计算机基础教育丛书

C程序设计实验指导

清华大学出版社

徐士良 编著

C 程序设计实验指导

徐士良 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是与谭浩强教授编著的《C 程序设计》一书配套的实验教材，也可与 C 语言的其它教材配套使用。内容包括程序设计概念、基本操作环境、Turbo C 编译环境、上机实验内容四大部分。所有实验均按内容分类，最后还安排了综合训练的几个实验供读者选用。

本书可作为大专院校《C 程序设计》及其它 C 语言课程的实验教材，也可作为自学 C 程序设计的实验参考书或培训用的实验教材。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目(CIP)数据

C 程序设计实验指导/徐士良编著. —北京：清华大学出版社，1997.9
(计算机基础教育丛书/谭浩强主编)
ISBN 7-302-02621-1

Ⅰ.C… Ⅱ.徐… Ⅲ.C 语言-程序设计 Ⅳ.TP312C

中国版本图书馆 CIP 数据核字(97)第 15848 号

出版者：清华大学出版社(北京清华大学校内，邮编 100084)
印刷者：北京昌平环球印刷厂
发行者：新华书店总店北京科技发行所
开　本：787×1092　1/16　印张：8.25　字数：205 千字
版　次：1997 年 9 月第 1 版　1997 年 9 月第 1 次印刷
书　号：ISBN 7-302-02621-1/TP・1344
印　数：0001～8000
定　价：9.00 元

《计算机基础教育丛书》出版说明

近年来，我国的计算机应用事业迅速发展，大批科技人员、大中学生、管理人员以及各行各业的在职人员都迫切要求学习计算机知识，他们已经认识到，计算机知识是当代知识分子的知识结构中不可缺少的重要部分。

计算机应用人才的队伍由两部分人组成：一部分是从计算机专业毕业的计算机专门人才，他们是计算机应用人才队伍中的骨干力量；另一部分是各行各业中从事计算机应用的人才，他们既熟悉本专业的业务，又掌握计算机应用的技术，人数众多，是计算机应用人才队伍的基本力量。他们掌握计算机知识的情况和应用计算机的能力在相当大程度上决定了我国计算机应用的水平。因此，在搞好计算机专业教育的同时，在广大非计算机专业中开展计算机基础教育是十分必要的。

非计算机专业中的计算机教学，无论就目的、内容、教学体系、教材、教学方法等各方面都与计算机专业有很大的不同，它以应用为目的，以应用为出发点。如果不注意这个特点，将会事倍功半。广大非计算机专业的师生、在职干部迫切希望有一套适合他们的教材，以便循序渐进地迈入计算机应用领域，并且不断地提高自己的水平。我们在前几年陆续编写了一些适合初学者使用的教材，受到广大群众的欢迎。许多读者勉励我们在此基础上进一步摸索和总结规律，为我国的广大非计算机专业人员编写一整套合适的教材。

近年来，全国许多专家、学者在这个领域作了有益的探索，写出了一批受到群众欢迎的计算机基础教育的教材。特别是全国高等学校计算机基础教育研究会作了大量的工作，在集思广益的基础上，提出了在高等学校的非计算机专业中进行计算机教育的四个层次的设想，受到广泛的注意和支持。我们认为：计算机的应用是分层次的，同样，计算机人才的培养也是分层次的；非计算机专业中各个领域的情况不同，也不能一律要求，在进行计算机教育时也应当有不同的层次。对于每一个学习计算机知识的人，还有一个由浅入深、逐步提高的过程。

我们认为，编辑出版一套全面而有层次的计算机基础教育的教材，目前不仅是十分必要的，而且是完全有条件的。在全国高等学校计算机基础教育研究会和许多同志的积极推动及清华大学出版社的大力支持下，我们决定编辑《计算机基础教育丛书》。它的对象是：高等学校非计算机专业的学生、计算机继续教育或培训班的学员、广大在职自学人员。

