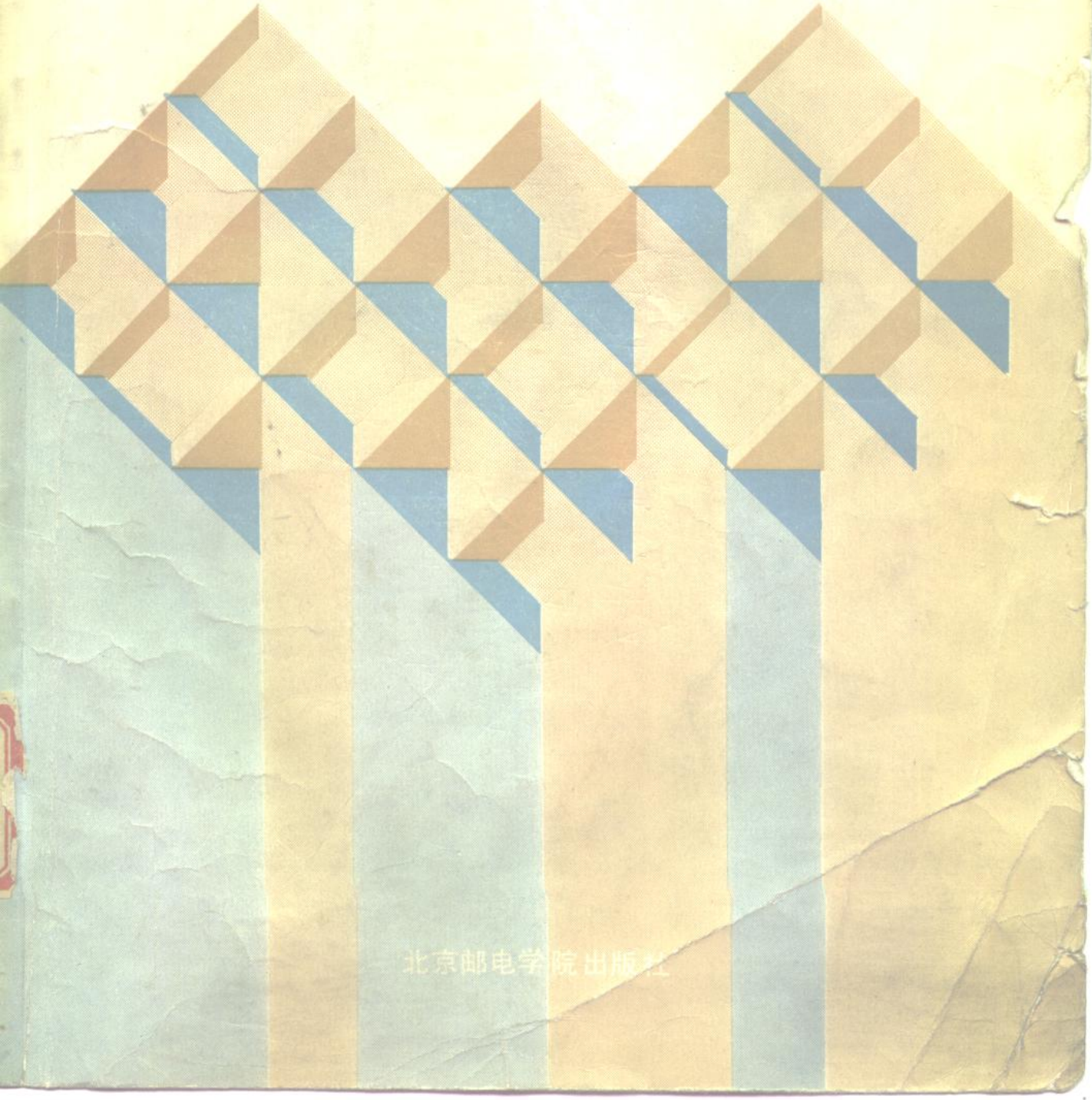
数据结构及应用

朱祥华　顾懋栉　编著



北京邮电学院出版社

数据结构及应用

朱祥华　顾懋祺　编著

北京邮电学院出版社

内 容 简 介

本书系统地介绍了数据结构的主要技术及有关算法的设计和分析。内容包括：线性表、数组、链表、树、图等主要数据结构，排序和检索等操作以及文件的结构和组织等。

本书概念清楚，理论联系实际。对所述运算及操作的算法，大部分配有PASCAL语言和BASIC语言的程序并附有执行结果。最后一章还给出了一些应用程序。

本书可作为高等院校有关专业的教材，也可作有关教学人员和工程技术人员的参考书。

数据结构及应用

编　　著　朱祥华　顾懋纳

责任编辑　时友芬

•

北京邮电学院出版社出版

新华书店北京发行所发行　各地新华书店经售

北京密云华都印刷厂印刷

•

787×1092毫米　1/16　印张17.25　字数427.5千字

1989年6月第一版　1989年6月第一次印刷

印数：1—3000册

ISBN 7-5635-0025-1/TP·6　定价：2.90元

季

前　言

随着计算科学的迅猛发展，计算机的应用已深入到各行各业各个部门。近年来计算机在情报检索、信息管理、办公室自动化、图象识别以及日常事务的处理等方面得到了越来越广泛的应用。这不但对计算机专业人员提出了更高的要求，而且也要求非计算机专业人员掌握一定的计算机方面的基本知识，尤其是如何处理非数值型信息显得更为重要。据统计，目前非数值型信息的处理占据了计算机90％以上的机时。

“数据结构”课主要是研究数据（特别是非数值型数据）的组织、存贮以及运算方法的课程。它是“操作系统”、“编译原理”、“数据库”及“人工智能”等课程的基础。对计算机专业是一门专业基础课。对非计算机专业来说，通过学习，要求达到能对非数值型数据作一般的运算处理。

全书共分三部分：

第一部分介绍基本的数据结构。其中包括线性表、栈、队列、链表、树、图、文件等等。

第二部分介绍基本的程序操作。如排序、检索等。

第三部分为应用举例。

为适应非计算机专业读者的水平和需要，书中侧重讲清概念和算法。每种算法基本上都配有实现的程序。此外还在运算时间和存贮空间方面对算法作定性的分析。所有程序用PASCAL语言及BASIC语言(扩展)实现，并全部在IBM-PC机或其兼容机上通过。凡学过一种高级语言，具有一定程序设计基础的读者都可以阅读本书。

本书可作为高等院校本科非计算机专业或大专计算机专业的试用教材。也可作为具有大专水平的计算机培训班的教材，讲授学时为36～51学时。若取下限则目录中画＊号的内容可以不讲。通过学习学会分析数据对象的特点，选择适当的数据结构，能够确定正确清晰的算法并能写出程序加以实现。

