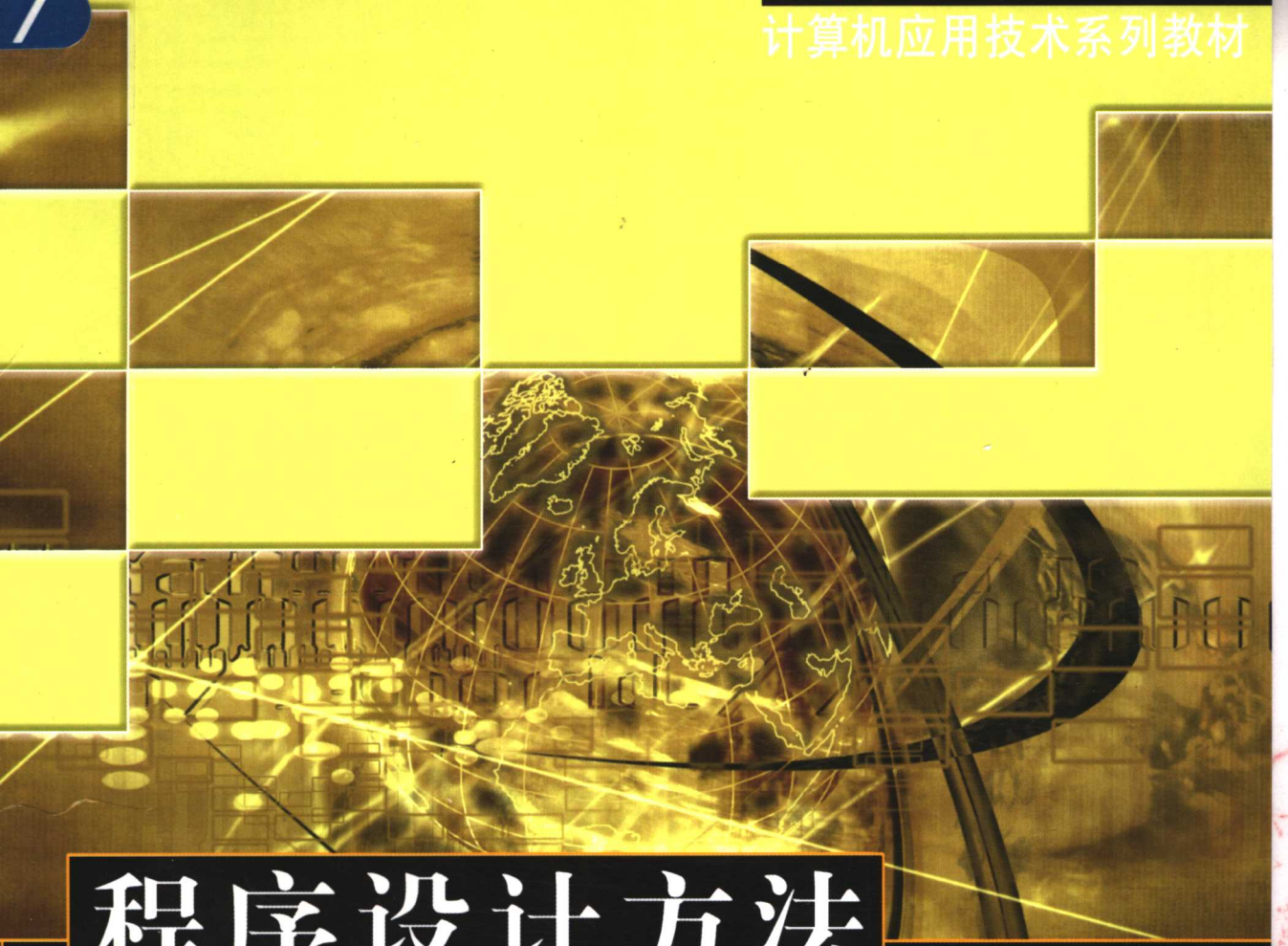
计算机应用技术系列教材



程序设计方法与案例分析

高　林　周海燕　主编

林志英　魏雪英　编著

人民邮电出版社
POSTS & TELECOM PRESS

计算机应用技术系列教材

程序设计方法与案例分析

林志英　魏雪英　编著

人民邮电出版社

图书在版编目（CIP）数据

程序设计方法与案例分析/林志英，魏雪英编著. —北京：人民邮电出版社，2005.2
ISBN 7-115-12833-2

Ⅰ. 程… Ⅱ. ①林… ②魏… Ⅲ. 程序设计 Ⅳ. TP311.1

中国版本图书馆 CIP 数据核字（2004）第 136198 号

内 容 提 要

本书以 C 语言为例介绍了程序设计的基础知识和方法。全书共分为 7 章。第 1 章介绍程序设计的基本知识，第 2 章介绍结构化程序设计的概念和方法，第 3 章介绍程序的风格，第 4 章介绍几个常用算法，第 5 章介绍数据结构，第 6 章介绍测试，第 7 章为实训指导。为了便于教学，书中配有相当数量的例题和一定数量的习题。

本书内容较丰富，语言通俗，实用性强，可作为高职高专学生的教材和相关专业教学参考书，也可供自学者学习参考。

计算机应用技术系列教材

程序设计方法与案例分析

◆ 编　　著　林志英　魏雪英
责任编辑　潘春燕

◆ 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　010-67129259
北京朝阳展望印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：11.75
字数：279 千字　　2005 年 2 月第 1 版
印数：1 – 5 000 册　　2005 年 2 月北京第 1 次印刷

ISBN 7-115-12833-2/TP・4320

定价：16.00 元

丛书前言

