数据结构

朱 望 规

.12
F4/1

西安交通大学出版社

数 据 结 构

朱望规 编著

西安交通大学出版社

内 容 提 要

本书系统地介绍了数据结构的主要技术及其有关算法的设计和分析。

全书共分九章，第一章为引论；第二、三章介绍 PASCAL 语言和线性数据结构；第四章介绍树结构；第五章至七章介绍内、外分类和检索；第八章介绍文件；第九章介绍一些问题的进一步探讨。

本书内容循序渐进，深入浅出，取材新颖，用 PASCAL 语言描述了绝大部分算法，并给出相应的框图，还将给出执行算法与框图各步骤的数字例子，便于读者理解。

本书的对象是非计算机专业的研究生和大学生以及从事计算机应用工作的工程技术人员，亦可作为计算机专业的教学参考书。

数 据 结 构

编 著 朱望规

责任编辑 刘 峻

西安交通大学出版社出版

（西安咸宁路28号）

西安交通大学出版社印刷厂印装

陕西省新华书店发行·各地新华书店经售

开本 787×1092 1/16 印张 19.75 字数 484 千字

1985年11月第一版 1986年9月第二次印刷

印数：5001—10,000

统一书号：13340·038 定价 3.90 元

前　言

二十世纪四十年代问世的电子计算机是极其重大的科技成果，标志着一次新的工业革命的到来。三十多年来，计算机的发展异常迅速，应用领域十分广阔。在我国，随着国民经济的发展，计算机技术也必将得到广泛的应用。

据机械工业部统计，全国约有三百万台机床，其中百分之三十要加配微型计算机或单板机使之成为数控机床，这样，就需要数以万计的电子计算机应用软件人员，特别是计算机辅助设计（简称 CAD）人员。因此我国大批技术人员需要学习维护、使用 CAD 硬件和软件的知识，并从事 CAD 的研究工作，他们面临着一个知识更新的任务。如果按照计算机软件专业的要求，需要学习程序设计、算法语言、编译原理、操作系统、数据结构、数据库，绘图软件、交互作用计算机绘图原理、软件工程等一系列课程，这对工程技术人员来讲是不现实的，必须寻找一条切实可行的途径。

作者从 1973 年起从事统一计算机绘图软件的研究工作，多次为各种 CAD 讲习班讲课，并为研究生讲授数据结构、数据库导论、数据库设计、计算机绘图等课程，整理出一套非计算机专业人员学习计算机知识的教材，《数据结构》就是其中的一种。

本书的写作，遵循了 PASCAL 语言创造者 N. Wirth 教授的观点，把程序设计、PASCAL 语言与数据结构融为一体。作者参考了 D. E. Knuth 的经典著作《The Art of Computer Programming》第一至第三卷及 N. Wirth 的名作《Algorithms＋Data Structures＝Programs》，吸取了这二部名著的营养。

对非计算机专业的人员来说，除了学习必要的理论知识外，程序设计技巧是至关重要的。如何学习程序设计技巧呢？一个很重要的方法是读已有的具有代表性的程序。作者在叙述数据结构的基本概念时，大量地给出了算法、框图和程序，这些程序是作者多年教学和科研中积累起来的，从程序设计技巧来讲，有一定代表性，通过程序也将介绍很多程序设计技巧；使读者能很快地熟悉概念，学会应用，提高程序设计能力。

数据结构对于非计算机专业人员来讲，似乎比较抽象，为此，作者愿意引用 N. Wirth 的一句话提示读者，即“数据结构存在于计算机程序中”。当你仔细阅读了本书的基本概念、算法、应用、框图和程序之后，定能体会 Wirth 教授的这句话，在你的 CAD 和其他计算机应用工作中，会很自然地应用你所学的数据结构知识。

作者借此机会对西安交通大学陶钟教授、蒋德明教授、汪应洛教授给予作者的鼓励和支持表示感谢；对西北工业大学徐秋元副教授认真审阅书稿，提出许多宝贵意见，使本书的质量得到很大提高表示感谢；对西安交通大学刘峻同志认真审阅全书，并对书稿进行编辑加工表示感谢。

