Pascal 语言程序设计教程

郭浩志 编著

国防科技大学出版社

Pascal

语言程序设计

PASCAL 语言程序设计教程

（修订版）

郭浩志　郭旭东　编著

国防科技大学出版社
·长沙·

图书在版编目(CIP)数据

PASCAL 语言程序设计教程/郭浩志,郭旭东编著.—长沙:国防科技大学出版社,1999.3

ISBN 7-81024-528-7

Ⅰ PASCAL 语言程序设计教程

Ⅱ 郭浩志　郭旭东

Ⅲ ①计算机②PASCAL 语言③程序设计

Ⅳ TP3

国防科技大学出版社出版发行

电话:(0731)4555681　邮政编码:410073

E-mail:gfkdcbs @ public.cs.hn.cn

责任编辑:文　慧　责任校对:黄　煌

新华书店总店北京发行所经销

国防科技大学印刷厂印装

787×1092　1/16　印张:20　字数:462 千

1999 年 3 月第 2 版 2000 年 3 月第 2 次印刷　印数:3001—8000

*

定价:24.00 元

前　言

Pascal 语言是第一个体现结构程序设计思想的高级语言。1968 年它由著名计算机科学家 N. Wirth 教授在瑞士苏黎世联邦工业大学讲授程序设计时提出，1973 年正式发表标准 Pascal 语言文本。它是在总结了几种人们最乐意使用的高级语言（尤指 ALGOL—60 和 ALGOL—W），并吸取了从 60 年代末开始提出的计算机科学若干新概念（如结构程序设计、数据类型定义、程序正确性等等）的基础上发展起来的。由于它自然、方便、简洁、可靠、高效、功能齐全、易于移植，很快就流行于世界，被誉为程序语言发展到 70 年代的一个里程碑。如今，它不仅成为国内外越来越多理工科大学的必修课程，而且在计算机系统软件和其它领域中得到日益广泛的应用。对于进一步学习目前最完善的面向过程的语言 ADA，先学习 Pascal 语言是必需的。

本教程是作者总结多年教学科研实践经验的结果。本书初稿定于 70 年代后期。由于教学需要，此次出版前又作了局部变动，充实了若干章节。

全书突出结构程序设计思想，共分十四章。第一章引论，从一个简单例子出发，引进了 Pascal 语言源程序结构、程序设计风格和 BNF 规则等基本内容；第二章叙述四种标准类型；第三、四、五章按结构程序设计概念依次介绍了顺序结构、选择结构和重复结构等三种基本控制结构；第六、十两章则进一步讨论了结构化程序的组成单位——模块（Pascal 语言的函数和过程）；第七、八、九、十一、十二和十三章分别谈了 Pascal 语言的重要特色——自定义数据类型：枚举、子界、集合、数组、记录、文件和指针；第三至六章突出语句结构化，而第七至十三章则强调数据结构化；第十四章是全书的总结与提高，比较详细地讲述了适合 PASCAL 语言的程序设计方法——自顶向下逐步求精法，并介绍如何利用这一方法解决 Pascal 程序的编制与调试问题。

全书力求做到：

（1）以标准 Pascal 语言为背景，全面、系统、深入地讲述该语言各种成分的语法、语义和语用；

（2）叙述清楚，重点突出，概念引进准确自然，层次组织合理分明。文字流畅、简洁、适于自学；

（3）强调程序设计风格，并贯穿于全书示例；

（4）数值、非数值并重，以非数值为主；

（5）联系实际，为学生将来开发软件提供一些典型问题的处理模型和方法；

（6）适应不同教学和不同程度学生的要求。目录中凡节号前打“*”号的内容可供选学。课内学时为 30 至 50 小时；

（7）每章最后各附有多个练习题。题目形式多样，难易兼顾，部分选自近几年我校计算机专业本科生试题以及研究生入学试题。前十三章中部分题可兼作阶段性实习题，第十四章多数题可作为综合性实习题。

在本书编写过程中，得到了我校计算机专业许多教师、学生的关心和支持。在此出版之际，一并表示衷心的感谢！

郭浩志

1987.9

第一章 引 论

1.1 基本概念

1.1.1 算法与程序

我们知道，对于绝大多数计算机来说，为了求解一个问题，必须先对问题进行分析，提出解决问题的办法，然后建立解决此问题的计算步骤。如果一个问题找不出算法，甚至对问题尚无法确切定义，而只有一些想象或猜测，这样的问题是无法在计算机上求解的。所谓算法，粗略地说，就是解题方法的精确描述，它与计算机没有必然的关系。我们可以用若干种方法来描述算法。

(1) 文字描述。即用自然语言(如汉语、英语)描述。其特点是易读易理解。

(2) 框图语言。框图是对算法逻辑顺序的图形描述。如用长方形框表示计算公式，用菱形框表示条件判断……此方法形象、清晰、画法简单、格式自由，可以不涉及太多的机器细节或程序细节，其主要弱点是计算机很难直接识别。