当今人类社会正经历着一场信息化革命。从1946年发明第一台计算机开始，人类社会经历了20世纪70年代的微机革命和90年代的网络革命。以计算机技术为基础，以信息技术为动力，以信息产业为带头产业，迅速形成了推动社会经济发展的空前强大力量，从而使知识经济初显端倪，信息社会即将到来。过去10年，以通信和计算机网络为基础的信息化基础建设已初具规模，计算机和网络设备制造业初步建成并形成一定的生产规模。面向新世纪之初，我国的信息化革命将进入以计算机应用为主题的新时代。计算机信息系统、电子政务、电子商务、办公自动化、远程教育、家庭计算机应用等在我国社会、经济，以及人们的生活、学习等各领域将逐步普及。

在计算机应用时代，需要培养大量的掌握计算机应用技术的人才。其中既包括信息产业的从业人员，也包括用信息技术改造传统产业的、各行各业都需要的计算机技术人才，还包括提升人民生活水平、普及信息技术所需要的其他计算机人才。他们不仅包括高端的研究人才，企业高层管理人才，也包括各种初、中级工程应用人才，即能够把科研开发成果转化为现实产品的工程化人才。

本套教材的编写旨在为培养计算机应用技术人才打好基础。本套教材包括：

1.《C语言程序设计教程》

2.《Visual Basic6.0程序设计教程》

3.《数据库技术》

4.《程序设计方法与案例分析》

5.《计算机网络技术》

6.《多媒体计算机技术基础及应用》

7.《管理信息系统与案例分析》

本套教材的特点是：

1. 以掌握计算机应用技术的基本能力要求为主。

2. 以应用为目的，在写作中尽量做到从问题出发，采用提出问题，分析问题，解决问题的思路，导出必要的概念和方法。

3. 在教学手段上强调以技术训练、实际操作为主。

4. 通过大量的实例和实训练习，帮助读者掌握计算机的基本知识和操作方法。

本套教材为高等职业教育、高等专科教育、成人高等教育、高等教育自学考试信息技术类和计算机应用类专业教材，也可用作计算机技术的培训教材和从事计算机应用的技术人员的自学读本。

编者的话

在科学技术高速发展的信息时代，我国的计算机产业也迅猛发展，对专业人才的需求也日益迫切。因此，学习计算机知识已成为各界人士的需求，更是当代大学生知识结构中必不可少的组成部分。

