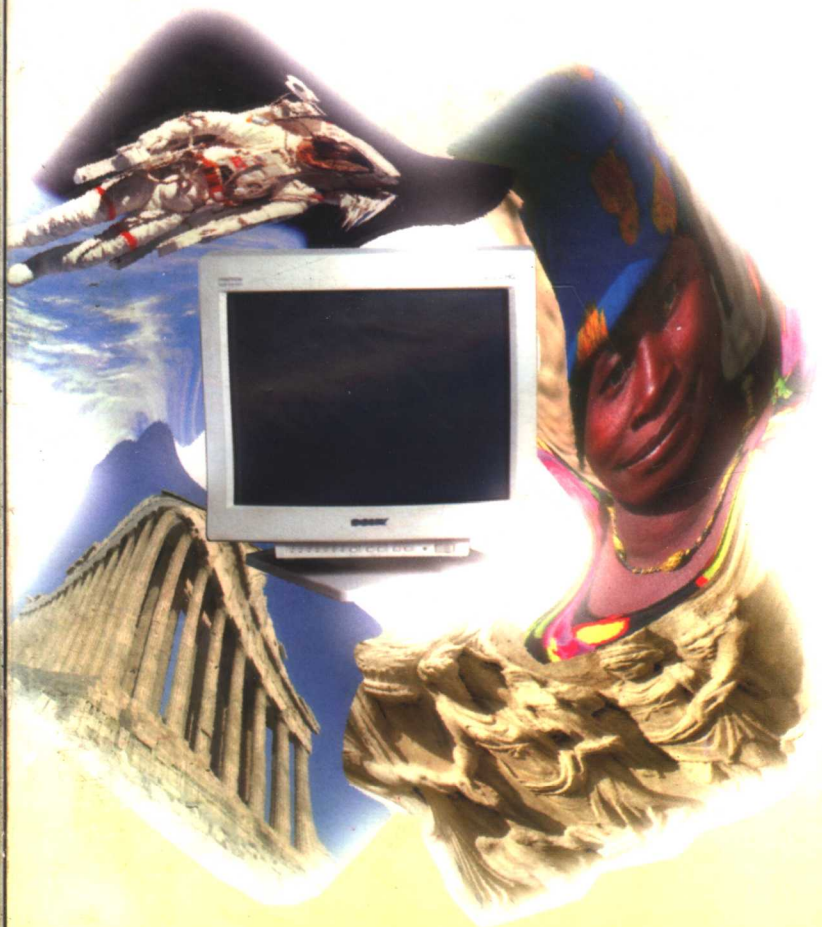
计算机程序设计基础

赵宏 靳小燕 苏玉心 编
宋开磊 审



中国铁道出版社

计算机程序设计基础

赵宏　靳小燕　苏玉心 编

宋开碥 审

（京）新登字 063 号

内 容 简 介

该书是作者在从事多年计算机程序设计教学的基础上编写而成的．全书以 C 语言为例，全面介绍了程序设计的基本概念、过程和方法，如程序设计的基本要素、控制结构、数组；数据文件的建立和使用；程序的调试方法等．该书突出了编程功能的四要素，即：语言、算法、数据结构和结构化程序数据技术．该书是学习计算机程序设计的一本好教材．

图书在版编目（CIP）数据

计算机程序设计基础／赵宏等编．—北京：中国铁道出版社，1998.2
ISBN 7-113-02923-X

I.计… II.赵… III.程序设计-基本知识 IV.TP311

中国版本图书馆 CIP 数据核字(98)第 03431 号

书　　名：计算机程序设计基础
著作责任者：赵宏　靳小燕　苏玉心 编，宋开疆 审
出版·发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：刘 波
责任编辑：刘 波
封面设计：马 利
印　　刷：中国铁道出版社印刷厂
开　　本：787 × 1092　1/16　印张：16.5　字数：398 千
版　　本：1998 年 2 月第 1 版　　1998 年 2 月第 1 次印刷
印　　数：1—4000 册
书　　号：ISBN7-113-02923-X/TP · 290
定　　价：23.50 元