由于时间仓促以及笔者的水平所限，书中如有错误之处，敬请读者批评指正。

编　者

1989年3月

目　　录

第一章 绪 论

§1.1 为什么要学习数据结构

随着科学技术的飞速发展，计算机的广泛应用已成为我们这个时代的标志。目前我国工业、农业、国防及科学技术、文化教育等等各个领域内，正在普遍推广和普及计算机的使用。它的应用也早已从仅限于科学计算发展到越来越广泛的数据处理和实时控制。计算机几乎影响着现代社会生活的每一个角落。不管人们是否意识到，计算机时代正以凌厉之势向我们迎面而来。如果说读书、写字是我们第一文化的话，那么计算机的广泛使用将要成为我们的第二文化。

为什么当今的社会要把计算机的推广应用提到这样的高度呢？我们知道，人类活动的整个历史离不开对信息和数据的收集、保存、处理和利用。起初人们利用绘画语言，结绳等传递和记录信息。后来以纸张作介质编制各式各样的表目、帐本、册子、字典来收集记录信息，并保存和处理它们。电子工业发展后，人们借助于磁介质来收集、保存信息并进行加工。60年代以后由于社会生产力的高速发展，新技术不断涌现，信息量急剧膨胀，迫使人们对信息的处理和数据的利用不得不实行自动化、网络化和社会化。也就是整个人类社会进入了信息化的社会。例如，工业生产上的自动流水线、科学研究、资料情报的检索、国民经济的计划和预决算、银行帐目、人口调查的统计分析、交通运输、档案管理、办公室自动化等等都需对大量数据进行处理加工并要求快速及时得出结果。如果仅仅依靠人工去翻阅资料文件是难以完成的，必须借助于电子计算机的高速度和大容量来解决。

计算机所处理的信息决不是杂乱无章的数据堆积。而是存在着内部联系的，只有分析清楚它们之间的内在联系并采用适当的结构组织，才能对大量数据进行行之有效的处理。因此单纯依靠程序设计员的经验和技巧已不能编出高效率的处理程序，必须对程序设计方法进行系统地研究。这不仅涉及到程序的结构和算法，也包含了研究程序所加工的对象——数据的结构。这里所说的算法是指精确定义的一系列的规则。指出怎样从输入信息，经过有限个步骤后得到所要求的输出结果。算法的具体实现即为程序。应该指出算法和数据结构之间存在着本质的联系，因为我们研究某种类型的数据结构时总是离不开要讨论加于这种数据结构的运算（如插入，删除等），只有通过某种算法实现这些运算，数据结构才能体现出它的意义和作用。反之，当我们研究某种算法时也离不开要考虑该算法处理对象的数据所采用的结构。所以，“数据结构”和“算法”是研究计算机科学的基本课题。