程序设计是计算机专业领域中最核心的工作。在程序设计工作中，既要对问题进行分析；又要考虑各种设计的可能性；选择适当的算法、数据结构及语言；编写代码；对代码进行测试。本书就对这些问题进行了详细的讨论。

本书共分为 7 章。第 1 章介绍程序设计的基本知识，包括算法、算法的描述工具及模块的概念。第 2 章介绍结构化程序设计的概念和方法。第 3 章介绍程序的风格，好的风格对于程序是非常重要的。第 4 章介绍了几个常用算法，包括穷举、递推和递归算法。第 5 章介绍了数据结构，这也是程序设计课程中重要的组成部分。包括线性表、栈、队列、字符串、多维数组、树、图、查找和排序等内容。第 6 章介绍测试。第 7 章为实训指导，通过这章练习，使学生较好地掌握程序设计的基本方法。

本书的读者对象是已学过一门高级语言的学生。在写作特点上力求通俗易懂、尽量切合实际应用，并以丰富的例题进行讲解，引导读者逐步掌握程序设计的方法。

本书由林志英和魏雪英负责编写。林志英负责第 1、2、3、4、6、7 及第 5 章的部分内容；魏雪英参加了第 5 章的编写工作。

在本书的编写过程中，高林教授、周海燕副教授提出了很多有益的建议，在此表示衷心的感谢！

由于编者水平有限，疏漏难免，敬请广大读者批评指正。

编　者

2004 年 12 月

目　　录

第 1 章　程序设计基本原理

本章主要介绍程序设计的一些基本概念，包括算法的概念、算法的两种描述工具（流程图和 N-S 图）、模块的概念、模块的内聚度和耦合度、局部化和信息隐藏的概念。

什么是程序设计呢？程序设计是将事先确定的解题步骤，用机器所能理解的语言描述出来的过程。著名的瑞士计算机科学家 N.Wirth 教授曾提出：算法+数据结构=程序。由此可见，程序设计的实质是对问题选择一种适合的数据结构，设计一个好的算法。

1.1 算　法

1.1.1 算法的概念

什么是算法（Algorithm）呢？我们先看一个生活中的例子。

【例 1.1】 洗衣服。

人们洗衣服的时候通常按下面的步骤进行。

Step 1：浸泡；

Step 2：洗涤；

Step 3：漂洗；

Step 4：拧干；

Step 5：晾晒。

在生活中，做事情需要一步步地完成，在使用计算机解决问题的时候，也需要按照一定的步骤来完成。通俗地讲，算法是对问题求解步骤的一种描述，是一种解题的方法。严格地说，算法是由若干条指令组成的有穷序列。

