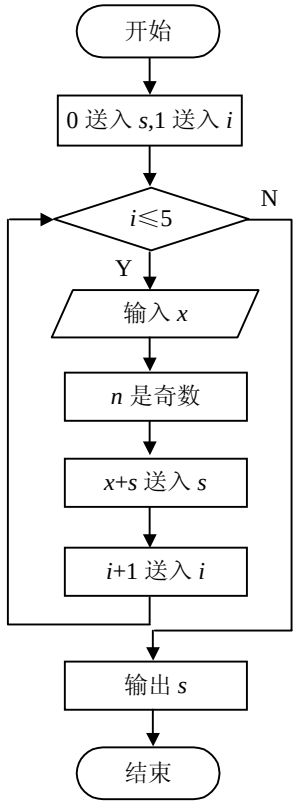


图 1.6

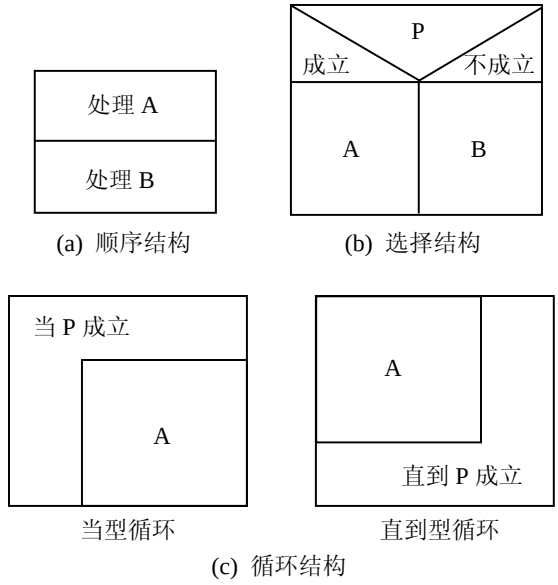


图 1.7

其中，A、B 表示一个操作或一个基本结构；P 表示条件，其值有两个：成立和不成立。

图(a)表示顺序结构，先执行 A，后执行 B。

图(b)表示选择结构，当条件 P 成立，则执行 A，否则执行 B。

图(c)表示“先判断条件，后执行 A”的当型循环。

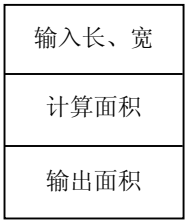
图(d)表示“先执行 A，后进行循环条件判断”的直到型循环。

例 1.6　将例 1.3、例 1.4 和例 1.5 的算法用 N-S 图表示如图 1.8 所示。

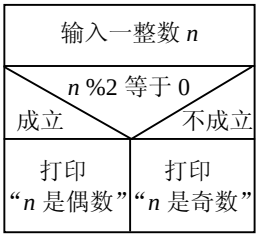
4．用伪代码表示算法

从以上各例可以看出，用流程图和 N-S 图表示算法，虽然直观易懂，但画起来比较麻烦，修改也很困难，而实际设计一个算法时（特别是复杂的算法），经常需要反复修改才能满足要求。因此，流程图和 N-S 图用在设计复杂算法过程中不是很方便，而用伪代码表示则更容易实现。

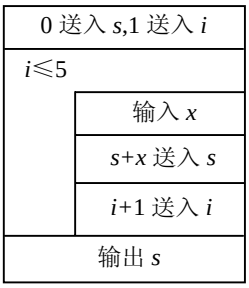
伪代码是指用计算机语言与自然语言相结合来描述算法。通常，算法的 3 种基本结构的描述用类似于高级语言的符号描述，具体操作用英文或汉字描述。例如，判断一个整数是奇数还是偶数的算法用伪代码描述如下：



例 1.3 N-S 图



例 1.4 N-S 图



例 1.5 N-S 图

图 1.8