数据結构講义

上　册

唐策善
牛忠荣　编

中国科学技术大学六系

前　言

在1979年的全国计算机专业教材会议上，《数据结构》被定为必修课程。并委托清华大学的同志负责编写“统编教材”。现在该教材已基本定稿。但因为排版印刷的时间紧，任务重，预计到明年秋季开学时才能使用它。为了满足我校77级～79级的教学要求，我们参考有关文献和统编教材的原稿，结合我校的具体情况，编写了这套讲义。

这套讲义分上、下两册。上册包括绪论、线性表和向量、链表、串、树等五章。下册包括图检索、内部排序、外部排序、文件等五章，大约20多万字。前面六章讲述了几种基本的数据结构及其相应算法和应用。后面几章介绍数据结构在系统应用中经常遇到的问题和算法，最后一章是为了适应数据库系统的发展需要而安排的。这些内容是计算机各专业的基础知识。学习这些内容，也为今后将要学习的“编译程序”、“操作系统”、“数据库系统”等课程打下了基础。

这套讲义使用的算法语言，是采用参考文献(1)中介绍的SPARKS语言。这是一种伪语言，功能较强，适合于算法的描述。为了上机计算实习，可以设计一个预处理器，将SPARKS语言翻译成某一种具体的语言。

这套讲义是按自成体系的精神编写的。如果学习这门课之前，已经学习完它的先行课，那么，相互重迭的内容可以删去，或者只作简要复习即可。如果学时安排紧张，某些章节可作为课外阅读材料，不必在课堂上讲授。但对计算机软件专业来说，这些内容都是必须具备的。

《数据结构》既是计算机各专业的一门重要基础课，又是一门较新的课程。这套讲义是编者边学边写编成的，由于编者水平所限，加上时间紧迫，讲义中存在问题肯定不少，缺点和错误在所难免，敬希读者批评指正。

在讲义编写过程中，得到了清华大学计算机工程与科学系沈佩娟、严蔚敏两位老师的热情帮助，编者谨致深切谢意。

唐泽圣　　牛忠荣

一九八〇年十月于合肥

数据结构讲义 （上册）

目　　录

第一章　绪　论

1·1　引　言

近二十年来，电子数字计算机一直在飞速发展。这种发展不仅是指计算机本身运算速度的不断提高，信息存贮容量的日益增大，而且也指计算机应用范围的逐渐扩大。早期的电子计算机几乎只是用于科技计算，六十年代以后，计算机则更多地用于数据处理和实时控制。与此相应，计算机加工处理的对象也从简单的数值发展到字符，进而发展到更复杂的具有一定结构的数据。从程序设计的观点出发，为了设计出效率高，可靠性强的程序，人们必须对程序设计的方法进行系统的研究。这就要求程序设计人员不但要掌握一般的程序设计技巧，而且还要分析研究计算机程序加工的对象，即研究数据的特性以及数据之间存在的关系。这样一来便促进了数据结构这门学科的发展。

在国外，数据结构作为一门独立的课程是在１９６８年才开始的。在这之前，它的某些内容曾在其它课程（例如“表处理语言”）中有所讲述。１９６８年，在美国一些大学的计算机系的教学计划中，虽然把《数据结构》规定为一门课程，但对课程的范围仍没有作明确的规定。当时，数据结构几乎和图论——特别是和表、树的理论是同义语。随后，数据结构这个概念被扩充到包括网络，集合代数论、格、关系等方面，从而变成了现在称之为“离散结构”的内容。然而，由于数据必须在计算机中进行处理，因此不仅要考虑数据本身的数学性质，而且还必须考虑数据的存贮结构，这样一来，就要求进一步扩大数据结构的内容。近年来，随着数据库系统的不

断发展，在数据结构课程中又增加了文件管理（特别是大型文件的组织等）的内容。

由此可见，数据结构在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件（特别是编码理论、存贮装置和存取方法等）的研究范围，而且和计算机软件的研究有更密切的关系。无论是编译程序还是操作系统，都涉及到数据元素在存贮器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以使查找和存取数据元素更为方便。有人在描述数据结构这门课程所研究的内容时，曾将其归纳为数学，计算机硬件和计算机软件这三个学科范围互相重迭的区域（图1·1）。

