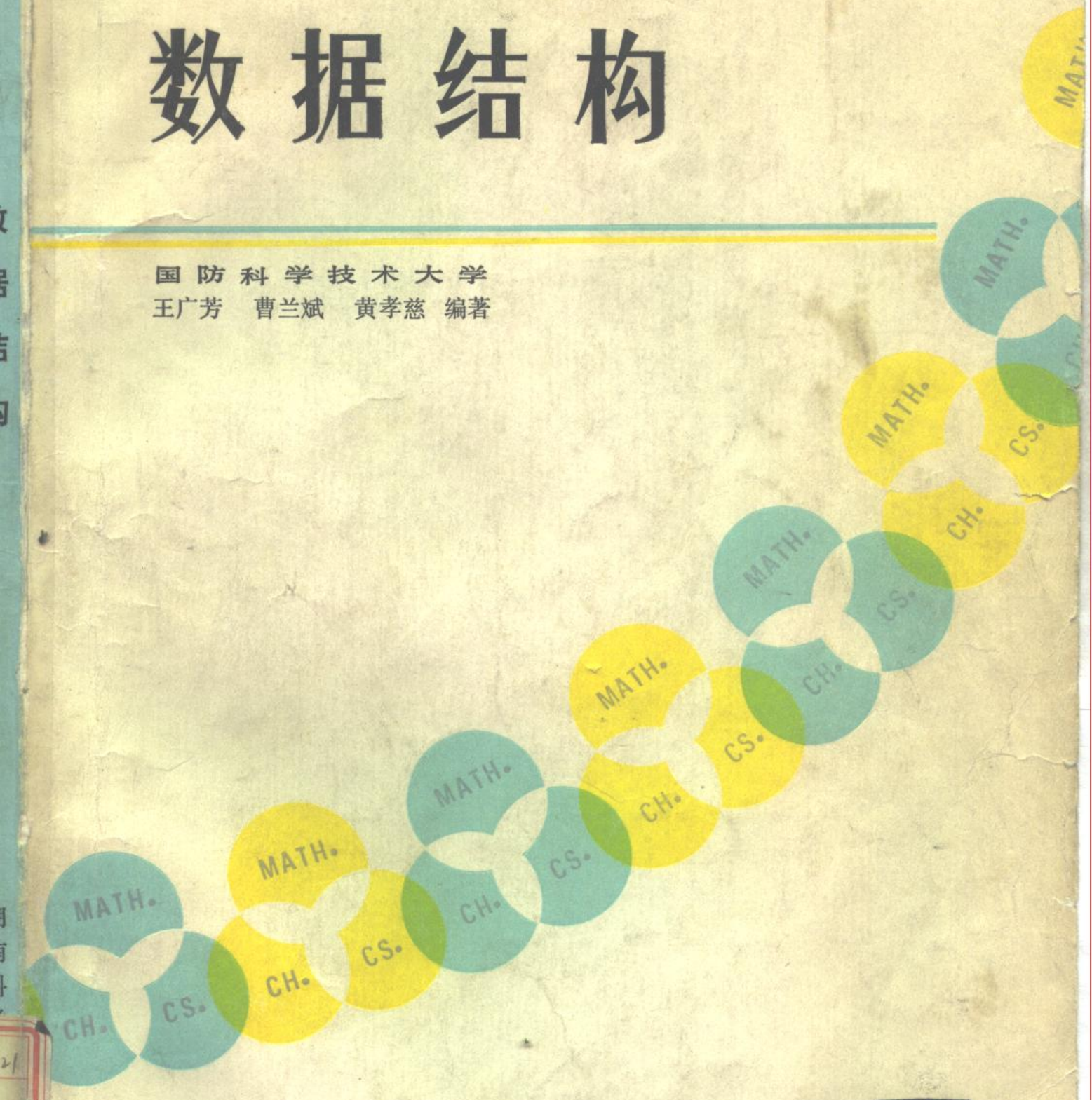
电子计算机软件

数据结构

国防科学技术大学
王广芳　曹兰斌　黄孝慈　编著



电子计算机软件

数据结构

国防科技大学　王广芳　曾兰斌　黄孝慈

湖南科学技术出版社

电子计算机软件

数 据 结 构

国防科学技术大学

王广芳　曹兰斌　黄孝慈　编著

责任编辑：周翰宗

*

湖南科学技术出版社出版

（长沙市展览馆路14号）

湖南省新华书店发行　湖南省新华印刷二厂印刷

*

1983年8月第1版　1985年9月第5次印刷

开本：787×1092毫米　1/16　印张：21.5　字数：533,000

印数：64,401——72,000

统一书号：15204·96　定价：3.50元

内 容 提 要

本书以通俗易懂的语言，采用独特的体系结构，按照由易到难，由简及繁的原则，详细介绍了各种数据结构的基本概念、逻辑特性和物理表示法，对各种结构定义了相应的运算，用类-PASCAL语言描述算法，并给出了各种算法的效率分析，以及这些结构在计算机科学与软件工程中的应用。此外，还介绍了存贮管理、文件、分类和查找等内容。最后给出数据结构应用示例。各章后面均附有习题，可供练习。

本书可作为大专院校计算机专业的教材，也可供从事计算机科学与软件工程的科技工作者及其它工程技术人员参考。

序　言