本课程主要是解决怎样合理地组织数据，建立合适的数据结构，如何提高计算机的效率。也就是说，要研究怎样选择有效的算法以提高计算机执行速度和节省存贮空间。随着数据库和人工智能的发展。所处理的数据越来越复杂，量也越来越大，促使人们更加重视数据结构的研究。

数据结构在计算机科学中是一门综合性的专业基础课。数据结构的研究不但要涉及到

计算机硬件的范围如编码理论、存贮器和存取方法；而且和计算机软件的研究有更加密切的关系，是计算机的操作系统和编译原理的基础。信息检索中也要研究如何组织数据的问题。近年来数据库和人工智能的发展大大促进了数据结构的研究。

本课程的理论基础是程序设计和离散数学。在分析和处理数据结构和算法时也用到概率、图论和集合等方面知识。但对非计算机专业来说数据结构主要还侧重于实践技术和应用，而不是理论上的深入探讨。因此我们力求避开一些数学上的推导，尽量深入浅出地说清概念。在每章中都列举应用例子，并附有程序，以帮助读者理解和应用。

§1.2 什么是数据结构

要了解什么是数据结构，首先要说明什么是数据？数据是描述客观事物的数字、字符以及所有能输入到计算机中被程序处理的符号的集合。也就是说，数据就是计算机加工的信息。例如，对于一个简单的计算所用的数据是数字。如果计算机用于事务管理，则除用数字型数据外大量的还要用到字符数据。再如对于一个编译程序，它所使用的数据则是程序员编写的源程序。可见不同的信息可用不同数据类型。所谓数据类型是指在某种程序设计语言中变量所具有的数据种类。且该种语言对变量提供一组有意义的运算。一个变量在程序中只能定义一种数据类型。例如，在BASIC语言中数字型数据有整型、实型（又分单精度型和双精度型），对于这些类型的数据可进行加、减、乘、除、乘方、取模等算术运算，还可进行关系运算和逻辑运算。另还有字符型数据。对这种数据可进行字符串连接运算、关系运算和逻辑运算。在PASCAL语言中，除了具有上述BASIC语言的这些数据类型外，还有布尔型、枚举型、集合型、记录型、文件型、指针型等。但是不管是哪种高级语言，它们的数据类型不外乎分为两大类，即初等型和组合型。只包含一个数据项的数据属于初等型，否则是组合型。

初等型的数据可分五类：

整数型——取值为计算机可以表示的范围内的整数。

实数型——取值为计算机可以表示的范围内的实数。

布尔型——取值为“真”（true），或为“假”（false）。

字符型——取值为计算机可以表示的字符集的元素。

指针型——取值为计算机存贮器中的一个地址或相对地址。这种类型的数据在BASIC，FORTRAN等语言中是没有的。

组合型数据是由初等类型用某种方式组合而成。组合方式不同就形成不同的类型，如数组型、记录型、文件型等等。

需要说明的是，不一定每种程序设计语言都具备上述各种数据类型。

这里我们介绍两个常用的术语：数据元素，即数据的基本单位。数据对象是性质相同的数据元素的集合。例如，一组整数的集合I={2,4,6,8,…}。一周七天的集合

date={Sunday,Monday,…,Saturday}。

I及date都是数据对象，而它们中单个数据项2，4，…或Sunday,Monday,…都是数据元素。数据对象可以是有限的，也可以是无限的。

通常数据对象中的元素不是孤立的而是彼此之间存在着关系，这种数据元素彼此之间

的关系称为数据的结构。在计算机程序处理这些数据时，首先应该很好地研究数据元素之间的相互关系，选用一个合适的结构将这些数据组织起来，以便在对这些数据进行运算时提高计算机的效率。

例如：职工情况表

姓　名	性　别	年　龄	政治面目	籍　贯	职　称	单　位	工　资
王　平	女	45	群　众	江　苏	讲　师	无线系	120
李　力	男	30	党　员	山　西	助　教	无线系	98
张红英	女	25	团　员	北　京	实验员	有线系	75.50
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

表中每行的信息说明了一个职工简单的人事情况。是职工情况表的基本单位，也就是该表的数据元素，也称为结点或记录。这里的数据元素是若干个表征事物属性的数据项组成。它们都是有独立含意的最小标识单位。由表可见除了表中第一个结点为表头，最后一个结点为表尾外，其它每个结点的前面都只有一个结点，在它们的后面也都只有一个结点，所以各结点之间的关系是线性的，我们称这种结构为线性表。这里，我们仅考虑了数据元素之间的逻辑关系，叫做数据的逻辑结构。