前　言

随着计算机技术的飞速发展和信息社会的需求，计算机已深入到社会生活的各个方面，并越来越成为一个不可缺少的工具。

本书是作者在多年来从事计算机程序设计教学工作的基础上，针对初学者编写的学习计算机程序设计课程的一本教材．以往对初学者选用哪门计算机语言有过争论，我们认为这并不是最重要的，重要的是要掌握程序设计的基本方法，培养严谨的、良好的程序设计风格．所以，我们的出发点是明确的，即不仅仅教会读者某一门语言，而是要培养程序设计的能力．基于这个出发点，在编写本书时，突出以下几点:

第一，体现编程能力四要素．根据“程序=计算机语言+算法+数据结构+结构化程序设计技术”的科学公式，以算法和数据结构为基础，以结构化程序设计技术为指导，以某一门计算机语言为工具，努力培养编程能力．

第二，在每部分以具体实例为引例，导出C语言的相关语法规则．

第三，书中不能包含C语言的全部内容，否则会导致内容的分散，抓不住程序设计的精髓和C语言在程序设计、实现过程中的真正特点．

全书共分为九章，第一章全面介绍程序设计的基本概念、过程、方法和结构化程序设计原理；第二章介绍程序设计的基本要素，即数据类型、运算符和表达式；第三章介绍程序设计的三种控制结构：顺序结构、选择结构和循环结构，并以C语言为主，介绍实现这三种控制结构的语句；第四章介绍构造数据类型——数组的定义和应用；第五章介绍模块化程序设计的方法，以C语言的函数为背景阐明了使用计算机语言进行模块化程序设计的思路；第六章介绍C语言的一个重要数据类型，即指针类型的概念、定义和应用；第七章介绍C语言另外两类构造数据类型——结构体和共用体；第八章介绍存储数据要用到的数据文件的概念以及文件的建立和使用；第九章介绍程序设计过程中一个很重要的方面，即程序的调试方法，并以C语言为例，介绍调试过程中常见的错误及解决方法．

第一、二、三、九章、第五章第六节由赵宏编写，第四、六、七章由靳小燕编写，第五章第一～五节、七节、第八章由苏玉心编写，全书由赵宏主编，宋开璠教授主审并定稿．在本书的筹划和编写过程中，得到了北方交通大学计算中心各位领导及多位从事计算机基础教学的老师的帮助，在此一并表示感谢．

参加编写的的作者都是多年来从事非计算机专业计算机基础教学一线的教师，有一定的教学经验，也尽了最大努力，但毕竟水平有限，书中难免有疏漏和不妥之处，请广大读者批评指正．

作　者

1997年10月

目　录

第一章　程序设计概述

本章涉及的内容非常广泛，主要介绍计算机语言和程序、程序设计的步骤、结构化程序设计原理、算法的概念、程序实现的软、硬件环境、C 语言概述等内容。通过这些内容的学习，可以对程序设计有一个概貌的了解，为以后各章的学习打下基础。学习本章时要掌握主要的概念和方法，在后续各章的学习中反复加深对这些概念的理解，反复运用这些原理和方法，从而提高程序设计的能力。

第一节　计算机语言和计算机程序

一、计算机语言

计算机是人们处理信息的一种重要工具，它在人的控制下，按照人的意志正确地工作。人给机器一个指令，机器就完成一个操作。如果把一系列指令输入计算机存储起来，计算机就能按照指令序列实现操作的自动化。人和计算机之间的这种通信必须使用人工设计的语言，即计算机语言。

自计算机诞生以来，计算机语言的发展经历了四代：

第一代计算机语言——机器语言。本质上计算机只能识别“0”和“1”这样的二进制信息，机器语言的程序全部由“0”和“1”表示出来，例如，一个 16 位的计算机，由 16 个二进制数组成一条指令，这些指令叫机器指令。16 个 0 和 1 可以组成 2^{16} 个不同的指令或信息，这些指令的集合叫机器语言。人们用这种语言编写出的程序非常繁琐，而且不论阅读程序、调试程序都非常困难。另外，机器语言依赖于具体机器，而不是通用的，因此只有在早期使用这种语言，现在已经没有人直接使用这种语言编程了。但机器语言是计算机能直接识别和执行的唯一语言，因此其他各种语言编写的源程序最终都要通过语言处理程序翻译成等价的机器语言程序——目标程序后，计算机才能执行。