限于作者水平，本书的不妥与错误之处在所难免，敬请读者指正。

作　者

1985 年 4 月

于西安交通大学

目　　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章 引 论

在使用计算机进行计算机辅助设计(CAD)*、计算机辅助制造(CAM)*、工业管理、生产过程监控、科学计算时，经常需要对大量的信息进行加工处理。在大多数情况下，这些信息并不是无组织的、杂乱无章的，它们之间含有数据之间的重要的结构关系。

要画一张平面工程图纸，实际上是把它分成了点、直线、圆、椭圆、双曲线、抛物线、三次曲线、折线和样条曲线，这些基本曲线被称为基本线素。一个线素通常有起点坐标、终点坐标、其它点坐标、线型、……等信息，如何对一个线素组织这些信息，如何把各种线素组合成一张平面图纸，这就是数据结构所应处理的问题。换句话说，数据结构是 CAD 与 CAM 的基础。

数据的最基本单位是数据项，英语一般用 Item 或 Field 表示，它说明对象的一种基本特性（或称属性）。

数据项的序列组成结点(node)，或者说一个结点可划分为一个个数据项，数据结构的基本单位是结点。

上述每个线素都以结点形式出现，而起点坐标、终点坐标、线型、……都是它的数据项。

通常是研究数据之间的逻辑关系(如前后关系，即哪个结点在前，哪个结点在后，研究结点之间的次序，这也叫"序关系"；又如"从属"关系，即哪个结点是父亲，哪个结点是儿子，或说一个结点从属于哪个结点；…)。对于不同的数据结构形式，由于不同的使用目的和方式，它们在计算机上的存贮分配也就不同，这种体现结点间逻辑关系的存贮分配叫做数据的物理结构。

数据结构主要是研究结点间的"序"关系、"层次"关系以及它们在计算机内部存贮和检索的实现方法。

要介绍数据结构，表达的方法很多，本书采用读者容易接受的自然语言，用细框图做总结，最后给出 PASCAL 语言的程序。这些程序是经过上机考验的，可直接供用户使用。

我们假定本书的读者学过诸如 BASIC、ALGOL 、FORTRAN 等算法语言之一；我们的对象是软件专业或非软件专业而将从事 CAD/CAM 专业的大学生或研究生、管理信息系统专业的大学生或研究生；并不要求读者预先了解 PASCAL 语言；然而，要求读者配合本书学习，进行上机练习，逐渐培养单独编写程序，独立上机的能力。如本书的每一个程序，每一个算法都上机，这就可望读者有较强的上机能力。作者曾对我校机械制造专业的 CAD 研究生及管理信息系统的研究生进行过试验，效果较好。

§1 PASCAL语言初步

一、程序结构

以"输入两个变量，计算两数之和，再输出"为例，看一个程序的结构：

* CAD 是 Computer Aided Design（计算机辅助设计）的缩写
CAM 是 Computer Aided Manufacturing（计算机辅助制造）的缩写。

```
PROGRAM SIMPLE (INPUT, OUTPUT);
      {THIS IS USED FOR S=X+Y}
      VAR X, Y, S: INTEGER;
      BEGIN READ (X, Y);
            S:=X+Y;
            WRITELN( 'S=', S)
      END.
```

从中可以看出，一个 PASCAL 程序分为三部分：

程序首部

程序说明部分

程序执行部分

(一) 程序首部

它由 PROGRAM（程序的标志）、SIMPLE（程序名称）、INPUT、OUTPUT（程序的参数）所组成。其中，程序名称实际上是标识符，变量名(X, Y, S)也是标识符，PASCAL 语言的标识符是字母、数字串，第一个字符必须是字母，最多可以保留八个字符有效。程序的参数用来表示该程序与外界的联系，它们一般是文件变量名，程序通过这些参数调用外部文件，最常用的程序参数为 INPUT、OUTPUT，它表示该程序具有输入、输出操作。