现在我们再进一步研究一下上述线性表中各结点的排列次序问题，如果表中结点无一定的规律排列，则要查看一个职工情况时必须逐个地查看直到找到为止（或找完该线性表还找不到则表示无此人为止）。这是相当花费时间的。如果我们将表中所有结点姓名按字典顺序排列则查找起来就方便得多。

假若一个单位的职工人数相当多，我们还可以采用分级管理的办法。如图 1.1 所示。图中的这种数据结构是按层次的分支结构，形同一棵倒立的树，学院是树根，故称作树结

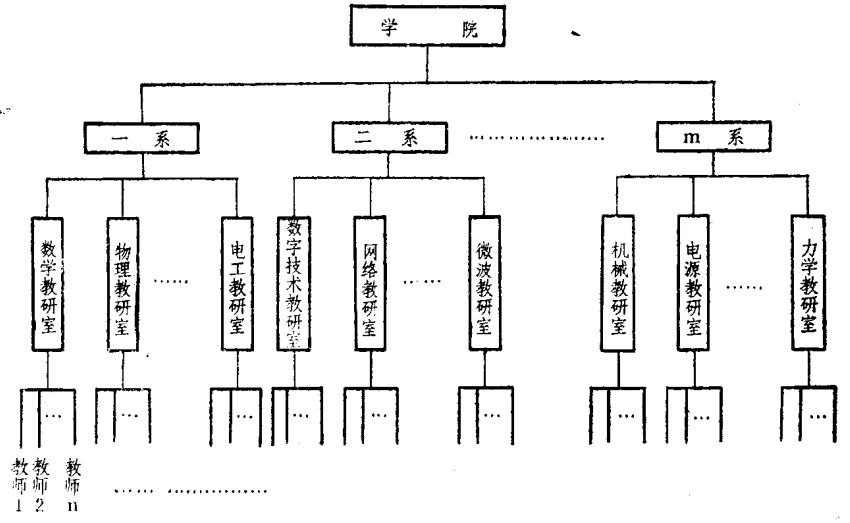


图 1.1　职工情况数据的分级管理示意图

构。如果要查询一个教师的情况只要知道该人的所在系和教研室即可很快找出。

数据的逻辑结构有各种形式，大体上分线性和非线性两部分。线性结构有表、堆栈、队列、链表等；非线性结构有树、图等。无论是哪一种结构都要考虑到数据在增加、减少或变动情况下如何处理，以及对已定的数据结构进行查询统计等操作时又怎样办等这样一些问题。例如，就职工情况表而言要考虑到当某一职工调出本单位后如何将有关他情况的数据从原结构中删除，当有人调入本单位时又如何将此人的数据插入到已有的数据结构中去；或者要查看某一个人的情况，或要统计一下本单位的党员人数等等。这就要求相应的数据结构定义各种满足要求的运算，并给出恰当的实现这些运算的算法。

几种常用的运算有：

检索：在数据结构中查找满足一定条件的结点；

插入：往数据结构里增加新的结点；

删除：把指定的结点从数据结构中去掉；

更新：修改指定结点的一个或几个数据项值。

排序：把数据结构中的结点序列按某一数据项值的大小顺序排列。

对数据进行运算首先要将数据的逻辑结构存贮在计算机中。数据在计算机中的存贮方式叫做数据的物理结构。计算机的存贮器是由有限多个存贮单元组成，每个存贮单元有唯一的地址。由于计算机的存贮区域有限，在存贮数据时应考虑到有效地利用存贮空间问题。数据在计算机中有四种基本的存贮方式：

1．顺序方式：把逻辑上相邻的结点存贮在物理上相邻的存贮单元。此法多用于线性的存贮结构。

2．链接方式：给每个结点附加一个指针域，存放它后继结点的地址（即指针）。由指针把各数据元素链接起来，故指针也就是链。后继结点可以是一个也可以是多个，也就是说链可以是单链，也可以是多链，随数据的逻辑结构而定。例如用链式存贮上述线性表则是单链的，如果存贮树结构则需多链。在链式存贮中由于有了指针，所以物理结构不一定与该数据的逻辑结构相同。

3．索引方式：以结点在线性结构中的序号（位置）建立索引表，以此来确定结点的存贮地址。

4．散列方式：根据结点的值进行某些函数计算后得到相应的存贮地址。

以上四种方法可以组合使用。同一个数据的逻辑结构可采用不同的存贮方式，这要根据运算的方便来作出选择。

综上所述，数据结构研究的主要范围包括三个方面：即研究数据的逻辑结构、物理结构以及适合这些结构的各种运算方法；研究各种结构在计算机科学及软件工程中的应用以及分析各种算法的效率。