本丛书包括计算机科学技术的一些最基本的内容，例如计算机各种常用的高级语言、计算机软件技术基础、计算机硬件技术基础、微型计算机的原理与应用、算法与数据结构、数据库基础、计算机辅助设计基础、微机网络与应用、系统分析与设计等，形成多层次的结构，读者可以根据需要与可能选学。

本丛书的宗旨是针对广大非计算机专业的需要和特点来组织教材，从实际出发，用读者容易理解的体系和叙述方法，深入浅出、循序渐进地帮助读者更好地掌握课程的基本内容。

希望我们的丛书能在这方面具有自己的风格。在实践中接受检验。

本丛书的作者大多数是高等学校中有较丰富教学经验的教师。但是，由于计算机科学技术的飞速发展以及我们的水平有限，丛书肯定会存在许多不足，丛书的书目和内容也应当不断发展和更新。我们热情地希望得到社会各界和广大读者的批评指正。

主编　谭浩强　林定基　刘瑞挺

1988.10

此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

实验是学习程序设计语言的一个重要环节。本书是与谭浩强教授编著的《C 程序设计》一书配套的实验教材，也能与其它有关 C 语言的教材配套使用。考虑到程序设计是学习任何一种计算机语言所必备的方法，在本书中专门安排了一章关于程序设计方面的内容。读者在实验过程中也应重视程序设计的基本方法，通过实验掌握程序设计的基本过程和基本的调试技术。

全书共分四章。

第 1 章集中介绍了程序设计的基本概念。包括：程序设计的基本过程，程序设计的基本方法，程序设计语言，程序设计的风格，程序的调试。

第 2 章介绍了实验所必需的基本操作环境。包括：文件、目录与路径的基本概念，DOS 操作系统的基本命令，实验中常用的一些编辑键与功能键的使用，两种常用的文本编辑程序 ED 与 EDIT，C 程序的编辑、编译连接与运行的基本过程。

第 3 章比较详细地介绍了 Turbo C 编译环境。包括：Turbo C 命令行编译方式和 Turbo C 集成编译环境，并用实例详细叙述了在这两种方式下的编译过程。

第 4 章为具体的实验内容。包括：基本操作练习，C 基本程序的设计与调试，综合训练。在基本操作练习中，共安排了三个实验，它们是：文本编辑与指法练习，DOS 基本命令的使用，C 程序的输入，编译连接与运行。在 C 基本程序设计与调试这一部分中，按顺序结构、选择结构、循环结构、模块化程序设计、数组、指针、结构体、文件等内容，共安排了 8 个实验。在综合训练这部分中共安排了 8 个实验，供教师或读者选用。

书中对每个实验都明确规定了实验目的，提出了具体要求，有的还给出了方法说明。

在附录中给出了 Turbo C 的编译错误信息和 Turbo C 常用库函数，供读者查阅。

本书的编著与出版得到了谭浩强教授的支持，他还对编写本书提出了一些指导性的意见，在此表示衷心的感谢。

虽然本书是作者在教学实践的基础上编写而成的，但由于时间仓促与作者的水平有限，书中难免有不足甚至是错误之处，恳请读者批评指正。

作　者

1997 年 7 月

目　录

第1章　程序设计概念

1.1　程序设计的基本过程

什么是程序设计?对于初学计算机的人来说,往往把程序设计理解为简单地编制一个程序。其实这是不对的,至少是不全面的。实际上,程序设计包括多方面的内容,而编制程序只是其中的一个方面。有人将程序设计描述成如下公式:

程序设计＝算法＋数据结构＋方法＋工具

由此可以看出,在整个程序设计的过程中,要涉及到算法的设计、数据结构的设计、方法的设计和设计工具的选择等几个方面。从这个基本概念出发,一般来说,程序设计的过程可以分为以下五个基本步骤:

(1) 问题的分析;

(2) 结构特性的设计;