程序首部有时还带有程序的注释部分，如本例的 {THIS IS USED FOR S=X+Y}，一般用于注释：该程序的名称、类型、主要功能、编写日期等。

(二) 程序说明部分

PASCAL 程序允许用户自己定义标号、常量、变量、类型、过程和函数等，这些都必须首先在程序的说明部分加以说明，然后才能在程序的执行部分引用。程序的说明部分应遵循如下次序：

(1) 标号说明部分；

(2) 常量定义部分；

(3) 类型定义部分；

(4) 变量说明部分；

(5) 过程与函数说明部分。

以下予以逐个介绍。

1. 标号说明部分

在各种算法语言中，几乎都有转移语句，然而，PASCAL 语言为了程序动态结构和静态结构的一致，不提倡使用转移语句。但是，大部分用户习惯利用转移语句来改变程序的执行次序。无论是条件转移还是无条件转移，将要转到的语句一定要有语句标号。语句标号为整数，允许 1 到 9999，语句标号一般是由小到大排列，但可以不连续。PASCAL 语言规定，一个分程序的所有语句标号，必须在这个分程序首的标号说明部分予以说明，标号说明部分说明了的标号，一定要用。

一般形式为：

LABEL 标号表；

其中，标号表由一系列标号组成，标号之间用 "," 号隔开；我们也可以说语句标号是

无符号整数。标号的例子请见下一节。

2. 常量定义部分

PASCAL 定义了三个常量：FALSE、TRUE、MAXINT（计算机所能表达的最大整数）。PASCAL 语言允许程序员定义一些常量，用标识符表示。还规定：常量名与常量必须一一对应；一个常量名一经定义，不允许在程序的执行部分再对这个常量进行赋值；任何常定量义必须是单值的。它的一般形式为：

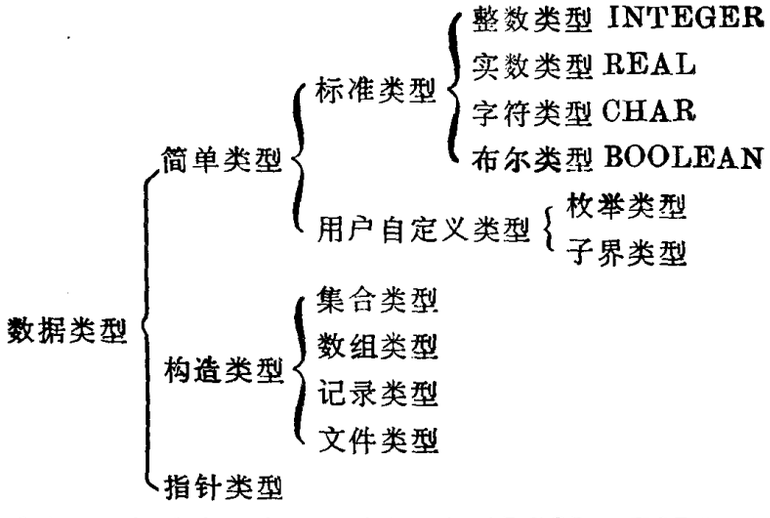
```
CONST
    标识符 1 =常量;
    标识符 2 =常量;
        .....
    标识符 n =常量;
```

例如：

```
CONST
        MAXNUM=10000;
        MINNUM=500;
        PI=3.1415926;
        PAGESIZE=60;
        ENDCHAR='$';
        A=35;
```

3. 简单数据类型

PASCAL 的数据类型可以图示如下：



这里只介绍标准类型，标准类型中的四种类型与 ALGOL、FORTRAN 的含义相似。

类型定义的一般形式为：

```
TYPE
    标识符 1 =类型;
    标识符 2 =类型;
        .....
    标识符N =类型;
```

4. 变量说明部分

任何一个有使用价值的程序，离不开设置变量，PASCAL 语言规定，一个分程序中所用的量变，必须在变量说明部分予以说明。PASCAL 不允许类似于 FORTRAN 语言的隐含说明，一切变量必须明显地加以说明；而且，一个变量只能属于一种确定的数据类型，即只能说明一次。例如：

```
VAR
    I, INDEX, NUMBEACH: INTEGER;
    ROOT1, ROOT2: REAL;
    CH1, CH2, CH3, ENDCHAR: CHAR;
    ENDOFPAT, OVERFLON: BOOLEAN:
```