§1.3 使用算法语言的说明

研究数据结构的目的是为了更好地进行程序设计，对于每一种算法必须运用一种适当的语言加以实现。

本书为了适应不同层次读者的需要，采用了 PASCAL 和 BASIC 二种语言同时并

举的方针。PASCAL 语言是一种结构式的语言，它有助于实现程序结构的模块化，在处理非数值型数据方面有较强的功能；它的数据类型丰富，特别是指针型变量在处理非线性的数据结构（如树和图等）及数据的链式存贮都很方便。但由于目前在各类高校的非计算机专业中 PASCAL 语言尚未普及，为了读者阅读的方便，我们对同一种算法一般又配了 BASIC 语言的程序。编者认为，BASIC 语言虽然在程序结构方面不如 PASCAL 语言的程序严谨，但它不但易学易懂，且扩展的 BASIC 在处理非数值性数据，特别是对字符串及文件的处理上具有很强的功能。因此，在不同的章节中我们对这二种语言的应用也有所侧重。还有一些运算，我们用文字比较详细地描述了它们的算法，读者可自行选择一种语言编制程序。不管怎样，书中的基本理论和概念具有一般性，可独立于任何语言。

§1.4 程序设计质量标准和方法

前面已经提到研究数据结构的目的主要是为了要很好地组织数据，寻求合理的存贮结构和运算方法。但所有这些最终还是要通过程序来实现的。如果我们不能熟练地掌握程序设计的方法和技巧，即使主观设想的数据结构和算法再好，仍然是不能达到目的的。

关于程序设计的基本方法，一般读者在学习算法语言时已经接触过。这里不作冗赘的叙述，只是帮助读者作一些复习，并介绍一种简单的、非规范化的程序设计方法。

1.4.1 程序设计的质量标准

程序设计就是根据题目给定的任务，进行抽象的思维，从而研究出编制程序的方法和技巧。它为开发计算机程序这一复杂而艰难的过程指明方向。保证设计出的程序具有合理的高标准。编制程序不能粗略地想到哪里写到哪里，这样写出的程序不但凌乱不易看懂甚至容易出错。在写程序前必须要仔细安排，选择合适的设计方法和结构。检查一个程序的好坏主要依靠下列几点：

一、正确性

如果一个程序不能正确地工作，则就失去了它的价值。对一个程序来说仅仅是提供正确输入时得到正确的输出是不够的。还必须保证在使用时遇到所有可能情况下都能正常工作。这在实际中是较难满足的条件。必须通过精心仔细的设计，详尽的检查，在各种设想的可能条件下反复运行，发现问题即时修改。例如，当我们用计算机实现人事档案管理时，首先要把职工的姓名、性别、出生年月等等情况存入计算机，请看下面的语句。

```
⋮
50  input "Sex(M/F):",S$
70  input "Birth(mm/dd/yy):",B$
⋮
```

上面语句，双引号内为提示语，圆括弧内是提示用以输入的格式。50 句是提示用户输入职工性别，M 代表男，F 代表女，请用户在 M 和 F 之间选择一个输入。70 句是提示用户输入出生年月，并按月/日/年格式输入。有了这样的提示语后，虽然增加了可靠性，但如果用户不按要求格式输入，程序照样能够通过，因此在查询或统计时就会产生错误。为了程序可靠，在编制程序时必须加入一些检测语句，如上面程序可改为：

```
⋮
50  input "Sex(M/F):",s$
60  if  s$<>"M" and s$<>"F" then print "Wrong enter!! Please
    try again!":goto 50
70  input" Birth(mm/dd/yy):",b$
80  if len(b$)<>8 or val(left$(b$,2))>12 or val(mid$(b$,4,2))>
    31 or mid$(b$,3,1)+mid$(b$,6,1)<>"//" then print "Wrong!!
    Please try again!":goto 70
⋮
```

当增加了 60 句及 80 句以后，用户若不按要求格式输入，就会有错误信息出现。并要求重新输入，显然使程序的可靠程度增加了。

二、结构清晰

所谓计算机程序结构就是将程序各部分放在一起形成整个程序的方式。程序的结构必须适应于它所处理的数据结构，程序还要反映它所执行的操作逻辑顺序。

结构化的程序一般由三种基本的结构组成：

1．顺序结构：执行时由小行号到大行号语句顺序进行。一般多用于简单的数据处理、简单表达式或函数的运算。程序流程如图 1.2 所示。

2．分支结构：一般用于需根据某条件判断确定程序执行流向的情况。如图 1.3 所示。

3．循环结构：一般多用于需要多次执行某些运算的情况。如数组运算，迭代计算等等（见图1.4）。

一个好的程序应尽量避免用过多的 GOTO 语句。要用也最好在这些基本结构的内部使用，否则就破坏了程序的结构层次。

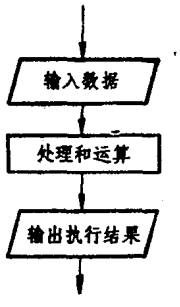
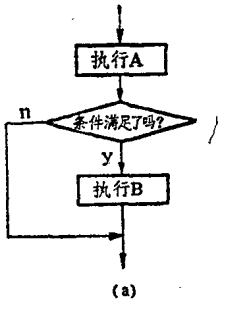