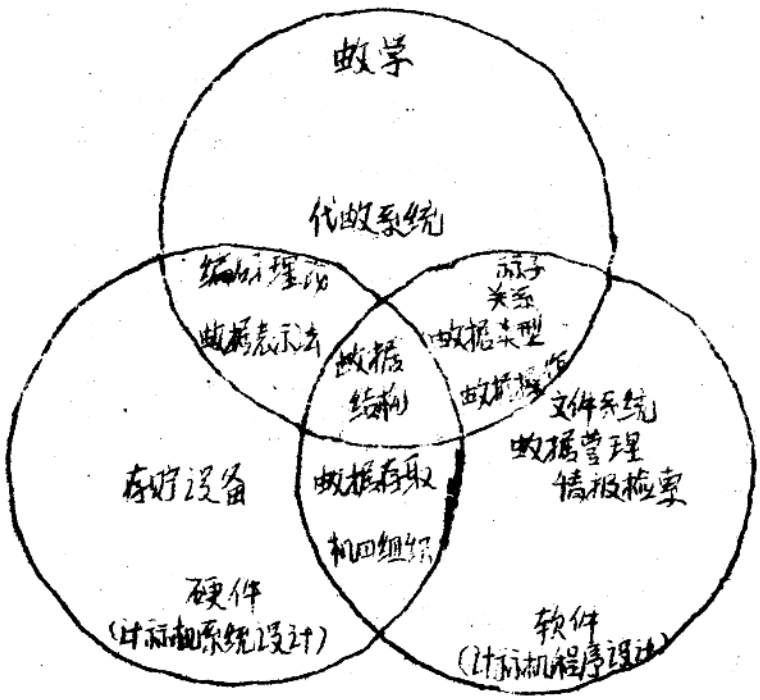


图1·1 数据结构的研究对象

在我国，最近几年也已将“数据结构作为计算机专业教学计划中的核心课程之一，这是因为，在计算机科学中，数据结构这门课程的内容不仅是一般程序设计的基础，而且是设计和实现编译程序、操作系统、数据库系统及其它系统程序的重要基础。

1·2　什么是数据结构

什么是数据结构？这是一个难于直接回答的问题。然而，我们可以把问题分割成两个部分。先回答什么是数据，然后再回答什么是数据结构的问题。

直观地说，数据是描述客观事物的数、字符、以及所有能输入到计算机中并被计算机程序处理的符号的集合。它是计算机程序使用和加工的“原料”。例如，一个简单的计算程序所使用的数据只是一些实数和整数，而对于一个编译程序来说，它所使用和加工的数据则是程序员写的源程序（即字符的集合）。

数据中最基本的单位是数据元素（Data Element）。性质相同的数据元素的集合就叫做数据对象（Data Object）。例如，整数的数据对象是集合D＝{ 0，±1，±2，……}；按字母顺序排列的字母字符数据对象是集合D＝{ 'A'，'B'，…，'Z'}。因此，数据对象是数据的一个子集，它可以是有限的，也可以是无限的。

除了最简单的数据对象外，通常，数据对象中的数据元素不是孤立的，而是彼此相关的。这种彼此之间存在的相互关系就叫做结构。数据结构是数据元素之间抽象化的相互关系，并不涉及数据元素的具体内容。数据结构和数据对象不同，描述一个数据结构不仅

要描述数据对象，而且要描述数据元素之间的相互关系，这个相互关系可以用一组能对数据对象诸元素进行的运算及运算规则来描述。例如，复数是一个简单数据结构，它的数据对象是一对有序的实数（即实部和虚部，它们之间存在着次序关系）的集合。允许对这种数据元素（一对有序的实数）进行的算术运算可以有加、减、乘、除等。于是，一对有序的实数的集合，再加上如何进行加、减、乘、除等运算的描述，这两个方面便构成了复数这个数据结构的定义。这就是说，在研究这种数据结构时，所需研究的不是复数的值，而是复数的表示、存贮、以及复数的各种基本运算。