二、PASCAL语言的语句类型

PASCAL 程序的执行部分主要有下述执行性语句所组成：

(一) 赋值语句

一般形式为：

变量：＝表达式；

其右端是一个表达式，它产生一个值，赋给左端的变量。

PASCAL 的算术表达式是变量与算术运算符的合法组合。其中算术运算符为：

＋、－、＊、DIV(整除)、MOD(求余)、/(实数除)。

它的规定如下：

(1) 加、减、乘是三种常用的运算，两个操作数都是整型数，结果是整型数；其中，若有一个是实数或两个都是实数，则结果为实数。

(2) 实数除不管操作数是实数还是整数，结果均为实数。整数除与求余的两个操作数必须是整数，结果一定是整数。

PASCAL 的布尔表达式是由布尔型变量和布尔（逻辑）运算符组成。其中布尔型变量可以由产生布尔型值的关系表达式代替，而关系表达式是算术表达式与关系运算符的合法组合。关系运算符主要有下列六种：

＝(判等)，＜＞(判不等)，＜(判小于)，＞(判大于)，

＜＝(判小于或等于)，＞＝(判大于或等于)。

若判定条件满足，则关系表达式取真(TRUE)值，否则取假(FALSE)值。布尔(逻辑)运算主要有下列三种：NOT, AND, OR 其运算法则为：

A	B	A AND B	A OR B	NOT A
TRUE	TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE

具体规定为：

(1) 数学上可以写成诸如 A≥B≥C≥D，PASCAL 语言必须写成：

(A>=B) AND (B>=C) AND (C>=D)；

(2) 逻辑运算符必须写成诸如 A AND B, A OR B, NOT A；

(3) 运算优先次序为，

(i) 圆括号，必须配对出现，配对处理，由内向外逐层展开；

(ii) NOT；

(iii) AND；

(iv) OR；

(v) *、/、DIV、MOD；

(vi) +、-；

(vii) =、< >、>、<、>=、<=；

(viii) 同一级从左到右顺序执行。

（二） 复合语句

它是由语句括号 BEGIN 及 END 括起来的几个语句所组成，形式如下所述：

```
BEGIN
    语句 1；
    语句 2；
      ……
    语句 n
END;
```

其中，任意一个语句本身还可以是复合语句；即 BEGIN,END 可以嵌套使用。

（三）如果语句

一般形式：

IF 布尔表达式 THEN 语句 1；

IF 布尔表达式 THEN 语句 1 ELSE 语句 2；

前者，布尔表达式的值为真时，做语句 1；否则，做下一个语句。后者，布尔表达式为真时，做语句 1，然后，做下一个语句；否则，布尔表达式为假，做语句 2，而后做下一个语句。

在如果语句中，若语句 1 或语句 2 又是一个如果语句，则称为复合 IF 语句，它可以写成下述形式：

```
IF 条件 1
  THEN IF 条件 2
          THEN IF 条件 3
                  THEN 语句 1
                  ELSE 语句 2
          ELSE 语句 3；
  ELSE 语句 4；
```

即：条件 1 =TRUE，且条件 2 =TRUE，且条件 3 =TRUE 时，做语句 1，而后，做这个复合语句 IF 的下一个语句；条件 1=TRUE 且条件 2=TRUE，条件 3=FALSE 时，做语句 2，而后做下一个语句；条件 1=TRUE 且条 件 2=FALSE 时，做语句 3，而后，做下一个语句；条件 1=FALSE 时，做语句 4，而后做下一个语句。为了避免引起如下形式的明显二义性（ELSE 属于哪一个 IF？）：

```
IF 条件 1
```


此为试读，需要完整PDF请访问：www.ertongbook.com

```
        THEN IF  条件2
                 THEN  语句1
                 ELSE  语句2;
```

规定ELSE子句属于最近没有ELSE子句的IF语句；即上述情况，规定ELSE子句属于IF条件2。如果上述ELSE子句确实属于IF条件1时，必须写一个空ELSE子句，即ELSE空语句；或改为：