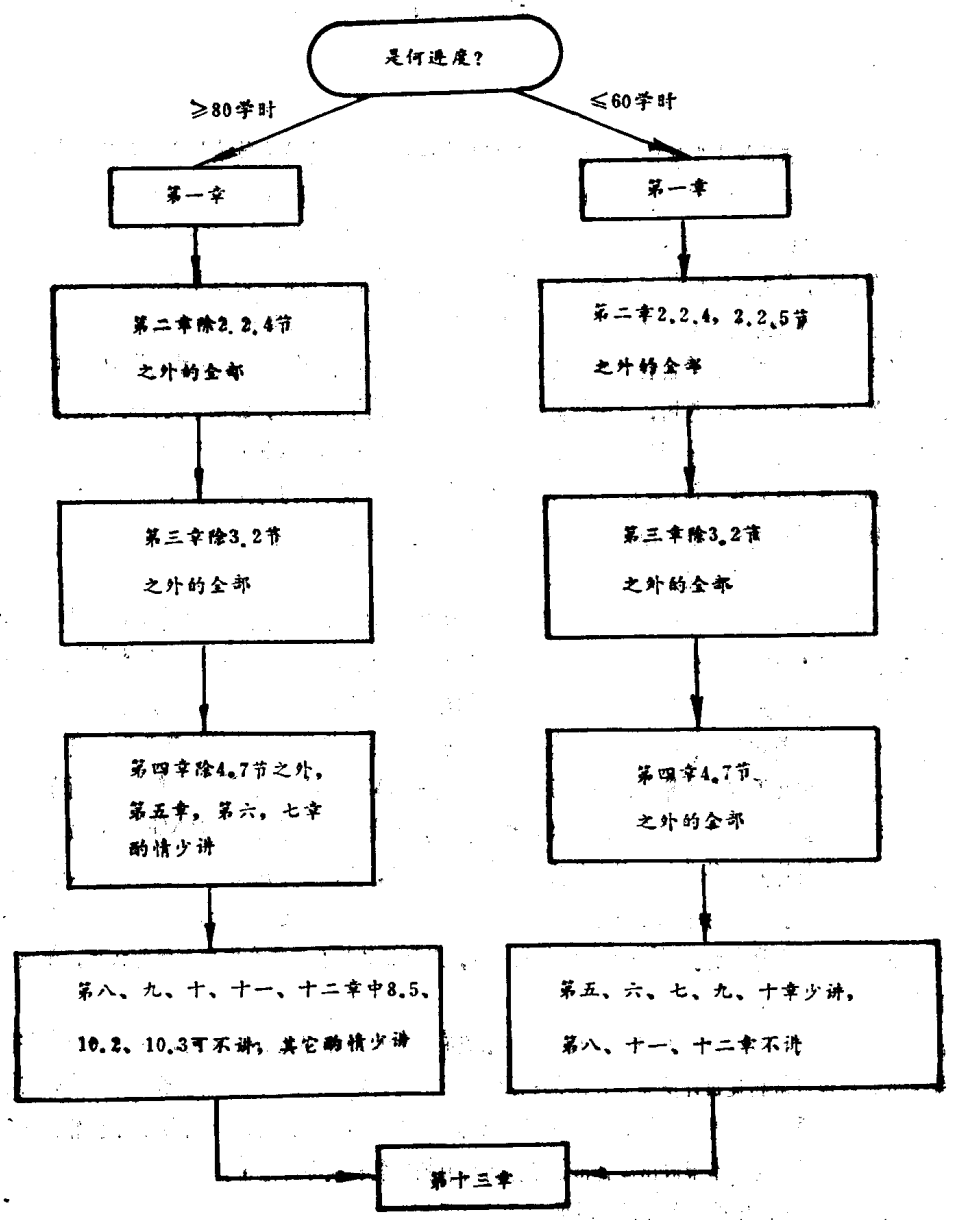
计算机科学和软件工程的惊人发展使其应用已深入到各个领域，不仅十分有效地解决了数值计算的各种问题，近年来，对诸如数据的分类与查找、情报检索、数据库、企业管理、系统工程、图形识别、人工智能以及日常生活等各领域的非数值问题，也能有效地加以解决。解决数值问题的许多理论、方法和技术一般不适于解决非数值问题，须要探索新的方法和技术来解决非数值问题，“数据结构”就是为研究和解决这些非数值问题而提出的理论和方法，是解决这些问题的重要基础知识，因此，从事计算机科学及其应用的科技工作者必须予以掌握。就计算机科学和软件工程的教学来说，“数据结构”也是一门不可少缺的核心课程。当前世界各地许多科技人员对于学习、研究“数据结构”都极其重视，并做了不少有益的工作，我国则由于众所周知的原因，使得计算机科学的发展受到了的阻碍，影响了人们对“数据结构”理论的学习和研究，所以还是一门新的需大力加强研究的课程，许多人都期望能有一本系统阐述“数据结构”的科技参考书，以满足教学和研究的需要。鉴于我们编写的《数据结构》讲义曾受到不少兄弟院校的赞赏和欢迎，使笔者倍受鼓舞，在领导和同行的大力支持、鼓励下，不揣浅陋，着手编写了可供教学和研究人员参考的《数据结构》，笔者希望能起到抛砖引玉的作用，在本书的启发下，同志们能获得更为出色的成果。