第二代计算机语言——汇编语言。为了克服机器语言的缺点，人们用一些特殊的符号（即助记符）来表示机器指令，例如用 ADD 代表“加”，用 SUB 代表“减”。这些助记符的使用增加了一点汇编语言的可读性。汇编语言的语句与计算机硬件操作有一一对应关系，每种汇编语言都是支持这种汇编语言的计算机所独有的。基本上有多少种计算机就有多少种汇编语言，因此汇编语言同机器语言一样也是面向机器的，通用性较差。尽管如此，汇编语言一直被人们所使用，主要是由于其执行速度快、占用存储空间小、对硬件操作灵活等特性。

第三代计算机语言——高级语言。机器语言和汇编语言都属于低级语言，后来人们创造出高级语言，它非常接近于人类的自然语言和数学语言，它的一个语句往往对应几条机器指令。一般说高级语言不再是面向机器的了，而是面向过程的语言，即把解题过程的每一步用计算机语言的语句描述出来，再配上适当的语言处理程序，计算机就能执行。因此，

这种语言也称“算法语言”。现就常见的高级语言介绍如下：

FORTRAN 语言是科学和工程计算中使用的主要编程语言。FORTRAN 语言的主要缺点是不能直接支持结构化程序设计。

BASIC 语言简单易学，适用于小型事务处理程序。

COBOL 语言是商业数据处理中广泛使用的语言，由于它本身结构上的特点，使它能有效的支持与商业处理有关的、范围广泛的过程技术。它的缺点是比较繁琐。

PASCAL 语言是最早出现的一种结构化语言，具有丰富的数据类型、严谨的语法结构。PASCAL 语言已广泛用于科学、工程和系统程序设计中。

C 语言作为 UNIX 操作系统的主要使用语言，由于 UNIX 操作系统的成功，C 语言得到了广泛使用。C 语言具有很强的功能和高度的灵活性。和其它高级语言一样，C 语言能提供丰富的数据类型及丰富的供运算和数据处理用的运算符。

ADA 语言是一种工程化的语言。ADA 语言在语言结构上是 PASCAL 型的，但它有完善的实时处理特性，包括中断处理、多重任务和进程通讯。

高级语言有很多种，不管哪种高级语言源程序都必须经过相应的语言处理程序翻译成机器指令才能执行。

第四代语言——非过程化语言。高级语言属于过程化语言，用它去解决一个问题，必须详细描述解决问题的每一步，既要解决“做什么”，又要解决“怎么做”。非过程化语言可以使不熟悉计算机的人方便地使用计算机，用户不必描述繁琐的解决问题过程，只需告诉计算机“做什么”而不必指明“怎么做”。FORTH 、SQL 等语言就属于非过程化语言。

二、计算机程序设计

1．程序和程序设计概念

程序并不是计算机程序设计中独有的概念，在日常生活中我们也常见到这个词，例如一个会议的日程、一场演出的节目单等，这些程序都是由人的一项一项活动组成，有序地完成每一项活动也就实现了程序的目标。计算机工作方式有两种，一种是交互式的，即人给机器一个指令，机器就完成一个操作。另外一种是程序控制式的，即把计算机要完成的操作用一条条指令按序排好，计算机一步一步执行了这个指令序列，也就完成了我们希望它做的事情，而且整个指令序列执行过程中不需要人来干预。为了解决某一特定问题用某一种计算机语言编写的指令序列称为程序。程序是程序设计的结果，在执行程序前必须先排定程序，排定以时间为进程必须完成的各种操作叫程序设计。

在用高级语言进行程序设计时要注意以下三个概念：

（1）语法。每种计算机语言都有自己的语法规则，这些规则是非常严格的。在进行编译时系统会按语法规则严格检查程序，如有不符合语法规则的地方，计算机会显示语法有错信息。

（2）语义。即某一语法成分的含义。例如，C 语言中用“ int ”定义整型变量，用“ char ”定义字符型变量，用“ while ”语句实现循环，用“*”表示乘法，用“！＝”表示不等运算等等。在使用时必须正确了解每一种语法成分的正确含义。

（3）语用。即正确使用语言。要善于利用语法规则中的有关规定和语法成分的含义有效地组成程序以达到特定的目的。

2．程序设计的准则

我们进行程序设计的目的是获得一个高质量的程序，那么什么样的程序才能算是一个好程序呢？度量程序质量的标准是什么？在计算机发展的早期，计算机内存容量有限，速度也不高，所以在程序设计时特别注重效率，近年来随着计算机软、硬件技术的不断发展及程序复杂性的增加，人们倾向于对程序质量做全面的评价。