数据结构有时是很复杂的。这时它具有一定的层次，数据结构表示了各数据元素之间的相互关系，而每个数据元素本身又可以是别的数据结构。

严格地说，数据结构包括数据的逻辑结构和数据的物理结构。数据的逻辑结构仅考虑数据元素之间的逻辑关系；而数据的物理结构，则是指数据元素在计算机存贮器中的表示及其配置。为了使其不至混淆，我们姑且把前者统称为数据结构，后者统称为存贮结构。一种数据结构㊀可以通过映象得到与它相应的存贮结构（包括字符串、栈、队列和数组）和非线性结构（包括树和图）。它们分别可以通过顺序映象（见下一章）和非顺序映象得到不同的存贮结构。

在本节的最后，我们还要引进一个后面将要用到的术语，这就是数据类型（Data Type）。数据类型指的是程序设计语言中允许的变量种类。例如数组、文件等。每种程序设计语言都有一组内部数据类型。例如：在ALGOL语言中，数据类型有整型、实型和

㊀ 指数据的逻辑结构，以下同。

布尔型；在PAS CAL语言中，除了有标准的整型、实型、布尔型、字符型，以及由用户自己定义说明的纯量类型之外，还允许构造用户定义的构造类型。

一个数据类型不仅定义了一个变量可以设定的值的集合（程序里所出现的每个变量，必须与一个，而且仅仅与一个数据类型相联系），而且还规定了允许对变量进行的一组运算和运算规则。例如，ALGOL语言中的整型变量 数值是某个有定义的整数子集的元素（子集的大小由具体的机器决定），允许对整型变量进行的算术运算有加、减、乘、除、整除和乘幂等（其中，除了整除和乘幂之外，运算结果仍为整数）。因此，我们可以把数据类型看成是程序设计语言中已经实现了的数据结构。

1·3 算法及其描述语言

在计算机科学中，“算法”（Algorithm）的概念总是基本的概念。在本书中，在讨论各种数据结构的基本运算时，都将给出相应的算法。

一个算法，就是一个有穷规则的集合，这些规则规定了一个解决某一特定类型的问题的运算序列。此外，一个算法还有五个重要的特性：

1） 有穷性 一个算法必须总是在执行有穷步之后结束。

2） 确定性 算法的每一个步骤，必须是确定地定义的。对于每种情况，有待执行的动作必须严格地和不含混地规定出来。

3） 输入 一个算法有0个或多个的输入。它即是，在算法开始之前，对算法最初给出的量。这些输入取自于特定的对象集合。

4） 输出 一个算法有一个或多个的输出。它即是，同输入有某个特定关系的量。

5） 能行性 算法中所有有待实现的运算必须都是相当基本的，即是说，它们原则上都是能够精确地进行的，而且人们用笔和纸做有穷次即可完成。

应当说明，具备算法的其它所有特征，但却不具备有穷性的步骤，不叫算法，可以叫做“计算方法”或“程序”，例如计算机的操作系统，除系统失灵以外，它是绝对不结束的，而是在一个循环内等待，直到输入更多的作业。这样的操作系统就不能称为算法，在本书中我们只处理总能结束的程序，因而对“算法”和“程序”也就不加区别了。还要说明，有穷性的限制对于实际的使用说来，实在是不够严的。一个有用的算法应该不仅要求有穷的步骤，而且应该是很有限的步骤，即取一个合理的数目。

描述一个算法，可以有许多方式。划框图的方法是大家熟悉的，这种方法把每一个处理步骤放在一个框框中，在框内用文字注明其功能，用箭头指向下一个执行步骤，用不同形状的框框代表不同类型的操作。在本书中我们不采用划框图的方法。此外，用现有的语言，诸如汇编语言、扩展BASIC语言、ALGOL、COBOL、LISP、PASCAL、PL/1、SNOBOL等，也能写出我们的算法。但也会带来一定的困难，我们不选用上述的某一种语言。这是因为：1）我们希望读者把注意力集中在算法上，而不要过多注意所用语言的特性。2）将要采用的语言，和上面提到的许多语言相比，有许多相近之处，跟人们的习惯亦相吻合，用它写出的算法简洁明了，有利于人们阅读和记忆。我们选用的语言可说是一种伪语言，取名SPARKS，它具备上述各种语言的许多优点，虽然不是某一种具体的语言，但是