图 1.2　程序的顺序结构



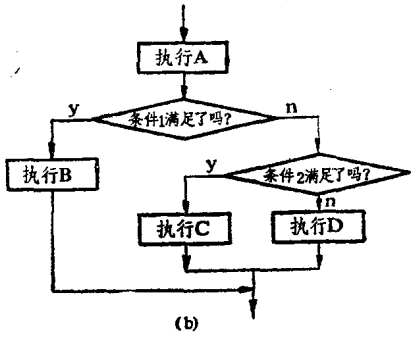


图1.3　程序的分支结构

对于大型的程序，最广泛应用的程序结构方式即是模块式的结构。一个模块是程序的一部分，它执行特定的操作。模块输入输出及其实现过程都要精确的定义。一个程序可以按其要完成的任务划分成若干过程，每个过程组成一个模块，把诸多的模块组合在一起则构成了模块式的程序结构。例如，如果我们要编制一个职工情况管理程序，要求程序具有输入数据、修改、删除、检索、统计等操作功能，则我们可将上述每个功能用一个模块完成，上层可用一个主控模块来选择功能。如图1.5 所示。

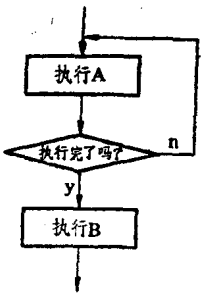


图1.4 程序的循环结构

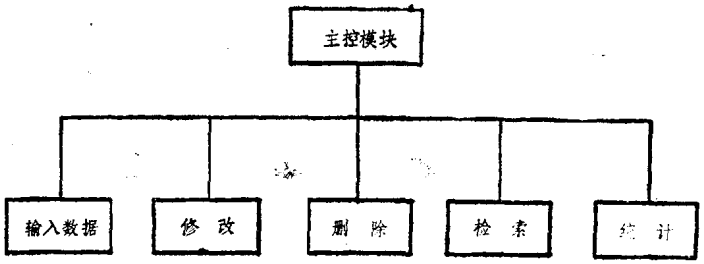


图1.5 模块式程序结构

从每个模块送出或由别的模块传送到当前模块的数据便形成该模块与其它模块的接口。各模块各自执行自己确定的任务，不会损害它所不处理的其它任何数据。因而便于修改和扩充，而且也便于采用覆盖技术，以解决程序大而内存不够的矛盾。

三、简单、易懂

这是程序设计中非常重要的要求。简单的程序容易了解、检查、修改和编制说明。将一个复杂的任务减化为简单程序的过程，要求有合适的结构，算法精炼，并要求有娴熟的技巧。

一个好的程序文本应该是容易看懂的，这可以便于程序的维修。这种易读性可以用仔细地选择变量名，清晰的指令安排和在程序中适当加注释达到。

1.4.2 一种简单的程序设计方法

在程序设计时如何能达到上述要求，有各种程序设计的技术，这里我们仅介绍一种简单的、非规范的设计方法即“逐步精炼”法(stepwise refinement)。此法使用还是很普遍的。

“逐步精炼”法设计程序的实质是连续地一步一步地将任务划分成较小的步骤，直到每一步能直接编制程序为止。先从任务的总体说明开始，再将任务划分成若干粗略的实现步骤，于是再把每一个粗略步骤又划分成几个较小的步骤。这样不断地精炼下去。每次精炼都对任务的各步作进一步说明。逐步精炼法的不足之处是一个任务可以用多种方式划分步骤，并不是每种划分方式都能最后编制出程序来。如果一种方式不行就必须试用另一种方式，或迟或早终究会找到合适的方式的。

下面我们说明逐步精炼法程序设计的过程：

例1 假定计算机内部没有平方根函数。试写一程序在规定精度下计算给定值的平方根。精度系指给定的数和程序计算的平方差值。设变量 N、A、S 分别为给定的数、精度和平方根值。