程序的质量难以定量地度量，我们从程序的运行、程序的修改和程序的转移三个方面给出一些程序设计定性的准则。见图 1-1 。

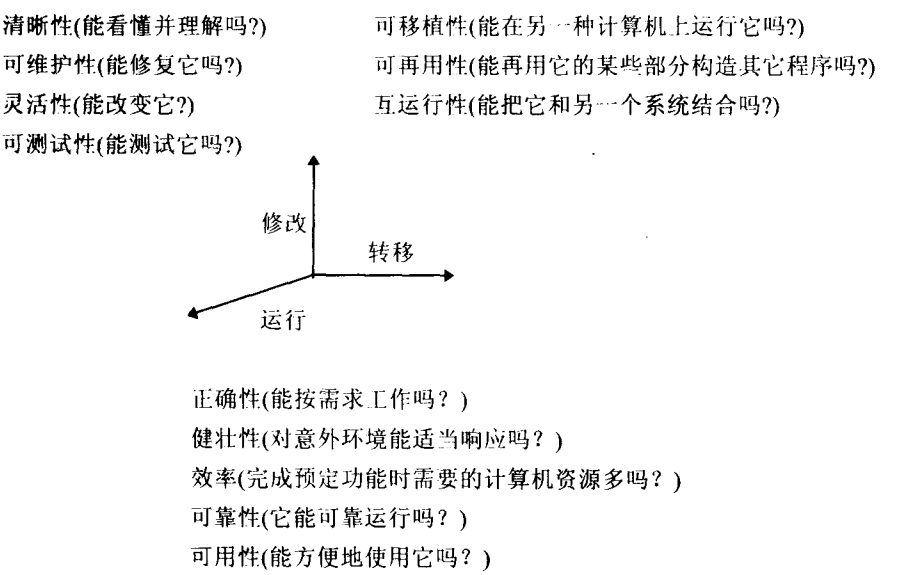


图 1-1 程序设计的准则

这些准则是相互联系、相辅相成的，但有时也可能是矛盾的。例如，片面强调执行效率，设计出来的程序就可能结构复杂，难于理解，也难于修改维护，追求可靠性一般也要以一定的时间和空间作为代价。

第二节 程序设计的过程

对于一个不太复杂的问题，设计一个程序实际要经过以下几个步骤：

- 建立数学模型
- 选定算法，用适当工具描述算法
- 编程
- 测试及调试

一、建立数学模型

提起数学模型，很多人认为就是一个数学公式。其实数学公式只是数学模型的一种，一张关系图、一张二维数据表也都是数学模型，因为它们规定了数据间准确的关系。

建立数学模型是程序设计中最复杂、最困难的一步，好的数学模型本身就是一个定律，例如惯性系统中力和运动间的关系有 Newton 定律：

$F = ma$

它要通过大量观察、分析、推理、验证等工作，还要加上人的天才才可以得到，我们往往不可能每个人都去发现定律，在进行程序设计时一般是利用已有的基本数学模型去构造出本问题的模型。各种数学，如微积分、运筹学、图论、高等代数等都是不同的基本数学模型。我们要善于把客观世界的问题归结到某种基本模型上，同时要在不同情况下利用不同的基本模型。

二、选定算法，用适当的工具描述算法

算法就是解决问题的方法与步骤。构造出数学模型后，具体算法还有很多种可供选择。例如用公式 $y=x^2-2x+3$ 计算 x=0，1，2，3，4，5 所对应的 y 值，它的模型即为一个数学公式，一种比较原始的算法是：

(1) 置 x 为 0；
(2) 计算 y 值；
(3) 输出 x,y 值；
…
(18) 置 x 为 5；
(19) 计算 y 值；
(20) 输出 x，y 值；
(21) 停止。

这样的算法虽然正确，但很繁琐，而且如果要求 x=0，1，2，…，100 所对应的 y 值时要写 304 个步骤，显然是不可取的。

由于对不同 x 值计算 y 值的过程是一样的，所有计算机高级语言中都有循环语句，因此可以使用如下算法：

(1) 置 x 为 0；
(2) 置 x 的上界 n=5；
(3) 当 $x \leqslant n$ 时,重复执行(3.1)，(3.2)，(3.3)步，否则，算法停止。
(3.1) 用公式计算 y 值；
(3.2) 输出一组 x 和 y 的值；
(3.3) x 值增加 1。