(3) 算法的设计;

(4) 流程的描述;

(5) 调试与运行。

下面分别对这五个步骤作简要的介绍。

1.1.1　问题分析

问题分析是进行程序设计的基础。如果在没有把所要解决的问题分析清楚之前就想着手编制程序,是很难得到预想结果的,这只能起到事倍功半的效果。根据所要解决的问题性质与类型,需要分析的内容可能是不同的,但作为最基本的分析内容主要有以下几个方面。

1. 问题的性质

人们所要解决的问题是各种各样的,而对于不同性质的问题,所用的方法、工具以及输入输出的形式一般也是不同的。通过对问题性质的分析,需要解决的问题是各种各样的。例如,你所要解决的问题是属于数值型还是属于非数值型的问题;如果是数值型的问题,则要求确定一个合理的精度要求;不管是数值型问题还是非数值型问题,都需要明确最终的结果是什么。如对于一元二次方程求根的问题,需要明确是只需要实根还是实根与复根都需要。

2. 输入/输出数据

数据处理是计算机应用最广泛的领域。在用计算机解决问题时,一般总要有一些输入数据,计算的结果也要以某种方式进行输出。因此,在进行程序设计的过程中,需要考虑对输入/输出数据的处理。一般来说,对于输入/输出数据主要应考虑以下几个方面:

(1) 数据的类型是什么?如整型、实型、双精度型、字符型等。

(2) 在何种设备上进行输入或输出?

(3) 采用什么样的格式进行数据的输入或输出?

3. 数学模型或常用的方法

对于数值型问题，一般要考虑数学模型的设计，或者要对常用的一些方法进行分析和比较，从而根据问题的性质选择一种合理的方案。对于非数值型的问题，通常也需要从众多的方法中选择一种合适的方法。

例如，为了求一元二次方程 $Ax^2+Bx+C=0$ 的两个实根 x_1 和 x_2，通常有以下三种方法：

(1) 求根公式

$$x_{1,2}=(-B\pm\sqrt{B^2-4AC})/(2A)$$

(2) 韦达定理

$$x_1+x_2=-B/A$$
$$x_1x_2=C/A$$

(3) 迭代法

对于上述三种方法，虽然从理论上讲都可以使用，但与实际应用之间还是有一定的差距。例如，因为计算工具的有效数字位数是有限的，在运算过程中不可避免地会出现误差，因此，利用理论上精确成立的求根公式计算得到的结果也不一定可靠；韦达定理虽然指出了一元二次方程两个实根之间的关系，但没有指明如何实现的具体步骤；迭代法一般一次只能求一个实根，并且还存在收敛性的问题。因此，在实际解决问题时，必须要对各种方法进行分析比较，不能随便使用某种方法。

1.1.2 结构特性的设计

结构特性设计得好坏，直接影响到程序设计的效率，乃至程序执行的效率。结构特性的设计主要包括控制结构和数据结构的设计。

1. 控制结构

一个程序的功能不仅取决于所选用的操作，而且还决定于操作之间的执行顺序，即程序的控制结构。程序的控制结构实际给出了程序的框架，决定了程序中各操作的执行顺序。在程序设计过程中，通常用流程图形象地表示程序的控制结构。常用的流程图有两种，传统流程图与 N-S 结构化流程图。

1966 年，Bohm 和 Jacopini 证明了任何复杂的程序都可以用顺序、选择、循环三种基本控制结构组合而成。

2. 数据结构

在各种计算机应用中，数据处理所占的比重越来越大。在实际应用中，需要处理的数据元素一般有很多，而且，各元素之间不仅具有逻辑上的关系，还具有在计算机中实际存储位置上的关系。显然，杂乱无章的数据是不便于处理的，而将大量数据随意地存放在计算机中，实际上也是“自找苦吃”，对处理更是不利。