【例 1.2】 交换两个变量 a，b 中的值。

要交换两个变量中的值，必须通过一个中间变量来完成，因此我们引入一个中间变量 t，交换的步骤如下。

Step 1：输入变量 a 和 b 的值；

Step 2：将 a 的值赋给 t（用 $t \Leftarrow a$ 表示）；

Step 3：将 b 的值赋给 a（用 $a \Leftarrow b$ 表示）；

Step 4：将 t 的值赋给 b（用 $b \Leftarrow t$ 表示）；

Step 5：输出变量 a 和 b 的值；

Step 6：结束。

上面的解题步骤就是交换两个变量值的算法。

【例 1.3】求 1 到 n 之间所有自然数的和。

Step 1：输入 n（自然数）；

Step 2：$sum \Leftarrow 0$；

Step 3：$i \Leftarrow 1$；

Step 4：当 i 的值小于等于 n 时，重复执行下面的操作

Step 4.1　$sum \Leftarrow sum+i$；

Step 4.2　$i \Leftarrow i+1$；

Step 5：输出 sum；

Step 6：结束。

对同一个问题，可以有不同的解题方法。我们在解决问题的时候，不仅要保证算法的正确，还要选择适合的算法。

1.1.2　算法的特性

算法是由若干指令组成的有限序列，一个算法应具有下列 5 个重要特性。

1. 有穷性

一个算法必须是（对任何合法的输入值）在执行有穷步骤之后结束，且每一步都可在有穷时间内完成。

2. 确定性

算法中每一条指令必须有确切的含义，无二义性，即对于相同的输入只能得出相同的输出。

3. 可行性

算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现。

4. 输入

一个算法有 0 个或多个的输入，这些输入取自于某个特定的对象的集合。

5. 输出

一个算法有一个或多个的输出。

在一个算法中，有些指令可能是重复执行的，因此指令的执行次数可能远远大于算法中的指令条数。

算法的概念和程序相似，但二者是有区别的。程序中的指令必须是机器可执行的，算法中的指令不一定是机器可以执行的。将一个算法用一个机器可以执行的语言来描述，则该算法就成为一个程序。

1.1.3　算法的分析

1. 算法评价

在解决同一个问题时，可能有多种算法，如何来评价这些算法的好坏呢？一个好的算法应达到以下目标。

（1）正确性。算法应能正确满足具体问题的要求。

（2）可读性。算法应易于阅读和理解，以便于调试和修改程序。

（3）健壮性。当输入非法数据时，算法能做出适当的反应和处理，不会产生不需要的输出结果。

（4）效率。算法的效率包括时间效率和空间效率。时间效率是指执行算法时所耗费的时间。空间效率是指执行算法时所耗费的存储空间，除了包括数据本身所占的存储空间外，还包括算法所需的辅助存储空间。要想选择运行时间短、占用存储空间小的十全十美的算法是比较困难的，因为要节约算法的执行时间往往要以牺牲更多的空间为代价；为了节省空间往往是以花费更多的时间为代价。因此我们只能根据具体情况有所侧重。

2. 算法分析

算法的执行时间需要通过依据该算法编制的程序在计算机上运行时所耗费的时间来度量，而一个程序在计算机上运行时所耗费的时间取决于下列因素。

（1）问题的规模。

（2）依据算法选用的策略。

（3）书写程序的语言。

（4）机器执行指令的速度。

（5）对源程序编译的时间。

由于同一算法与实现算法的语言、采用的编译程序，以及所使用的计算机均有密切的联系，要精确地确定一个算法的执行时间是非常困难的。因此，在讨论算法的时间效率时，通常不考虑计算机硬件和软件等不确定因素，而用算法中所有语句的频度之和来表示一个算法所需要的时间。语句的频度（Frequency Count）是指该语句重复执行的次数。

【例 1.4】下面是两个 $n \times n$ 矩阵相乘的算法。

```
#define n 5
MatrixMlt(float a[n][n],float b[n][n],float c[n][n])
{
    int i,j,k;
(1) for(i=0;i<n;i++)
(2)     for(j=0;j<n;j++)
        {
(3)         c[i][j]=0;
(4)         for(k=0;k<n;k++)
(5)             c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
}
```

在该算法中，每条语句执行的频度如表 1-1 所示。

表 1-1　　矩阵相乘算法的语句频度

语　句	语句频度
(1)	$n+1$
(2)	$n(n+1)$
(3)	n^2
(4)	$n^2(n+1)$
(5)	n^3

该算法中耗费的时间为：

$$T(n)= n+1+ n(n+1)+ n^2+ n^2(n+1)+ n^3$$
$$=2n^3+3n^2+2n+1$$

从上式可以看出，矩阵相乘算法的执行时间是一个矩阵阶数 n 的函数。我们将问题的输入量称为问题的规模，如矩阵相乘算法的规模是矩阵的阶数。一般情况下，算法中的执行时间是问题规模 n 的某个函数 $f(n)$，算法的时间度量记作：

$$T(n)= O(f(n))$$

它表示随问题规模 n 的增大，算法执行时间的增长率和 $f(n)$ 的增长率相同，称作算法的渐近时间复杂度（Asymptotic Time Complexity），简称时间复杂度。