这个算法不仅简洁，而且还有很好的通用性和灵活性。如果要求计算 x=0，…，100 所对应 y 的值，只需在第 2 步把 x 的上界 n=5 改为 n=100 即可。

一个算法的好坏直接影响到一个程序的质量。

三、编程

编程就是将选定的算法从非计算机语言的描述形式转换为计算机语言的语句形式描述出来。将上述算法转换为 C 语言的程序：

```
/*c11*/
main()
```

```
{ int x,y,n;                    /*定义 x,y,n 为整型*/
 x=0;n=5;                       /*置 x 初值为 0,n 初值为 5*/
 while  (x<=5)                  /*当 x 值小于 5 时执行循环*/
   {
    y=x*x-2*x+3;                /*计算 y 值*/
    printf("%d%d",x,y);         /*打印一组 x,y 值*/
    x++;                        /*x 值增加 1*/
    }
}
```

四、测试及调试

编程完成以后，首先应静态审查程序，即由人工“代替”或“模拟”计算机，对程序进行仔细检查，然后将高级语言源程序输入计算机，经过编译、连接，然后运行。在编译、连接及运行时如果在某一步发现错误，必须找到错误并改正以后再从相应的地方开始上述过程，直到得到正确结果为止。

计算 $y=x^2-2x+3$ 的值这个例题中，数学模型是一个很简单的公式，相比较来说，有些问题要用较多的数学表达式且每个表达式间关系较复杂，或者根本没有现成的数学表达式，需要设计者综合数学知识、专门知识和经验构造出模型来。

例 1.1：有三辆卡车，需指派到三个不同的目的地，各种指派的成本见表 1-1，试求能使总成本最低的最优指派。表中 A_i（i=1,2,3）代表三辆卡车，B_j（j=1,2,3）代表三个目的地。

表 1-1 运输指派成本表

	B_1	B_2	B_3
A_1	40	～	37
A_2	24	31	39
A_3	29	37	～

注：“～”表示不能指派

显然这个问题无现成公式可用。通过分析，我们把这个现实问题归结为线性规划模型：

$Min\ Z=C_{ij}X_{ij}$ ——总费用最小

$\Sigma\ X_{ij}=1 \quad i=1,2,3$ ——一个目的地只能指派一辆车

$\Sigma\ X_{ij}=1 \quad j=1,2,3$ ——一辆车只能去一个目的地

$X_{ij}=0$ 或 1

其中 C_{ij} 代表第 i 辆卡车到第 j 个目的地的费用

X_{ij} 代表第 i 辆卡车是否指派到第 j 个目的地，取值为 0 或 1

考虑到网络模型的直观性、形象性，本问题也可归结为网络模型，然后利用网络算法求解（图 1-2）：

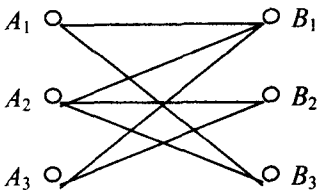


图 1-2 网络模型

第三节 程序设计的思维方法——算法

一、算法的概念

学习计算机语言的目的是利用该语言工具设计出可供计算机运行的程序。在进行程序设计时，要掌握四大要素：即计算机语言、算法、数据结构和结构化程序设计技术。其中最关键的一个要素就是设计算法。广义地说所谓算法就是解决问题的方法和步骤，程序设计依赖于算法，而算法是不依赖程序独立存在的。只要掌握了算法设计，就可以用学过的任何一种计算机语言去实现这个算法，只要有了正确的算法，实现起来并不困难。

对同一个问题，可以有不同的解题方法和步骤，即有不同的算法。程序的质量主要取决于算法的质量。例如，求S=1－1/2＋1/3－1/4…＋1/99－1/100，第一种方法先计算1－1/2，再加1/3，再减1/4，一直到1/100；第二种方法是先计算S_1=1+1/3+…+1/99，再计算S_2=1/2+1/4+…+1/100，则$S=S_1-S_2$。还可以有别的算法，这些算法虽然都达到了同一个目的，但有的算法执行的步骤多，有的算法执行的步骤相对较少。因此为了有效地进行解题，不仅要保证算法正确，还要考虑算法的质量。