概略地说，数据结构是指互相有关联的数据元素的集合。数据结构又分为数据的逻辑结构和数据的存储结构。

数据的逻辑结构是指数据元素之间抽象化的相互关系，即数据的逻辑关系。

数据的存储结构是数据的逻辑结构在计算机中的存储方式。

在对数据进行处理时，数据的不同组织形式，其处理的效率是不同的。下面的例子说明了对数据的不同组织形式对处理效率的影响。

设有一学生情况登记表如表 1.1 所示。在表 1.1 中，每个学生的情况是以学号为顺序排列的。显然，要在表 1.1 中查找给定学号的某学生的情况是很方便的。但是，如果要在表1.1中查找成绩在 90 分以上的所有学生的情况，则需要从头到尾扫描全表，才能不会漏掉某一个成绩在90 分以上的学生情况。在这种情况下，成绩在 90 分以下的所有学生也都要被扫描到。由此可以看出，要在表 1.1 中查找成绩在某个分数段中的学生情况，其效率是很低的，尤其是当表很大时更为突出。

为了便于查找成绩在某个分数段中的学生情况，可以将表 1.1 中所登记的学生情况进行重新组织。例如，将成绩在 90 分以上、80～89 分、70～79 分、60～69 分之间的学生情况分别登记在四个独立的表中，如表 1.2～1.5 所示。现在如果要查找成绩在 90 分以上的所有学生情况，则可以直接在表 1.2(即子表 L_1)中进行查找，避免了对成绩在 90 分以下的学生情况进行扫描，从而提高了查找效率。

表 1.1　学生情况登记表

学号	姓　名	性别	年龄	成绩
80156	张小明	男	20	86
80157	李小青	女	19	83
80158	赵　凯	男	19	70
80159	李启明	男	21	91
80160	刘　华	女	18	78
80161	曾小波	女	19	90
80162	张　军	男	18	80
80163	王　伟	男	20	65
80164	胡　涛	男	19	95
80165	周　敏	女	20	87
80166	杨雪辉	男	22	89
80167	吕永华	男	18	61
80168	梅　玲	女	17	93
80169	刘　健	男	20	75

表 1.2　子表 L_1

学号	姓名	性别	年龄	成绩
80159	李启明	男	21	91
80161	曾小波	女	19	90
80164	胡　涛	男	19	95
80168	梅　玲	女	17	93

表 1.3　子表 L_2

学号	姓名	性别	年龄	成绩
80156	张小明	男	20	86
80157	李小青	女	19	83
80162	张　军	男	18	80
80165	周　敏	女	20	87
80166	杨雪辉	男	22	89

表 1.4　子表 L_3

学号	姓名	性别	年龄	成绩
80158	赵　凯	男	19	70
80160	刘　华	女	18	78
80169	刘　健	男	20	75

表 1.5　子表 L_4

学号	姓名	性别	年龄	成绩
80163	王　伟	男	20	65
80167	吕永华	男	18	61

此为试读，需要完整PDF请访问：www.ertongbook.com

由这个例子可以看出，在对数据进行处理时，可以根据所需要作的运算不同，而将数据组织成不同的形式，以便提高数据处理的效率。

1.1.3 算法的设计

算法是指为在有限步内解决一个具体问题而规定的意义明确的解题步骤的有限集合。概括地说，算法是指解题方案的准确而完整的描述。从程序来说，也可以说算法是一个有限条指令的集合，这些指令确定了解决某一特定类型问题的运算序列。

算法不同于一般的计算公式。作为算法应具有以下一些基本特征。

1. 算法的特征

作为一个算法，应具有以下四个特征。

(1) 可行性

算法的可行性包括两个方面：一是算法中的每一个步骤必须是能实现的。例如，在算法中，不允许出现分母为零的情况，在实数范围内不能求一个负数的平方根等；二是算法执行的结果要能达到预期的目的。