两个矩阵相乘算法的时间量级是 $T(n)=O(n^3)$。

算法常见的时间复杂度有：常数阶 $O(1)$、对数阶 $O(\log_2 n)$、线性阶 $O(n)$、平方阶 $O(n^2)$ 和指数阶 $O(2^n)$等。

与算法的时间复杂度类似，一个算法的空间复杂度（Space Complexity）定义该算法所需的存储空间，记作：

$$S(n) = O(f(n))$$

它也是问题规模 n 的一个函数。

1.2 算法描述工具

一个算法可以用自然语言、数学语言、流程图或伪码等形式来描述，上一节中的例题是用自然语言描述的。下面再介绍算法的两种描述工具：程序流程图和 N-S 图。

1.2.1 程序流程图

程序流程图（Program Flow Chart）也称程序框图，它独立于任何一种程序设计语言，比较直观、清晰，易于学习掌握。常用程序流程图的符号如图 1.1 所示。

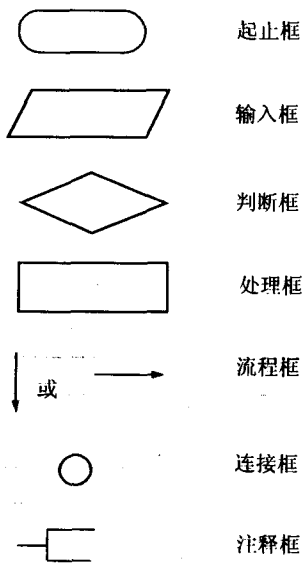


图 1.1 常用程序流程图符号

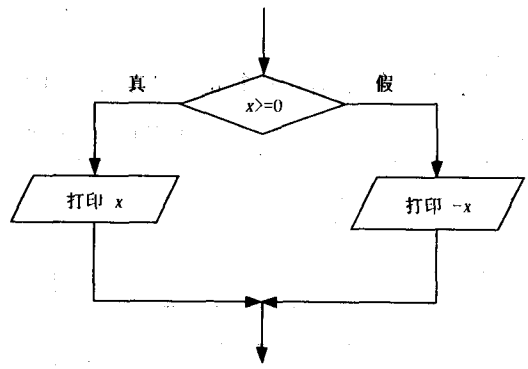


图 1.2 判断框的使用

图 1.1 中的判断框（菱形框）的作用是对一个给定的条件进行判断，根据给定的条件是否成立来决定如何执行其后的操作，如图 1.2 所示。

连接点的作用是将画在不同地方的流程线连接起来。图 1.3 中有两个以 1 为标志的连接点（在连接点圈中写上“1”），它表示这两个点是互相连接在一起的，实际上它们是同一个点，只是画不下才分开来画。用连接点可以避免流程线的交叉或过长，使流程图更加清晰。