容易翻译成某一种具体的语言。图1·2指出，在任何机器上，一个SPARKS程序是如何执行的。在参考文献1的附录A中，给出了SPAKS到FORTRAN的翻译程序，即图1·2中对于FORTRAN语言的予处理器。

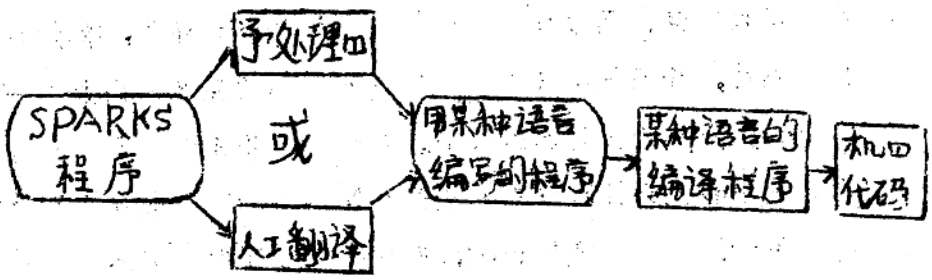


图1·2　　SPARKS　的翻译

SPARKS 能对数字、布尔量和字符执行操作。它包括赋值语句、条件语句、循环语句、输入—— 输出语句等等。编写程序时，语句之间用分号“；”隔开，一行可写几个语句，但一定要记住用分号隔开。

1，赋值语句

赋值语句为：

Variable ←—— expression

表达式或者是算术表达式，或者是布尔表达式，或者是字符形式的表达式。在布尔表达式的情况下，表达式的值，只能是两个值

TRUE　或　FALSE

中的一个。为了生成这些值，提供了逻辑操作符

AND , OR , NOT

和关系操作符

$<$, $\neq$, $\leq$, $=$, $\neq$, $\geq$, $\not>$, $>$ 。

2，条件语句

条件语句具有如下两种形式：

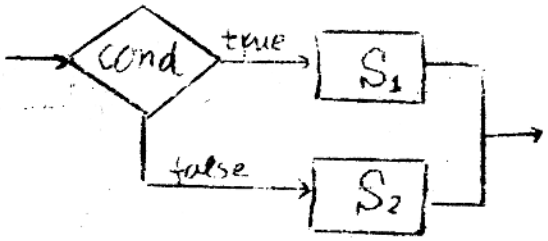
1)， IF —— THEN 语句

IF Cond THEN S_1

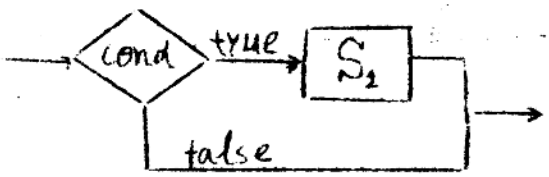
ELSE S_2

或 IF Cond THEN S_1

其中 Cond 是布尔表达式，S_1 和 S_2 是任意的 SPAR KS 语句组。如 S_1 或 S_2 包含的语句多于一个语句，则用方括号把它们括起来。我们用下图给出条件语句的意义