针对实际问题设计的算法，人们总是希望能得到满意的结果。算法总是与特定的计算工具有关。例如，某计算工具具有七位有效数字，在计算下列三个量

$$A=10^{12},B=1,C=-10^{12}$$

的和时，如果采用不同的运算顺序，就会得到不同的结果，即

$$A+B+C=10^{12}+1+(-10^{12})=0$$

$$A+C+B=10^{12}+(-10^{12})+1=1$$

而在数学上，A+B+C 与 A+C+B 是完全等价的。因此，算法与计算公式是有差别的。在设计一个算法时，必须考虑它的可行性，否则是不会得到满意结果的。

(2) 确定性

算法的确定性是指算法中的每一个步骤都必须有明确的定义，不允许有模棱两可的解释，也不允许有多义性。这一性质也反映了算法与数学公式的明显差异。在解决实际问题时，可能会出现这样的情况：针对某种特殊问题，数学公式是正确的，但是按此数学公式设计的计算过程可能会使计算机无所适从。这是因为根据数学公式设计的计算过程只考虑了正常使用的情况，而当出现异常情况时，此计算过程就不能适应了。例如，某计算工具规定：大于 100 的数认为是比 1 大很多，而小于 10 的数不能认为是比 1 大很多；且在正常情况下出现的数或是大于 100，或是小于 10。但指令“输入一个 X，若 X 比 1 大很多，则输出数字 1，否则输出数字 0”是不确定的。这是因为，在正常的输入情况下，这一计算可以得到正确的结果，但在异常情况下(输入的 X 在 10 与 100 之间)，其输出结果就不确定了。

(3) 有穷性

算法的有穷性是指算法必须能在有限的时间内做完，即算法必须能在执行有限个步骤之后终止。数学中的无穷级数，在实际计算时只能取有限项，即计算无穷级数数值的过程只能是有穷的。因此，一个数的无穷级数表示只是一个计算公式，而根据精度要求确定的计算过程才是有穷的算法。

算法的有穷性还应包括合理的执行时间的含义。因为如果一个算法需要执行千万年，显然失去了实用价值。例如，克莱姆(Cramer)规则是求解线性代数方程组的一个数学方法，但不能以此设计为算法。这是因为，虽然总可以根据克莱姆规则设计出一个计算过程用于计算

所有可能出现的行列式，但此计算过程所需的时间实际上是不能容忍的。

(4) 有足够的原始数据

一个算法是否有效，还取决于为算法所提供的数据是否足够。例如，对于指令“如果小明是学生，则输出字母 Y，否则输出字母 N”。当算法执行过程中提供了小明不是学生的情报时，执行的结果将输出字母 N；当提供的情报中只有部分学生的名单，且小明恰在其中，执行的结果将输出字母 Y。但如果在提供的部分学生的名单中，找不到小明的名字，则在执行算法过程中无法服从此指令，因为在部分学生的名单中找不到小明的名字，既不能说明小明是学生，也不能说明小明不是学生。

通常，算法中的各种运算总是要施加到各个运算对象上，而这些运算对象又可能具有某种初始状态，这是算法执行的起点或是依据。因此，一个算法执行的结果总是与输入的初始数据有关，不同的输入将会有不同的结果输出，当输入不够或输入错误时，算法本身就无法执行或执行有错。一般来说，当算法拥有足够的原始数据时，此算法才是有效的，而提供的数据不够时，算法并不是有效的。

综上所述，所谓算法是一个过程，这一过程有一组严谨地定义运算顺序的规则，并且每一个规则都是有效的且是明确的，此顺序将在有限的次数下终止。

1.1.4 流程的描述

程序设计的过程，实际上就是确定解决问题的详细步骤，而这个步骤通常称为流程。在程序设计过程中，根据不同的设计阶段以及具体要求，可以使用不同的描述流程的工具。常用的流程描述工具有以下几种。