本书是笔者根据十余年来计算机软件实践与历时三载讲授“数据结构”的体会，并参考了国内外数据结构方面的最新文献编写而成的，全书结构力求能充分体现“数据结构”的特点和系统性，使之不是一种离散的结构，而是本着由简及繁、由易到难、循序渐进、前后衔接自然的原则，把问题的脉胳交待清楚，读之能一目了然，便于理解掌握各种数据结构的实质。因此，我们编写时是按静态、半静态、动态特性及线性结构与非线性结构的顺序展开的，着重研究了各种结构的逻辑特性和物理表示法，定义了相应的算法，并用易于教学的类－PASCAL 语言描述了算法，进行了适当的算法复杂性分析。全书中的各种算法虽不能直接上机运行，但只须稍加修改，就能变成上机执行的PASCAL 程序。故本书的实用性很强，不仅适宜于高等院校计算机科学与工程系作为专业的教学用书，亦适应于应用数学专业、电子、信息处理专业及从事计算机工作的科技人员参考。全书共分十三章，内容比较丰富，可供各种教学进度选择讲授，各章都备有习题供读者练习，授课计划可参考下页框图的程序制订。预期读者学完此书后，对数据结构会有较全面而系统的认识，能独立地进行有关领域的科研与工程实践。

中国科学院学部委员、国防科技大学付校长慈云桂教授在百忙中对本书进行了审定。国防科技大学陈火旺教授和俞成宜副教授就全书的内容与结构提出许多指导性意见；宫德荣、王鸿武副教授、吴家杰、郑若忠等同志提出了许多宝贵的建议；湖南大学陈滇英同志用初稿试行教学后，就全书结构曾提出许多建设性的建议；海军工程学院曾垂昌同志试作了部份习题，藉此机会谨向上述同志及为本书的出版付出了辛劳的所有同志致以衷心的感谢！

本书如有谬误不当之处，恳望读者批评、指正。

作者　一九八二年十二月于长沙


是何进度？
≥80学时
≤60学时
第一章
第二章除2.2.4节之外的全部
第三章除3.2节之外的全部
第四章除4.7节之外，第五章，第六，七章酌情少讲
第八、九、十、十一、十二章中8.5、10.2、10.3可不讲，其它酌情少讲
第一章
第二章2.2.4，2.2.5节之外的全部
第三章除3.2节之外的全部
第四章4.7节之外的全部
第五、六、七、九、十章少讲，第八、十一、十二章不讲
第十三章