第一次精炼：

此任务可粗略地分为以下几步：

1. 输入给定的数和精度；
2. 检查该数是否为正数；
3. 计算平方根；

4．输出平方根值。

第二次精炼：

其中输入、检查、输出这几步显然可以直接编程了。而计算平方根的那一步如何求法尚需进一步精炼。从数学中我们知道用下式可进行平方根的估算：

设S为某数n的平方根的估计值。那么一个更为精确的估计值 S^+ 可由下式得到：

$$S^+ = (S + N/S)/2 \qquad (1\text{-}1)$$

若用（1-1）式，那么计算平方的步骤如下：

1．选定一个平方根的初始值；

2．重复以下过程：

用上述公式计算出一个更为精确的平方根估算值

直到该估计值达到足够的精度为止。

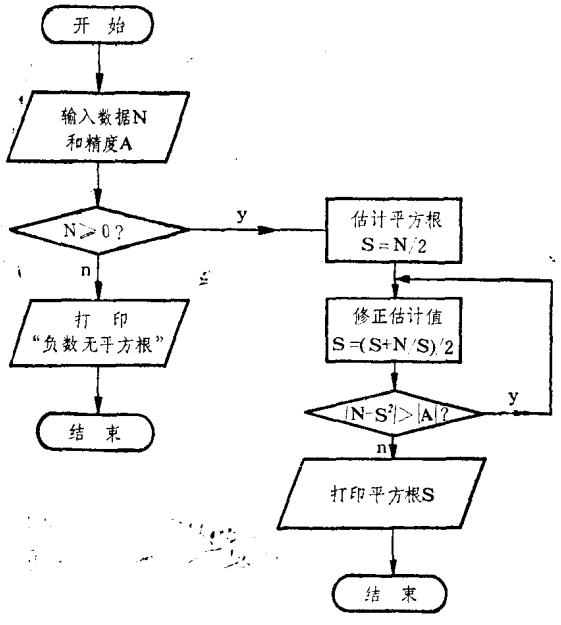


图1.6 逐步精炼法程序设计举例流程图

第三次精炼：

虽然上面给出的步骤已能编制出程序，但还需解决如何选择平方根初始值及确定检查精度的关系式的问题。首先选用一个平方根的初始值：

考虑到不至使迭代运算的次数太多，选用 $S = \frac{N}{2}$ 为初始估算值。

重复用（1-1）式进行迭代计算，每计算一次得到一个更为精确的估计值，并且用下面的关系式检查精度：

$$N - S^2 < A$$

直到满足此式时，停止运算，输出计算结果。

经过三次精炼后，即可画出程序流程图如图1.6所示。

用PASCAL语言编制的程序如下：

```
{ 程 序 1.1.1 }
PROGRAM    ROOT(INPUT,OUTPUT);
 CONST  A=1E-4;
 VAR  N,S:REAL;
 BEGIN
  WRITE('Please input number:');
  READ(N);  WRITELN;
  IF N<0 THEN WRITELN('Negative number, no root!')
         ELSE [ S:=N/2;
                WHILE ABS(N-S*S)>ABS(A) DO
                 S:=(S+N/S)/2;
                WRITELN ('Square root is:',S:6:2) ];
 END.
```

```
A>ROOT1
Please input number:121

Square root is: 11.00
A>ROOT1
Please input number:-121

Negative number, no root!
```

例2　模块结构程序设计举例

这个程序的目的是保持记录飞机场上现有飞机的最新清单。本程序必须响应以下命令：

INN ID　　将具有某标识符的飞机加到清单中；

OUT ID　　从清单中把某标识符的飞机消除。若未发现该标识符，则显示一个信息。

QUERY ID　搜索清单，看是否有某标识符的飞机。搜索结果由信息显示出来。

DISPLAY　显示机场上所有飞机标识符。

QUIT　　程序停止

其中ID代表飞机标识符。现规定飞机标识符组成是二个字母后跟下划线再跟三字符，如 PH_BUI，DM_FIY 等，命令IN PH_BUI 的意思即是要把标识符为 PH_BUI 的飞机加入到飞机清单中去。

程序设计：仍用逐步精炼方法：

第一次精炼：确定程序的总体结构

显然该程序将包含若干程序模块，每块完成上述一个功能。该结构可分为三层：最上层为控制模块，它识别每个命令。根据输入的命令，控制模块将控制转移至相应的操作模块。操作模块形成结构的中间层。结构的最下层是一组服务模块。它直接用来和存贮信息的数据结构相互作用。

第二次精炼：确定数据结构及各个模块的操作要点。

对本程序来说，采用字符串数组结构是合适的。由于在数组定义后其大小便固定，而该程序要存的数据数量又是变化的，所以必须选用适当的“空元素标记”。用字符串“* * * * * *”表示空元素标记较为适宜，因为它和飞机标识符长度相同，但又不会错当成真实的标识符。