1. 自然语言

自然语言是人们在日常生活、工作、学习中通用的语言，一般不需专门的学习和训练就能理解用这种语言所表达的意思。但用自然语言描述程序(或算法)的流程时，一般要求直接而简练，尽量减少语言上的修饰。例如，为了描述计算并输出 z=y/x 的流程，可以用自然语言描述如下：

(1) 输入 x,y。

(2) 判断 x 是否为 0：

若 x=0，则输出错误信息；

否则计算 y/x→z，且输出 z。

2. 算法描述语言

为了说明程序的流程，还可以用用户专门规定的某种语言来描述，这种语言通常称为算法描述语言。算法描述语言一般介于自然语言与程序设计语言之间，它具有自然语言灵活的特点，同时又接近于程序设计语言的描述。但必须指出，用算法描述语言所描述的流程，一般不能直接作为程序来使用，最后还需转换成用某种程序设计语言所描述的程序。算法描述语言与程序设计语言最大的区别就在于，算法描述语言比较自由，不像程序设计语言那样受语法的约束，只要描述得人们能理解就行，而不必考虑计算机处理时所要遵循的规定或其它一些细节。

在程序设计过程中，一般不可能开始就用某种程序设计语言编制计算机程序，而是先用某种简单、直观、灵活的描述工具来描述处理问题的流程。当方案确定以后，再将这样的流程

转换成计算机程序，这种转换往往是机械的，已经不涉及功能的重新设计或控制流程的变化，而只需考虑程序设计语言所规定的语法要求以及一些细节问题。

例如，对于计算 z=y/x 这个例子，其处理的流程可以描述如下：

```
INPUT x,y
IF (x=0) THEN
  OUTPUT "ERROR"
ELSE
  { z=y/x
    OUTPUT z
  }
```

对于这样的描述，只要懂一些英语，是不难理解的。有时为了更简练，在算法描述语言中还可以用一些自然语言。例如，上面的描述可以改为

```
输入 x,y
IF (x=0) THEN
  输出错误信息
ELSE
  { z=y/x
    输出 z
}
```

3. 流程图

人们在程序设计的实践过程中，还总结出一套用图形来描述问题的处理过程，使流程更直观，易被一般人所接受。用图形描述处理流程的工具称为流程图。目前用得比较普遍的是 N-S 图，有时就称为结构化流程图。本书及其多数教材中都使用结构化流程图，在本书的 1.2.1 节中将介绍这种流程图。为了便于比较，我们先给出用结构化流程图描述计算 z=y/x 的流程，如图 1.1 所示。

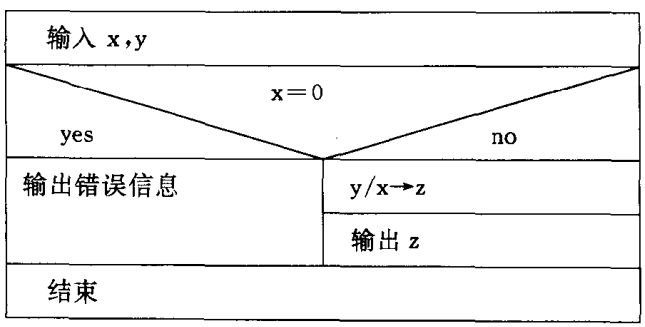

图 1.1

4. 编程

用某种程序设计语言编写的程序本质上也是问题处理方案的描述，并且是最终的描述。但在一般的程序设计过程中，不提倡一开始就编写程序，特别是对于大型的程序。程序是程序设计的最终产品，需要经过每一步的细致加工才能得到，如果企图一开始就编写出程序，往往会适得其反，达不到预想的结果。

下面是用C语言编写的计算z=y/x的程序：

```
#include "stdio.h"
main()
{ float x,y,z;
  printf("input x,y:");
  scanf("%f,%f",&x,&y);
  if (x=0) printf("error! x=0\n");
  else
    { z=y/x; printf("z=%f\n",z); }
  }
```