目　录

第一章 绪 论

计算机科学是一门研究信息表示和处理的科学。而信息的表示和组织又直接关系到处理信息程序的效率。由于许多系统程序和应用程序的规模很大，结构又相当复杂，因此单纯依靠程序设计人员的经验和技巧已不能编制出高效率的处理程序，因而必须对程序设计方法进行系统地研究。这不仅涉及到研究程序的结构和算法，同时也涉及研究程序的加工对象——数据（信息）的结构。

§1.1 什么是数据结构

众所周知，计算机的程序是对信息进行加工处理。在大多数情况下，这些信息并不是无组织的，信息之间往往具有重要的结构关系，这就是数据结构的重要内容。那么，什么是数据结构呢？先不妨举例说明，然后给出明确含义。

例1 设有一个电话号码簿，它记录了n个人的名字和其相对应的电话号码，假定按如下形式安排：

$$(a_1,b_1)(a_2,b_2)\cdots(a_n,b_n)$$

其中a_i，b_i($i=1$，2，…，n)分别表示某人的名字和对应的电话号码。要求设计一个算法，当给定任何一个人的名字时，该算法能够打印出此人的电话号码；如果该电话簿中根本就没有这个人，则该算法也能够报告没有这个人的标志。

该算法的设计，严格说来依赖于计算机如何存贮人的名字和对应的电话号码，或者说依赖于名字和其电话号码是如何结构的。

如果名字和对应的电话号码在计算机中的排列次序没有任何规律，则只能用逐一比较的方法即线性查找法来解决问题，这种方法是相当费时的。若城市很大，具有电话的人又很多，那么自动电话查号管理系统采用这种方法是行不通的。

然而，假定把电话号码簿进行适当的组织，即按字典顺序排列名字，在名字之后跟随着对应的电话号码。在此组织之下，问题就容易解决了。例如，若查找某人的电话号码，他的名字的第一个字母是S，那么只需要查找以S打头的那些名字就可以了，另外的25个字母打头的名字就不需要查找了。

由此可见，数据的结构，直接影响算法的选择和效率。

上述的电话号码簿就是一种数据结构问题。可以将名字和对应的电话号码这种数据设计成二维数组形式，也可以设计成表结构，当然还可以设计成向量，把名字和对应的电话号码一起作为向量的一个元素。

假定名字和其电话号码数据逻辑上已安排成n元向量的形式，它的每个元素是一个数对(a_i,b_i)，$1\leqslant i\leqslant n$ 即有向量

$$((a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)).$$

对于这样一个电话号码簿，当需要增加一个名字和电话号码时，或者某人调动到其它城市工

作，他的电话应该拆掉时，就需要在已安排好的结构上进行插入或删除。这样，就会产生下面一系列问题：将增加的名字和对应的电话号码插入到什么地方？是插入到前头，还是插入到末尾，或者插入到中间某个合适的位置？插入后，对原有的数据是否有影响？有什么样的影响？删除某人的名字和对应的电话号码后，原来的某些数据要否移动？若需要移动，则应如何移动？这一系列问题说明为适应数据的增长或减少的需要，还必须对这样的数据结构定义某些运算。上述只涉及到两种运算，即插入和删除运算，当然也会提出其它一些可能的运算。

对于这些运算，显然不可能用手工的办法完成，因为那样既费时，又易出错，应该用计算机来完成。这样就要求设计“插入”和“删除”的算法，因此数据结构还要提供每种结构类型所定义的各种运算的算法。