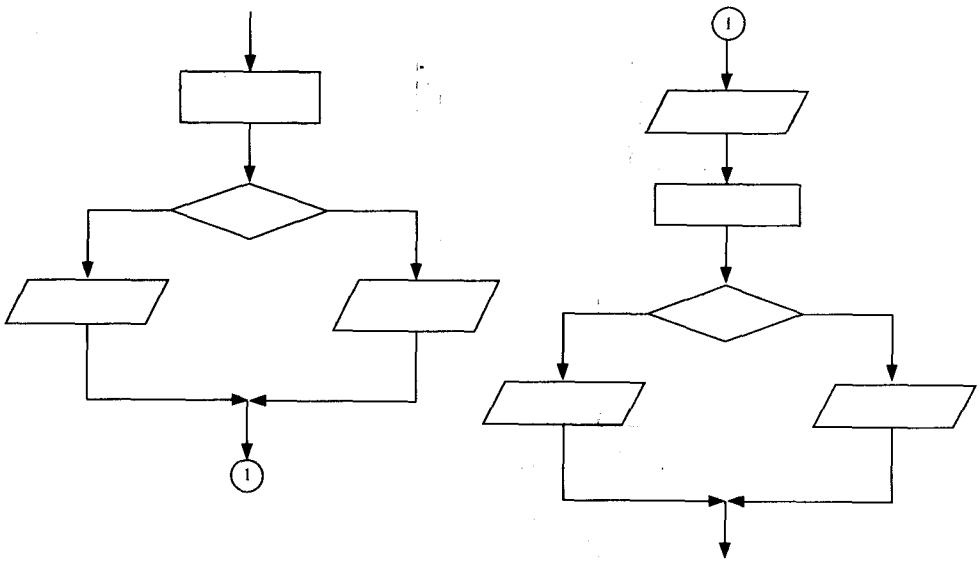


图 1.3　连接点的使用

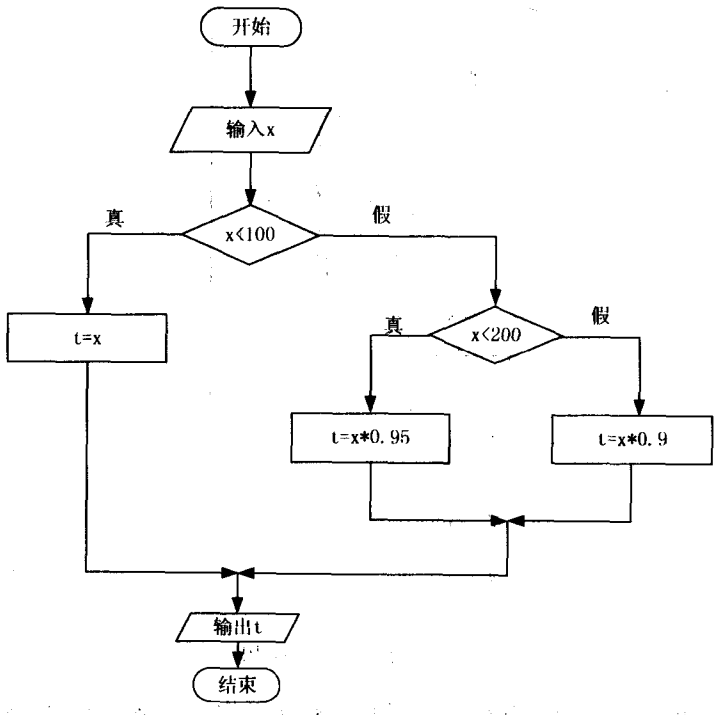


图 1.4　计算优惠价格的流程图

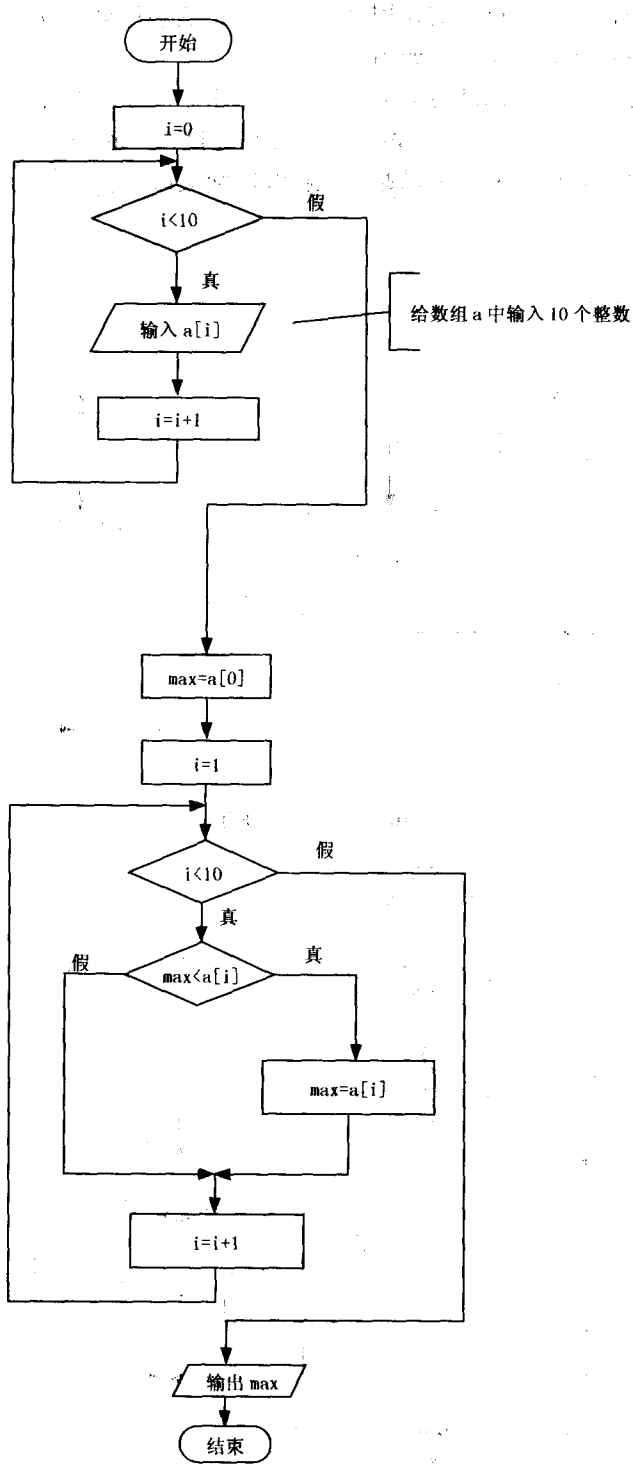


图 1.5　求最大值的流程图

注释框不是流程图中必要的部分，不反映流程和操作，只是为了对流程图中某些框的操

作做必要的补充说明，以帮助读者更好地阅读流程图。

用流程图描述下列算法。

【例 1.5】某商店为了促销，采用购物打折的优惠办法：每位顾客一次购物 100 元以下，没有折扣；100 元到 200 元之间，按九五折优惠；200 元以上，按九折优惠。

假设每位顾客的购物花费用 x 表示，计算折扣后用 t 表示实收的费用，算法的流程图如图 1.4 所示。

【例 1.6】一维数组 a 中存放着 10 个整数，查找其中的最大值并输出。

假设用变量 *max* 存放最大值，在一维数组中查找最大值的流程图如图 1.5 所示。

在画流程图时，需要注意的是不要忘了画流程线上的箭头，它反映了流程执行的先后次序。

使用流程图表示算法直观形象，能较清楚地显示出各部分之间的逻辑关系，并且学习起来非常容易。但它占用的篇幅较多，当算法较复杂时，流程图也变得复杂和庞大起来，不利于阅读。

1.2.2　N-S 流程图

1973 年美国学者 Nassi 和 Shneiderman 提出了新的流程图形式，称为 N-S 图，也称盒图。在这种流程图中，去掉了带箭头的流程线，全部算法写在一个矩形框内，在该框内还可以包含其他的从属于它的框。