下面举一个例子，使读者对什么是算法有个初步的了解。

例1.2：给定两个正整数m和n，求它们的最大公因数。

算法如下：

(1) 输入两个正整数m，n；

(2) 如果$m<n$，则交换m，n的值；

(3) 求m除以n的余数r；

(4) 如果$r\neq0$，则重复执行(4.1)，(4.2)，(4.3)步，否则打印n，算法终止。

(4.1) 把n值赋给m；

(4.2) 把r值赋给n；

(4.3) 求m除以n的余数r。

例如m=132，n=341，因为$m<n$，所以交换m，n，置m=341，n=132。341除以132，商是2，余数r为77，因为$r\neq0$，所以置m=132，n=77，依次类推，最后22除以11时，r=0，算法终止，132和341的最大公因数是11。

计算机算法包括数值算法和非数值算法两大类：数值运算的主要目的是求数值解，例如求方程的根，求函数的定积分，求函数的最大值、最小值等；非数值运算的应用也十分广泛，例如计算机售票、办公自动化、图书情报的检索等。

二、算法的特征

并不是任何解决问题的方法、步骤都能称得上是计算机算法，一个真正的算法必须满足以下五个特征：

（1）有穷性。一个算法必须在执行有穷步骤之后结束，而不能是无限的。例1.2满足了这个特征，它的执行导致一个递减的正整数序列，必然要终止。

（2）确定性。算法的每一个步骤必须是确定的，不应当是含糊的，模棱两可的。对于每种情况，有待执行的动作必须严格规定，不能出现二义性。例如对例1.2而言，确定性意味着*m*和*n*都是正整数，两个正整数相除余数仍是正整数，否则例1.2就是不确定的。

（3）有0个或多个输入。所谓输入是指在算法开始执行之前，对算法中的输入变量所给的初始值。例如，在执行例题1.2 时有两个输入，即*m*和*n*。也有的算法可以没有输入。

（4）有一个或多个输出。一个算法的目的就是为了求解。没有输出，这个算法也就没有意义了。例1.2有一个输出，即*m*和*n*的最大公因数。

（5）可行性。算法中的每一个步骤都能精确地进行，而且进行有穷次即可完成。例如例题1.2中用整除测试一个整数是否为0，置一个变量的值等于另一个变量的值，这些运算都是可行的，结果也是确定的。

三、算法的表示方法

可以采用很多工具来描述一个算法，它们可以分为图形、语言、表格三类。不论是哪一类表示方法，对它们的基本要求是能够提供对算法没有二义的描述，从而在编程阶段能把设计的算法直接翻译成程序。常用的工具有流程图、N—S图、PAD图、过程设计语言（PDL）、判定表等。

1．流程图

流程图是用一些图框来表示各种类型的操作，它的主要优点是描绘很直观，初学者容易掌握。它的缺点是对流程线的使用没有严格限制，使用者可以毫不受限制地使流程任意转来转去，使流程图变得毫无规律，难以阅读。目前，越来越多的人已经不再使用这种流程图了，这里仅用它来描述第三节中例1.2的算法。

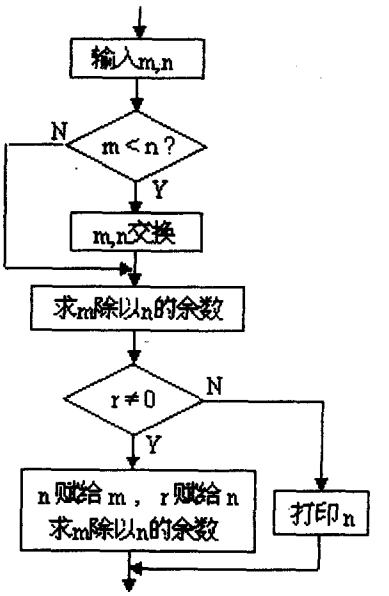


图1-3 流程图

2．N—S图

N—S图是美国学者I.Nassi和B.Shneiderman提出的一种新的流程图。N—S图取消了流程线，全部算法写在一个矩形框内，该框内还可以包含其它从属于它的框。N—S图是一种结构化流程图，根据N—S图编写的程序一定是结构化的程序。见图1-4。

3．PAD图

PAD是问题分析图（Problem Analysis Diagram）的缩写，它用二维树形结构的图来表示程序的控制流，描述的程序结构非常清晰，将这种图翻译成程序比较容易。描述例1.2算法的PAD图见图1-5。