1.1.5 调试与运行

最后编写出的程序还需要进行测试和调试，只有经过调试后的程序才能正式运行。

所谓测试，是指通过一些典型例子，尽可能多地发现程序中的错误。因此，测试的目的是为了发现程序中的错误，而不是为了证明程序正确。

所谓调试，是指找出程序中错误的具体位置，并改正错误。

由此可以看出，测试与调试往往是交替进行的，通过测试发现程序中的错误，通过调试进一步找出错误的位置并改正错误。这个过程需要重复多次。关于程序调试的问题，将在1.5节中进行介绍。

1.2 程序设计的基本方法

1.2.1 结构化设计

结构化程序设计要求把程序的结构限制为顺序、选择和循环三种基本结构，以便提高程序的可读性。这种结构化程序具有以下两个特点：

(1) 以控制结构为单位，只有一个入口和一个出口，各单位之间的接口比较简单，每个单位也容易被人们所理解。

(2) 缩小了程序的静态结构与动态执行之间的差异，使人们能方便、正确地理解程序的功能。

下面简要介绍三种基本控制结构的形式。

1. 顺序结构

顺序结构的流程图如图1.2所示。在图1.2中，块S_1、S_2、S_3是按顺序执行的。

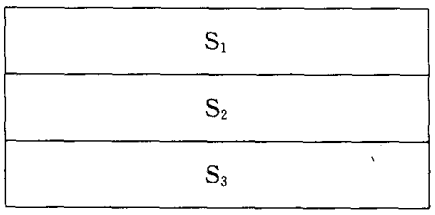

图 1.2

2. 选择结构

选择结构分为两路分支选择结构和多路分支选择结构，其中多路分支选择结构又称为分情形选择结构。

(1) 两路分支结构

两路分支选择结构的流程图如图 1.3 所示。

(2) 多路分支结构

多路分支选择结构的流程图如图 1.4 所示。

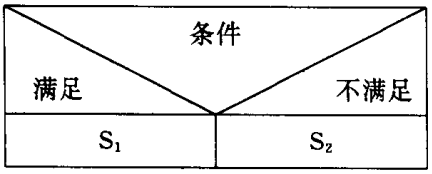

图 1.3

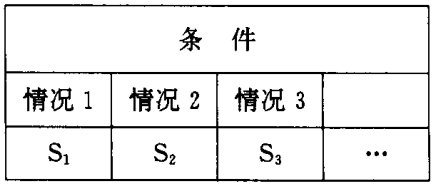

图 1.4

3. 循环结构

循环结构分为当型循环结构和直到型循环结构。

(1) 当型循环

当型循环结构的流程图如图 1.5 所示。当条件成立时，就执行模块 S，否则退出循环，执行该循环结构后面的程序。显然，在当型循环结构中，模块 S(称为循环体) 有可能一次也不执行(即一开始条件就不成立)。特别要指出的是，在循环体 S 中，应该要有改变条件的成分，否则将会造成死循环。

(2) 直到型循环

直到型循环结构的流程图如图 1.6 所示。由图 1.6 可以看出，直到型循环与当型循环的不同之处是，在直到型循环中，首先执行循环体 S，然后判断条件，若条件不成立，则继续执行循环体，这个过程直到条件成立为止，此时退出循环，执行该循环结构后面的程序。显然，在直到型循环中，循环体至少要执行一次。与当型循环一样，在循环体 S 中，应该要有改变条件的成分，否则将会造成死循环。

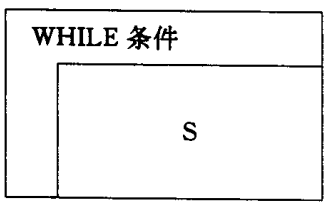

图 1.5

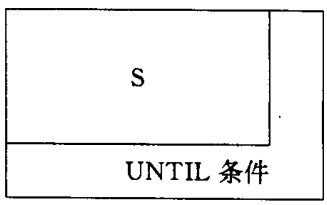

图 1.6

在以上三种基本结构中，每一个模块 S 或 S_1、S_2、S_3 等都又可以是这三种基本结构之一。图 1.7 是顺序输出 2～100 之间所有素数的流程图，在这个流程图中，表示了三种基本结构互相嵌套的情况。