N-S 图使用以下的流程图符号。

（1）顺序结构：如图 1.6 所示。A 和 B 分别表示两个语句块，A 和 B 依次执行。

（2）分支结构：如图 1.7 所示。当条件 P 成立时执行 A 操作，不成立时执行 B 操作。

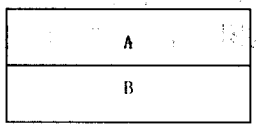


图 1.6　N-S 图的顺序结构

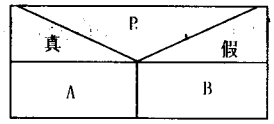


图 1.7　N-S 图的分支结构

（3）当型循环结构：如图 1.8 所示。当条件 P 成立时反复执行 A 操作，直到条件 P 不成立为止。

（4）直到型循环结构：如图 1.9 所示。反复执行 A 操作，直到条件 P 为真为止。

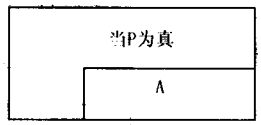


图 1.8　N-S 图的当型循环结构

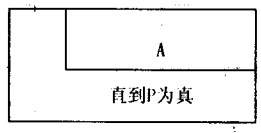


图 1.9　N-S 图的直到型循环结构

（5）多分支选择：如图 1.10 所示。

（6）调用子程序（函数）：调用某个名为 A 的子程序（或函数），如图 1.11 所示。

【例 1.7】用 N-S 图来描述例 1.5 中的问题。

例 1.5 中的问题用 N-S 图来描述如图 1.12 所示。

【例 1.8】判断自然数 n（n>1）是否为素数。

判素数的 N-S 图如图 1.13 所示。

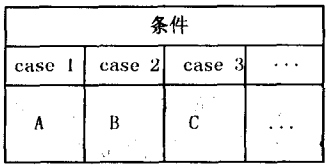


图 1.10 N-S 图的多分支结构

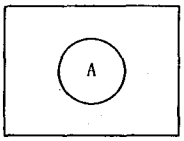


图 1.11 N-S 图的调用子程序（函数）

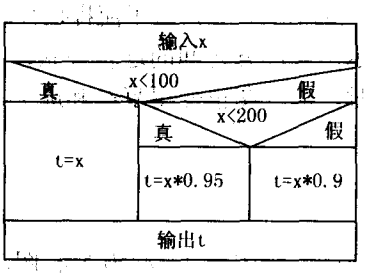


图 1.12 计算优惠价格的 N-S 图

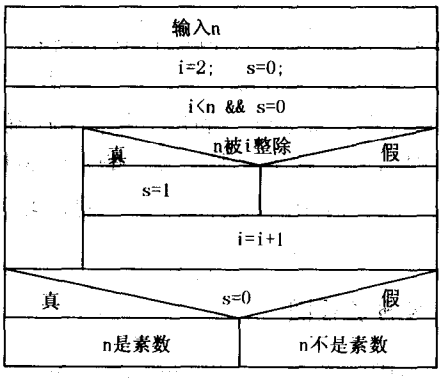


图 1.13 判素数的 N-S 图

1.3 模　块

我们首先看一个例子。下面列出的是某公司的职员名单：宋宁、张立、李小、王虹、黄明、王和、李强、赵磊、肖屏和钱小东。通过这个名单，可知这些人均是该公司的职员。如果换一种形式，给出如图 1.14 所示的公司组织结构图，就可以非常清楚地看到该公司的机构设置和人员归属。

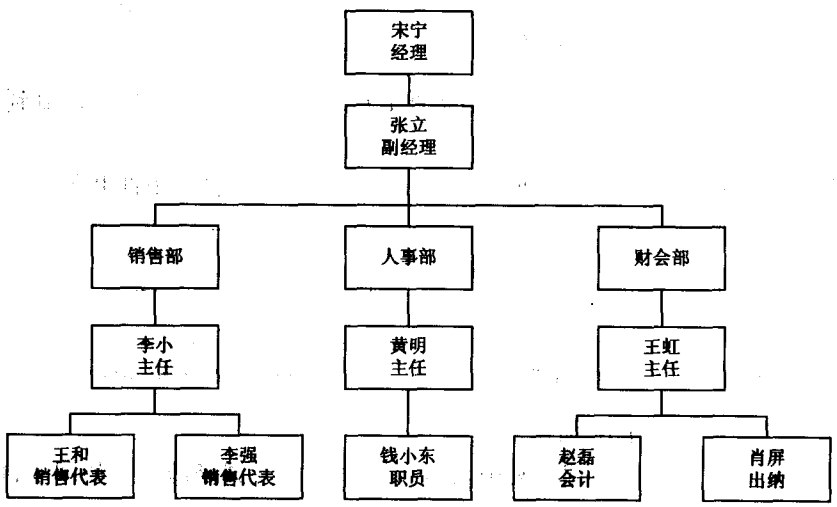


图 1.14 某公司组织结构图

软件的设计和上面的例子很相似，如果一个大型的软件完全由一个代码块构成，要调试和维护这段代码几乎是不可能的。因此，需要将整个代码划分成一些较小的部分，这些部分