例 2　建立人事档案也是数据结构的一个典型例子。假定建立一个大学的教师和学生档案，逻辑上这种结构应如图1.1形式。

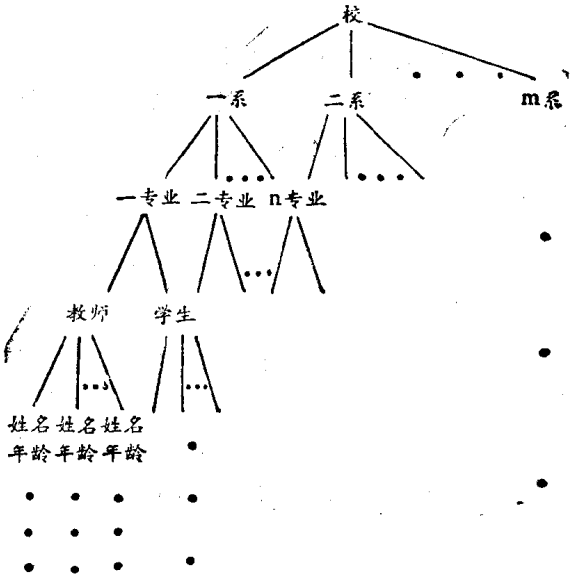


图1.1

这种形式实际上是一种树结构。下面讨论关于询问某个教师或学生有关情况的处理问题。为了处理这个问题，首先必须解决如何把这些数据存贮到计算机中，也就是要解决数据的物理表示问题。假如这个问题解决了，那么又如何实现对于某教师或学生有关情况的询问呢？如果询问者已经给出了该教师或学生所属的系和专业，则该问题还比较容易解决。若询问者根本不知道所询问的教师或学生属于何系何专业，那么问题的解决就要复杂得多，这时必须按系按专业逐步查找，这就是所谓树的遍历问题。

当然，还会出现下列问题：当需要增加一个系时，将此系插入何处？是将它作为 1 系呢，还是作为3系，或者作为m＋1系呢？这就导致了所谓树结构的插入问题。另外，如果某系某专业被取消了，此时必定会影响这种结构，怎样将该专业的所有教师和学生的数据从该结构中删除呢？这实际上是树结构的删除问题，而删除的过程势必引起有关系和专业的变动。如果需要进行重新整理，就会涉及到既删除又插入的问题。因此，为了适应这种数据的查找、增长和缩小，必须定义相应的查找、插入、删除的运算及其算法，并应当保证在插入和删除之后不能破坏原来的结构类型。

通过以上二例可以直观地认为：数据结构就是研究数据的逻辑结构和物理结构以及它们之间的相互关系，并对这种结构定义相适应的运算，设计出相应的算法，而且确保经过这些运算后所得到的新结构仍然是原来的结构类型。

在数据结构中，往往涉及数据类型与数据对象的概念，那么它们之间有什么不同呢？一般说来，数据类型是指在一种程序设计语言中，变量所具有的数据种类。如 FORTRAN 语言中，变量的数据种类有整型、实型、复型等。由于每种程序设计语言，有一组内部的数据类型，这就意味着程序设计语言允许变量指定数据类型，并且对变量提供一组有意义的运算。有些数据类型是容易提供的，因为这些类型的运算，机器语言本身就实现了，如整数与实数的算术运算就是一个例子。而有些数据类型的提供，需要相当大地努力才能实现。

数据对象是指某种数据类型的元素集合。如整数的数据对象是{0，±1，±2，…}。英文字符类型的数据对象是{A,B,…,Z}。数据对象可以是有限的，也可以是无限的。

数据结构不同于数据类型，也不同于数据对象，它不仅要描述数据类型的数据对象，而且要描述数据对象各元素之间的相互关系，比如需要描述数据对象元素之间的运算，并使得这些运算可以合法地用于数据对象的各元素上。

数据结构的研究范围主要包括：研究各种数据结构的性质，即不仅研究数据的逻辑结构和物理结构（也就是每种数据结构在计算机中是如何表示的），而且对每种结构定义相适应的运算，并使用某种高级程序设计语言给出各种运算的算法，分析算法的效率，同时也研究各种数据结构在计算机科学及软件工程中的某些应用，讨论数据分类和检索等方面的技术。数据结构类型包括向量、数组、记录、栈、队列、简单链表、广义链表、串、树、图、文件等。

数据结构的侧重点在于实践技术而不是理论方面，它是以程序设计、离散数学为基础，但在分析、处理数据结构和算法时，使用了有关的理论工具，特别是集合论、图论、概率统计等方面的理论基础。

§1.2 数据结构发展概况

回顾一下数据结构课程的发展与形成过程对于理解数据结构的内容和重要性是很有意义的。六十年代初期，国内外还没有专门的《数据结构》课程，但数据结构的有关内容已散见于《编译原理》和《操作系统》之中。六十年代中期，有些国家的大学开始设立有关的课程，但当时课程的名称并不叫"数据结构"，而命名为"表处理语言"。它的主要内容是研究当时已经出现的几个表处理系统。如 J. Weizenbaum 在50年代初设计的 SLIP 系统（简单表处理语言）；1954～1959年期间由 A. Newell 等人设计的 IPL-V 系统（信息处理语言）；1959～1960 年，J. Mc Carthy 设计的 LISP 系统(表处理语言)；以及1962年由D. Farber 等人设计的 SNOBOL 系统（串处理语言）。它们的共同特点是数据对象的结构形式或者是表结构，或者是树结构。如 LISP 的数据结构就是二叉树结构，SNOBOL 的数据结构是表和树结构。