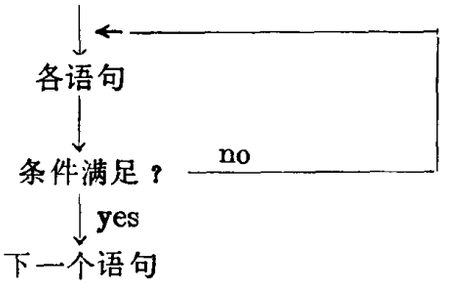
```
    IF  条件1
        THEN BEGIN
                 IF  条件2
                     THEN  语句1
             END
        ELSE  语句2;
```

（四） 重复语句

一般形式：

```
REPEAT
        各语句
UNTIL  条件;
```

它是重复执行的语句，条件不成立时，重复执行；条件成立时，终止重复，作下一个语句，它至少执行各语句一次。它的工作流程如下图：



举一例，计算 $H(n)=\sum_{i=1}^{n}1/j$ (n≥1)可写成：

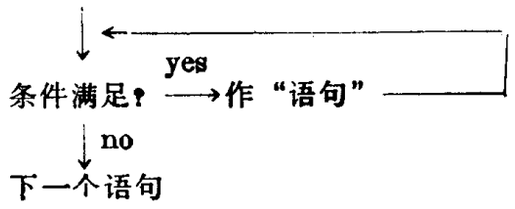
```
VAR I, N: INTEGER;
    H: REAL;
    H:=0;  I:=1
    REPEAT
      H: =H+1/I;  I:=I+1
    UNTIL I≥ N+1;
```

（五） 当语句

一般形式：

```
  WHILE  条件  DO
         语句;
```

它也是一个重复语句，它与REPEAT语句相反，当条件成立时，重复执行语句，条件不成立时，终止执行语句，作下一个语句。工作流程图如下图：



上例可以写成：

```
VAR I, N: INTEGER;
    H: REAL;
    I:=N;
    WHILE I>0 DO
      BEGIN
        H: =H+1/I;
        I:=I-1
      END;
```

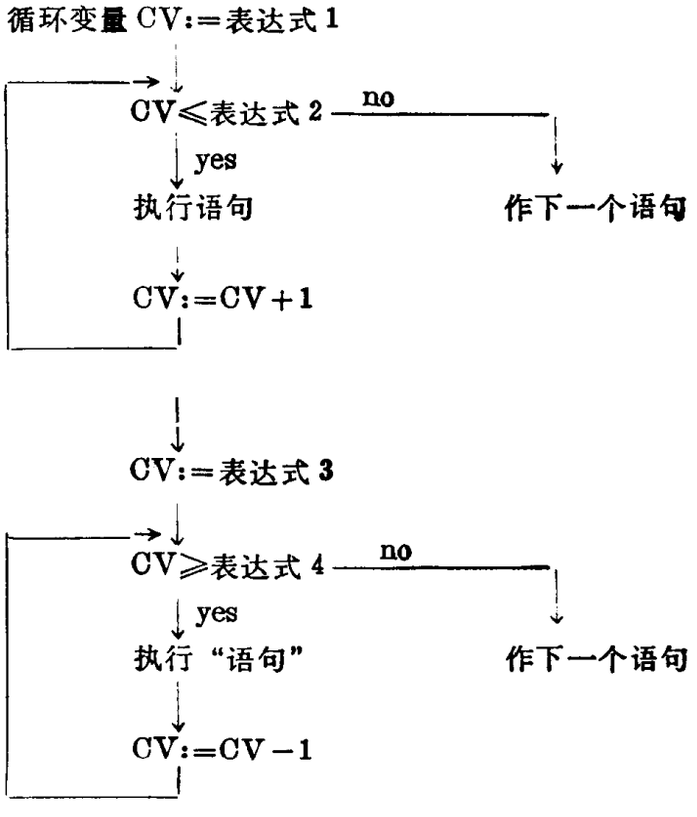
（六） 循环语句

一般形式：

FOR 循环变量：=表达式 1 TO（或 UPTO）表达式 2 DO 语句；

FOR 循环变量：=表达式 3 DOWNTO 表达式 4 DO 语句；

其中，前者要求表达式 1 ≤表达式 2，每次循环的步长为+1；后者要求表达式 3 ≥表达式 4，步长为−1。其工作流程分别为：



上例可以写成：

```
VAR I, N: INTEGER;
    H: REAL;
    H:=0;
    FOR I:=1 TO N DO H:=H+1/I;
    (或 FOR I:= N DOWNTO 1 DO H:=H+1/;)
```