各操作模块的要点如下：

INITIATE　使数组初始化，即将所有位置都装入空元素标记。

INN ID　　在数组中搜索空元素标记。若找到一个则将飞机标识符装入数组该位置，同时显示肯定信息。若数组中无空元素标记则显示存贮器已满信息。

OUT ID　　在数组中搜索给定的飞机标识符，若找到，则将该位置装入空元素标记，并示显示肯定信息。如该标识符找不到，则显示飞机不存在信息。

QUERY ID　在数组中搜索给定标识符。若找到则显示肯定信息，否则显示该飞机不存在信息。

DISPLAY　　显示全部数组元素

QUIT　　停止程序执行，此命令执行时无须搞一个独立模块，只是在控制块中识别该命令后用 END 指令结束执行即可。

由上述各模块的操作要点看出。在几个模块中都要进行对数组元素的搜索或要把元素装入数组的操作。为了避免程序对同一操作内容的多次重复，故我们把搜索和装入另立两个服务性的模块，供上述操作模块调用。它们是：

LOAD　在给定数组的下标位置装入元素。

SEARCH　在数组中搜索给定元素，并将其下标值返回到调用程序。如果要找的元素数组中没有，则返回的值为0。

至此，我们可以画出该程序的结构图，从中可以看到各模块之间的关系。如图 1.7 所示。

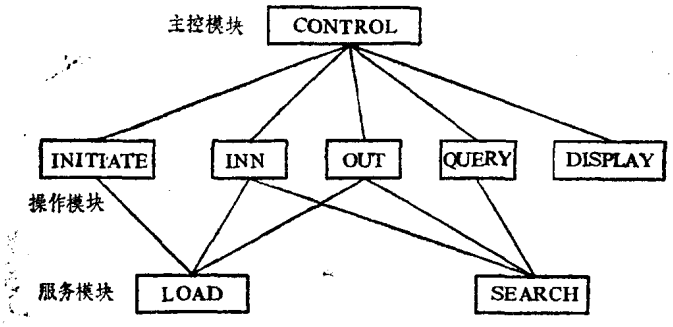


图 1.7 例2的程序结构图

主控模块是主程序，其它模块是子程序。

第三次精炼应该确定实现上述各个模块要点的具体算法。因为这些模块的算法都比较简单，读者可自行练习。下面将直接给出实现上述模块的 PASCAL 程序和 BASIC 程序。

程序变量：（变量定义后面括弧中为 BASIC 语言程序中所用的变量标识符）

A〔1:30〕　存放飞机清单的数组（A$（30））

C　命令字符串（C$）

CC　命令字符串的子串。

D　飞机标识符（D$）

N　子程序用飞机标识符（N$）

K　元素位置

E　命令长度

F　命令执行标志（F=1，命令已执行；F=0，命令未执行）

I, J　循环控制变量

PASCAL语言程序如下：

```
{ 程 序 1.2.1 }
PROGRAM    ID(INPUT,OUTPUT);
 TYPE   LST=LSTRING(12);
 VAR    A:ARRAY [1..30] OF LST;
```

```
      C,CC,N,D:LST;
      E,I,J,K:INTEGER;
PROCEDURE  LOAD(I:INTEGER;N:LST);
  { LOAD ELEMENT INTO ARRAY }
  BEGIN
   A[I]:=N
  END;
 PROCEDURE  SEARCH(N:LST; VAR K:INTEGER);
  { SEARCH FOR ELEMENT IN ARRAY }
  BEGIN
   K:=0;
   FOR J:=1 TO 30 DO
    IF A[J]=N THEN [ K:=J;  BREAK ]
   END;
 PROCEDURE  INITIATE;   {INITIAL ARRAY }
  BEGIN
   WRITELN('Aircraft identifier system');
   FOR I:=1 TO 30 DO
    [ D:='* * * * * *';  LOAD(I,D) ]
 END;
PROCEDURE  INN;
 { LOAD AIRCRAFT IDENTIFIER AT POSITION I }
 BEGIN
  N:='* * * * * *';  SEARCH(N,K);
  IF K=0 THEN WRITELN('Storage full,identifier  ',D, 'rejected!')
         ELSE [LOAD(K,D);
               WRITELN('Identifier has been loaded') ]
 END;
PROCEDURE  OUT;
 { REMOVE A IDENTIFIER }
 BEGIN
  SEARCH(D,K);
  IF K=0 THEN WRITELN('Identifier  ',D,' not present!')
         ELSE [N:='* * * * * *'; LOAD(K,N);
               WRITELN('Identifier  ',D,' remove') ]
 END;
PROCEDURE  QUERY;
 { SEARCH A IDENTIFIER AT POSITION K }
 BEGIN
  SEARCH(D,K);
  IF K=0 THEN WRITELN('Identifier  ',D,' not fount!')
         ELSE WRITELN('Identifier ',D,' present at position ',K:2)
 END;
PROCEDURE  DISPLAY;
 { DISPLAY ALL IDENTIFIERS }
 BEGIN
  FOR I:=1 TO 30 DO
   [ WRITE(A[I]:10);
   IF I MOD 5=0 THEN WRITELN ]
```