或



我们假定：条件表达式是按“短路”方式来计算的，即给定条件表达式 ($Cond_1$ OR $Cond_2$)，如果 $Cond_1$ 为真，

则不计算 $Cond_2$；或者，给出（$Cond_1$ AND $Cond_2$），如果 $Cond_1$ 为假，则 $Cond_2$ 不计算。

2），CASE 语句

CASE

:$Cond_1$: S_1

:$Cond_2$: S_2

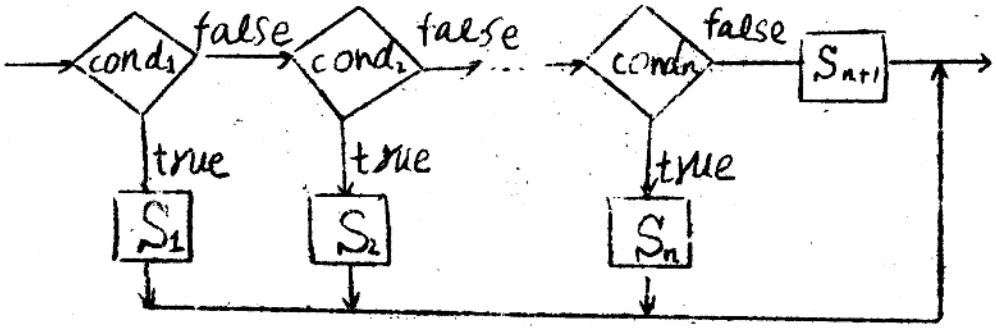
⋮

:$Cond_n$: S_n

: ELSE : S_{n+1}

END

其中：$S_i(1 \leq i \leq n+1)$ 是SPARKS语句组，而ELSE子句是可有可无。下面图，描述了它的语义：



3，循环语句

循环语句有四种：

1）WHILE —— DO 语句

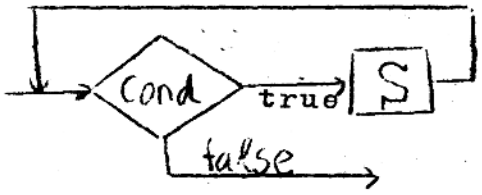
它的形式是

WHILE Cond DO

S

END

这里Cond与前面一样，S同上面的S_1，这种语句的意义由下图所示：



2） REPEAT —— UNTIL 语句

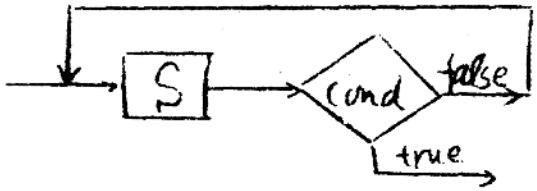
这种语句的形式是

REPEAT

S

UNTIL Cond

其中：S和Cond与上面的相同，这种语句的意义是



与 WRILE —— DO 语句相反，REPEAT —— UNTIL 语句保证S语句组至少执行一次。

3），LOOP ——. FOREVER 语句

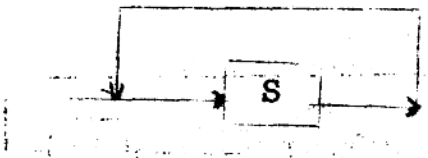
这种语句的形式为：

LOOP

S

FOREVER

它的意义是



从表面看，这是一个无限循环！但我们在使用这个语句时，在S语句组内必包含某个条件测试语句，而此条件测试语句就能导致一个出口。导致循环出口的方式之一是使用

GO TO label

语句。它将控制转移到标号为"label"的语句。语句标号可以在程序的任何地方。GO TO 语句的一种特殊形式是指令

EXIT

它将控制转移到包含它的最里的那层循环后面的第一个语句。这个循环语句可以是WHILE、REPEAT、FOR 或 LOOP —— FOREVER语句。EXIT 可以是有条件的，也可以是无条件的。例如

LOOP

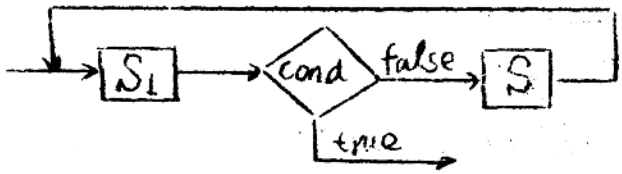
S_1

IF Cond THEN EXIT

S_2

FOREVER

它将按如下的图来执行



4），FOR 语句

这种循环语句的形式为：

```
FOR vble←start TO finish BY increment DO
        S
END
```

其中：vble是变量，而start、finish和increment是算术表达式，变量和常数是表达式的简单形式。子句“ BY increment”是可有可无的，如果它不出现，则取成+1。这个循环语句也可用SPARKS的其它语句写出如下：

```
vble ← start
fin ← finish
incr ← increment
WHILE (vble - fin)*incr ≤ 0 DO
        S
    vble ← vble+incr
END
```

4，CALL 语句

它的形式是

```
CALL name(Parameter list)
```

这个语句是调用子程序时用的，name 为被调用的子程序名字，