同样若要重复执行 n 个语句：语句 1，语句 2，……，语句 n；上述三种重复语句可分别写成：

```
REPEAT      WHILE条件 Do      FOR CV:=表达式1 TO 表达式2 DO
 语句1;       BEGIN语句1;        BEGIN    语句1;
 语句2;              语句2;              语句2;
   ⋮                  ⋮                   ⋮
 语句n                语句n               语句n
UNTIL 条件;      END;               END;
```

（七） 情况语句

一般形式：

```
CASE 〈表达式〉 OF
    〈分情况标号表〉:〈语句1〉;
    〈分情况标号表〉:〈语句2〉;
            ..............
    〈分情况标号表n〉:〈语句n〉
END;
```

此处的分情况标号表由若干个标号（正整数及 0 或文字常数）组成，标号之间用逗号分开；这里的标号不是一般的标号，不能为任何转语句引用；故在有些 PASCAL 语言书上，情况语句写成：

```
CASE 〈表达式〉 OF
    〈常数表1〉:   〈语句1〉;
    〈常数表2〉:   〈语句2;
          ..........
    〈常数表n〉:   〈语句n〉
END;
```

其中，常数可以是 0 及正整数或文字常数。

例如：

```
VAR WEEK: INTEGER;
CASE WEEK OF
  0: WRITELN ('SUNDAY');
  1: WRITELN ('MONDAY');
  2: WRITELN ('TUESDAY');
  3: WRITELN ('WEDNESDAY');
  4: WRITELN ('THURSDAY');
```

```
        5: WRITELN ('FRIDAY');
        6: WRITELN ('SATURDAY')
    END;
```

其中，WRITELN 为打印语句，下面将详细介绍。

又如：

```
VAR I: INTEGER;
    CH: CHAR;
CASE I OF
    0: X:=0;
    1: X:=SIN(X);
    2: X:=COS(X);
    3: X:=EXP(X);
    4: X:=LN(X)
END;
```

有些PASCAL语言对情况语句有扩充，提供其它情况的执行方式，形如：

```
CASE 〈表达式〉 OF
        〈分情形标号表1〉:  〈语句1〉;
        〈分情形标号表2〉:  〈语句2〉;
            ..................
        〈分情形标号表n〉:  〈语句n〉
        ELSE 〈语句n+1〉
END;
```

（八） 转移语句

一般形式：

```
GOTO 语句标号;
```

PASCAL不提倡使用转移语句。

（九） 输入、输出语句

一般形式：

```
READ (〈变量表〉);
READLN (〈变量表〉);
WRITE (〈变量表〉);
WRITELN (〈变量表〉);
```

请注意其中的区别：

1. READ仅要求数据一个一个不断输入，并不要求换行。如果本行还有其它数据，下一个READ语句可以接着使用。READLN语句不仅要求一个一个数据接着输入，一旦本语句读完所要求的数据，则不管本行还剩下多少数据，都要跳到下一行去；如果还有READ或READLN语句的话，它要求从另一行中读取数据，原来一行数据不能再读。

2. READ语句必须至少输入一个数据；而READLN可以不输入任何数据，而只执行换行的要求。

3. WRITE语句是一项接一项的输出，但不换行；WRITELN语句是一项接一项输

出，输出完最后一项即自动换行。

4. WRITE 语句至少输出一项，WRITELN 允许不输出任何内容，仅仅换行。

举一例，求圆锥体的体积V，已知底圆半径R，高为H。

```
CONST  PI=3.1415926;
VAR  H, R, V: REAL;
BEGIN  READLN (R, H);
       V:=PI * SQR(R) * H;
       WRITELN ('V=', V)
END;
```