上述的几种语言是以数据为中心，为处理非数值问题设计的，如 LISP 就是为解决人工智能问题而设计的表处理语言。而我们所熟知的 FORTRAN、ALGOL 等算法语言则是为解决数值问题而设计的，它们侧重于以程序为中心。如 ALGOL-60分程序和过程就是以程序为中心的例子。

以程序为中心的观点侧重于建立程序，只是当数据成为程序加工的对象时，才考虑到数据。这种观点适合于数值计算问题，这类问题属于在简单数据结构上进行复杂函数变换的问题。以数据为中心的观点是把数据结构作为问题的中心部分(如数据库)，而把程序看成是围绕着数据结构缓慢爬行的小虫，它时而询问，时而修改或扩充当前驻留在内存中的数据。这种观点适合于航空订票系统、信息管理系统、情报检索系统等非数值问题的解决，它们都要求采用复杂的数据结构描述系统的状态，它们的运算是实现对于数据结构的访问或改变等。某些科学家曾断言，程序设计以数据为中心的观点，将对未来程序设计语言的设计产生重大影响。

1968年，美国某些大学的计算机科学系教学计划中(如68教程)，曾明确规定"数据结构"为一门课程，但该课程的内容范围并没有具体限定。其后发表的一些文章和出版的书籍中，对"数据结构"这个术语有各种不同的理解和解释。最初，数据结构几乎和图论是同义语，

特别是表和树的理论，(这些理论是描述分层数据的有力武器)。随后这个概念又扩充到包括网络、代数、集合论、关系等现在称之谓“离散数学结构”的那些内容。它与现在称为“数据结构”的某些内容混合在一起，总称为“数据结构”。

由于数据必须在计算机中进行处理，因此不能局限于数据本身的数学概念的研究，还必须考虑数据的物理结构，即数据在存贮器中的表示问题，这就进一步扩大了数据结构的内容。

特别是1968年，著名计算机科学教授，美国D. E. Knuth 所著的《The Art of Computer Programming》Volume 1 出版，对计算机科学的发展做出了重大贡献。该书作者论证了任何语言都可以采用象表处理语言那样的技术。这部著作比较全面地、系统地讨论了几种数据结构(Knuth 称为信息结构)，定义了运算，并用英文加上汇编语言描述算法，比较详细地分析了算法的效率。尔后，逐渐将数据的有关数学概念独立出来，形成了现在的“离散数学结构”，而把数据的逻辑结构、物理结构以及对每种结构所定义的运算形成了“数据结构”的主要内容。

近年来，由于数据库，情报检索系统的不断发展，在数据结构的技术中，又增加了文件结构，特别是大型文件的组织等方面的内容。

经过以上论述可知，数据结构技术的形成还不太久，仍然处于迅速发展的阶段。一方面，它的内容在不断发展和扩充；另一方面，计算机硬件的发展和更新也必将对数据结构产生重大影响。

§1.3 数据结构课程与其它课程的关系

数据结构与数学、计算机硬件和软件有十分密切的关系，它们之间的关系如图1.2所示。

由图1.2可以看出，数据结构是介乎于数学、计算机硬件和计算机软件之间的一门计算机科学专业的核心课程，是高级程序设计语言、编译原理、操作系统、数据库、人工智能等课程的基础。同时，数据结构的技术也广泛应用于信息科学、系统工程、应用数学、以及各种工程技术领域。