1.2.2 模块化设计

模块化设计是指把一个大程序按人们能理解的大小规模进行分解。由于经过分解后的各模块比较小，因此容易实现，也容易调试。

在进行模块化程序设计时，应重点考虑以下两个问题：

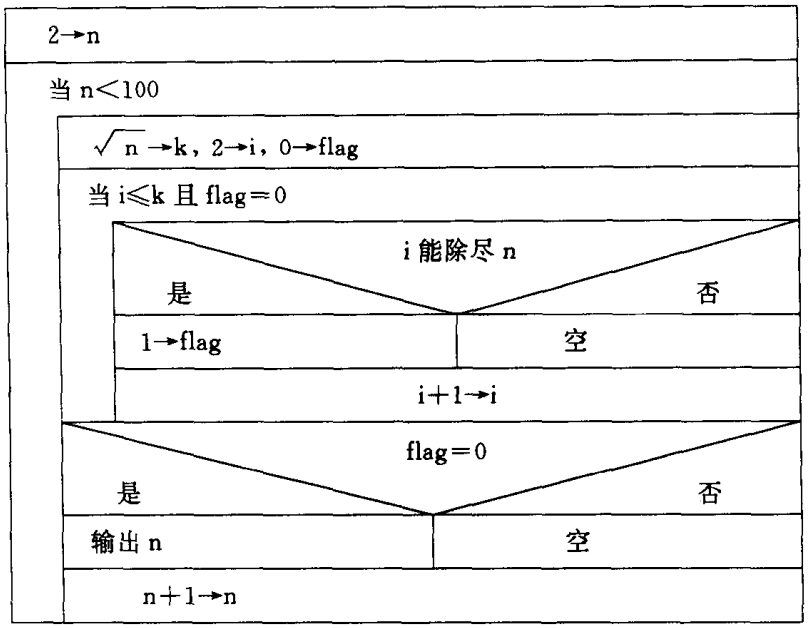

图 1.7

按什么原则划分模块？

如何组织好各模块之间的联系？

(1) 按功能划分模块

划分模块的基本原则是使每个模块都易于理解。而按照人类思维的特点，按功能来划分模块最为自然。按功能划分模块时，要求各模块的功能尽量单一，各模块之间的联系尽量少。这样的模块其可读性和可理解性都比较好；各模块间的接口关系比较简单；当要修改某一功能时只涉及一个模块；其它应用程序可以充分利用已有的一些模块。

(2) 按层次组织模块

在按层次组织模块时，一般上层模块只指出“做什么”，只有在最底层的模块才精确地描述“怎么做”。例如，在图 1.8 所示的层次结构中，主模块只需要指出总任务就可以了，而模块 1、模块 2 与模块 3 分别指出各自的子任务，模块 4、模块 5 与模块 6 才精确描述“怎么做”。

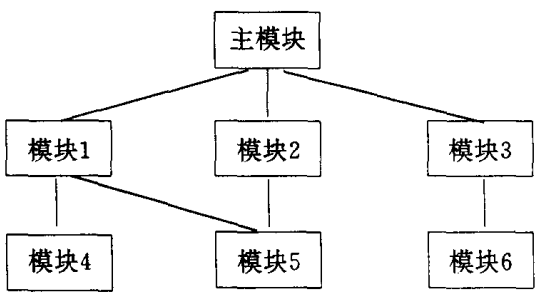

图 1.8

1.2.3 自顶向下、逐步细化的设计过程

自顶向下、逐步细化的设计过程，包括以下两个方面：

此为试读，需要完整PDF请访问：www.ertongbook.com