PASCAL 语言无格式场的概念，它规定：

整型数场宽 10 位

实型数场宽 22 位

布尔型数场宽 10 位

ALFA 型数场宽 10 位

字符型场宽 1 位

字符串型场宽为字符串长度。

其中 ALFA 型和文字串以后再介绍。

三、用户自定义的数据类型

(一) 枚举类型：

一般形式为：

TYPE‹类型标识符›=(‹标识符 1›, …‹标识符 n›)；

可以说枚举类型是常量（可以是整数、实数、文字常数、逻辑型常数）集合的标识；但是这里不直接写出常量，而是代之以与其对应的标识符。一个枚举类型，必须明确列出该类型所包含的全部数据，即全部对应的标识符。在一个类型的全部数据之间，建立一个“序”关系：对每个数据能回答：它是第几个数据？它的前一个及后一个数据分别是什么？一共可以有多少个数据。对枚举类型的数据可以依次寻找给定的数据。有几个枚举类型时，一个数据只能属于一个枚举类型，且不允许不同的类型之间赋值。

例如： type(*)

```
        color=(while, red, orange, yellow, green, blue. purple,
               black);
        day=(monday, tuesday, wednesday, thursday, friday,
             saturday, sunday);
        units=(inches, feet, furlongs, miles);
var
        workday, rest, holiday: day;
        scale: units;
```

pascal 语言允许枚举类型定义与变量说明加以合并，

例如： var workday, holiday, rest: (mon, tues, wednes, thurs, fri, satur, sun)

* PASCAL 语言的源程序可以全部用小写，也可以全部用大写，两者等价（以下同）

scale, (inches, feet, furlongs, miles);

对诸如 color 等枚举类型数据，可以有 succ(x)(求后继数据)，pred(x)(求前超数据)，ord(x)(求数据的序号)等运算。例如：

succ(blue)=purple;

pred(blue)=green;

ord(blue)=5;（序号从 0 开始）

注意：枚举类型不允许直接通过读语句输入，也不允许通过写语句输出，它可以作为条件语句的条件，重复性语句(包括重复语句、当语句、循环语句)的控制变量等。

例如： while workday<=saturday do语句;

for scale:=inches to miles do 语句；等。

(二) 子界类型

一般形式为：

TYPE 〈类型标识符〉=〈常量 1〉…〈常量 2〉

这里常量 1＜常量 2，常量 1，常量 2 分别称为下界和上界。

这里类型标识符由用户自己确定，常量 1 与常量 2 必须是两个属于同一类型的有序数据，一般是整数，字符或已经定义为枚举类型的数据。实数因有稠密性，不能用来构造子界类型。

例如：

```
TYPE
DAY=(SUN, MON, TUE, WED, THU, FRI, SAT);
YEAR=1981…2000;
DIGIT= '0' … '9' ;
LETTER='A'…'Z';
WORKDAY=MON…SAT;
VAR
  A: 1…10;
  B: 0… 30;
  C: 20…30;
```

能对上述子界类型 A、B、C 进行运算与赋值，但要防止计算结果超出给定的范围。例如：

B:=A+5; C:=A+20;

注意，对子界类型求序号 (ord) 时，如给出的是相应枚举类型的数据，则序号为原来枚举类型中的序号。同样不允许对子界类型变量进行输入和输出。

四、数组类型

PASCAL 语言的数组类型是由固定数量的同类型(这个类型称为基类型)的元素组成。数组元素的数量不允许变动；元素的类型必须相同，不允许混合；这是数组的两大特点。每个数组元素通过数组变量的名字跟以通常所说的下标来直接访问。下标是可计算的，其类型称为下标类型。这种访问是完全随意的，故数组被称为随机访问结构。

一个数组的定义既规定元素类型，又规定下标类型，其一般形式为：

TYPE 类型名：ARRAY [T_1] OF T_2;

T_1 为下标类型，T_2 为任意数据类型，也可以是构造型数据类型。注意：T_1 是枚举类型、子界类型、字符类型之一。

此为试读，需要完整PDF请访问：www.ertongbook.com