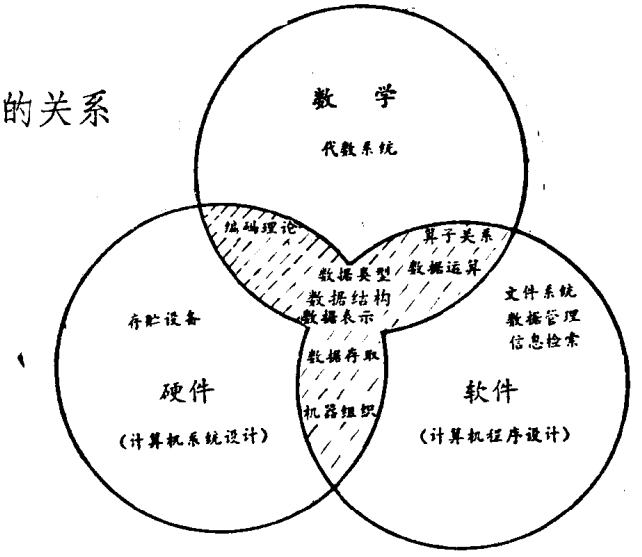


图1.2 数据结构与其它课程的关系

§1.4 算法所用语言的几点说明

数据结构中算法设计是十分重要的一部分内容，为了便于理解和掌握算法的思想和实质，选择一种合适的高级语言描述算法就是一个很重要的问题。根据这个指导思想，我们选择了一种抽象的高级语言来描述算法，它的绝大多数语句是 PASCAL 语言中的相应语句，个别地做了某些修改。因此它不是严格的 PASCAL 语言，而是类—PASCAL 语言。由于书中的算法是采用类—PASCAL 语言描述的，所以它们并不是直接可以在机器上执行的程序或过程。它的优点很多：易于书写，便于阅读和理解算法，使读者把注意力集中于算法的实质，而不是把精力花费在某种实际高级语言的许多具体约定之上。同时，该书的算法中略去了对

于变量的说明，也就是说对于变量没有使用说明性语句加以说明。

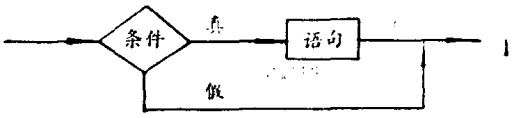
该类-PASCAL 语言有下列几个主要的语句：

(1) 赋值语句

〈变量〉：＝〈表达式〉|〈函数标识符〉：＝〈表达式〉

(2) 复合语句　由两个以上的语句构成的语句组称为复合语句。复合语句中的成分语句组按其书写顺序执行，成分语句组写在 begin 与 end 之间，实际上 begin 与 end 是起括号作用的专用符号，复合语句的形式为：

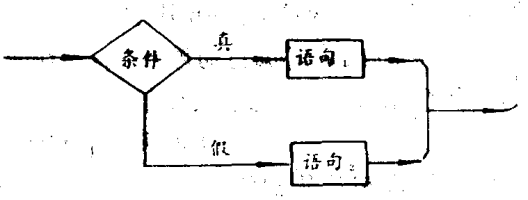
begin〈语句〉{；〈语句〉} end



(3) if 条件语句　if 条件语句的形式有两种：

if〈条件〉then〈语句〉|if〈条件〉
then〈语句$_1$〉else〈语句$_2$〉

或



它的含义是仅当条件为真时，才执行 then 之后的语句；否则，分两种情况，或者执行 if 以下的语句，或者执行 else 之后的语句，它们的执行过程的逻辑示意图见图1.3。

图1.3

(4) case 情况语句　这里所定义的 case 语句和 PASCAL 稍有不同，其形式和执行过程的逻辑示意图如图1.4。

case
〈条件$_1$〉　〈语句$_1$〉
〈条件$_2$〉　〈语句$_2$〉
⋮　⋮
〈条件n〉　〈语句n〉
〈否则〉　〈语句$_{n+1}$〉
end

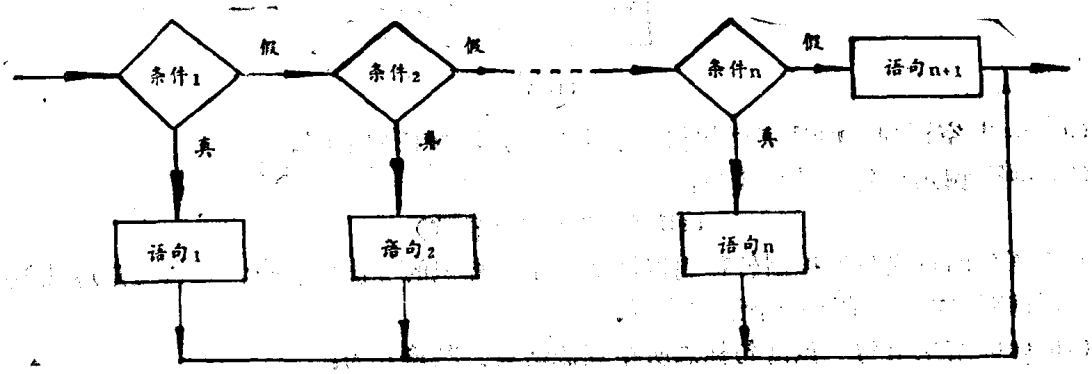


图1.4

(5) while 循环语句　其形式为：

while〈条件〉do〈语句〉

它的含义是只要条件成立，就执行 do 之后的语句，直到条件变成假时循环才结束。如果条件一开始就是假，那么循环中 do 后面的语句根本就不执行，而跳过循环顺序执行 while

之下的语句。其逻辑示意图如图1.5所示。

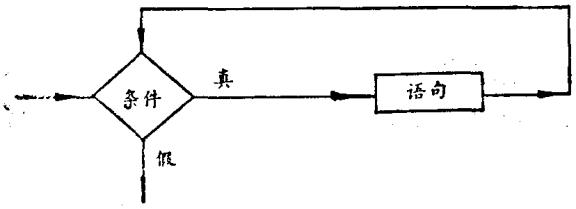


图1.5

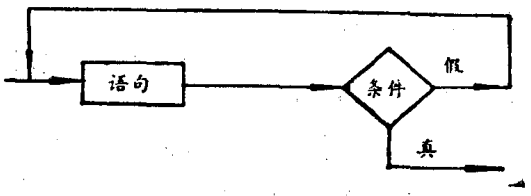


图1.6

(6) repeat 循环语句　其形式为：

repeat 〈语句〉 until〈条件〉

它的意义是 repeat 与 until 之间语句重复地执行若干次(至少一次)，直到 until 之后的条件为真才停止重复执行而顺序执行 repeat 之下的语句。其逻辑示意图如图1.6。

(7) for 循环语句　其形式为：

for $i=e_1$ step (h) to e_2 do〈语句〉

for 循环语句是从初值e_1开始，步长为h到终值为e_2都执行 do 之后的〈语句〉，这里e_1和e_2只能是整型表达式，步长h为整数。如果$h=1$，则步长h可省略。若步长h为负整数，则该语句相当于 PASCAL 语言的

for $i:=e_1$ downto e_2 do

它的逻辑示意图如图1.7。

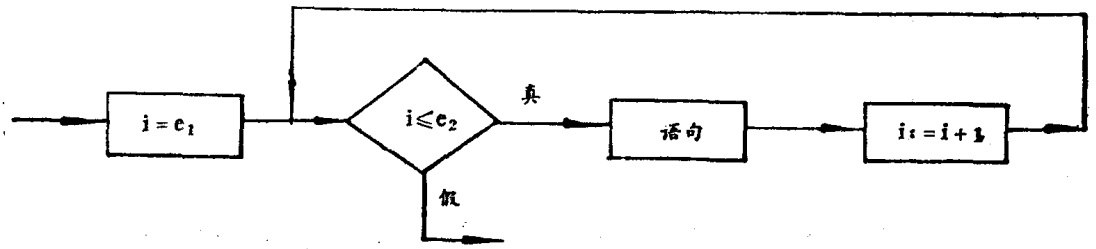


图1.7

(8) null 空语句　null 空语句相当于空操作，不做任何事情。

(9) call 调用语句　其形式为：

call 算法名字 (参数)

调用语句的意义是把控制转移到以算法名字所标识的算法中执行，在被调用算法执行完之后再返回到call语句的下一个语句执行。

(10)end 结束语句　该语句标志所指算法结束。其形式为：

end (算法名字)

(11)输入/输出语句　其形式为：read (参数)

write (参数)

输入/输出语句有时也使用input 或 print 表示，只关心它们本身的作用，没有定义相应的格式语句。

另外，为方便起见，有时在粗略地描述算法时，也使用某些英语句子和汉语句子。

为了对算法或某些步骤作必要的解释，采用双斜线把解释的内容括起来，这种双斜线括号及其解释的内容可以出现在算法的任何地方。

最后，应该强调指出，用该语言编写的算法是结构式，虽然有时看起来算法长一些，但它便于阅读、检查和修改。对于书中所有算法，除了几个粗糙的算法之外，都能较容易地变成 PASCAL 语言程序在机器上执行。