(3) 程序语言。所谓程序是为描述某一问题的步骤而设计的一系列命令。这种命令在高级语言中即为语句，在机器语言中即为机器指令(在汇编语言中即为符号化的机器指令)。它们都是算法在计算机系统中的具体表示，但各有不同特点。机器语言或汇编语言是面向机器的语言，可以把机器的特征和灵活性充分发挥出来，其严重缺点是使用极其不便。因此，30 多年来，不少软件工作者都致力于高级语言和编译程序的研究。高级语言对用户来说方便了许多，但计算机软件工作者必须为系统配置相应的编译程序。编译程序是一种计算机系统程序，负责将用高级语言书写的源程序转换为机器可识别的代码程序。

1.1.2 语法、语义与语用

在以后章节中对每一种语言成分都将着重从三个方面讨论 Pascal 语言。一个是语法，它是语言正确成分的各种构造规则；另一个是语义，它给出语言成分含义的解释；第三个是语用，它说明了语言成分的使用方法和用处。正确了解一个语言的语法、语义和语用，不仅对于正确运用语言描述问题和实现编译程序是必须的，而且对于进一步研究有关语言的理论和计算机科学也是必不可少的。

1.2 从一个简单例子谈起

例 1.1 输入两个整数到 a 和 b，计算其和，放入 s 中，最后输出之。

我们先用汉语描述此问题的算法核心。据题意，可按时间先后顺序给出执行逻辑如

下：

- 读入 a 和 b；
- 计算 a 与 b 之和，存入 s；
- 输出 s.

还可用框图语言描述，如图 1.1 所示。

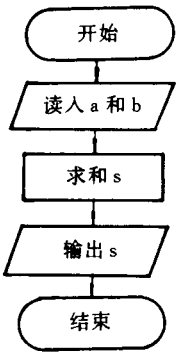

图 1.1　简单例子的框图

这里基本图形及其含义遵循国家标准，详见文献[15]。

若改用非形式 Pascal 语言描述，则成为下列形式的程序段：

```
BEGIN
  read(a,b);
  s :=a+b;
  writeln(s)
END
```

初看上去，上列描述有点像英语句子。整个程序段与框图对应。BEGIN 标志该程序段的开始，END 标志该程序段的结束。由 BEGIN 与 END 括住的三行代表三个语句，是该程序段的核心。其中第一句称为输入语句，表示从输入设备（例如键盘、磁盘）输入两个数，依次存入 a 与 b 中。第二句称为赋值语句，表示将 a 与 b 之和赋给 s. 第三句称为输出语句，表示将 s 输出到外界。输出设备可以是行打印机、屏幕显示器、磁盘等等。

有经验的程序员，为了在程序运行过程中及时提醒用户输入 a 和 b，往往在输入语句之前先输出一些提示信息（习惯上在屏幕上显示）。譬如，我们可以安排下列输出语句：

```
writeln('please input!')
```

执行上列语句时，会把两引号之间的字符串从屏幕上原样显示出来：

```
please input!
```

即通知上机人员准备输入。

为了检验输入数据是否正确，最简单的办法是在输入语句之后立即将该数据输出。譬如，我们可以安排下列输出语句：

```
write(a,b)
```

执行上面这个输出语句，只输出 a 和 b 的内容，无回车换行动作。之后，当执行 writeln(s) 时，将 s 之值输出在 a,b 之右边，然后再回车换行，使 a，b，s 在同一行输出。为了使 a，b，s 之间的关系显得更清晰更明确，在输出 a 之后安排一个加号“+”，输出 b 之后安排一个等号“=”，即将上面的输出语句改为：

```
write(a,'+',b,'=')
```

例如，当 a=1,b=2时，它输出：

```
1+2=
```

以上用若干语句描述的仅仅是算法的核心，它并不是一个可以交付机器执行的完整的 Pascal 程序。其中出现的 a，b，s 诸变量是什么类型的数据，我们尚未指明。Pascal 语言规定，凡是程序中使用的变量一律要预先说明。对于上面这个问题，应在诸语句之前加上如下的说明语句：

```
VAR   a, b, s: integer;
```

它表示:变量a, b, s都是整型的。VAR为保留字,表征变量说明。

再考虑程序的头和尾,就可把上面描述的算法整理为一个完整的Pascal程序:

```
PROGRAM addition(input,output);
  VAR a,b,s:integer;
BEGIN
   writeln('please input!');
   read(a, b);
   write(a,'+',b,'=');
   s :=a+b;
   writeln(s)
END.
```

以上程序的第一个字PROGRAM是保留字,表示程序的开始;接下来的addition为程序名。圆括号中的input参数表明该程序要求从外界输入一些数据,而由read语句实际执行这个请求。output参数表明该程序要求产生一些结果,而由write与writeln语句实际执行有关输出。

上列程序进入机器后,一旦启动执行,用户即应按提示输入两个整数给a和b,譬如将"1"给a,将"2"给b. 此时运行过程的显示信息为:

```
please input!
1+2=3
```

1.3 Pascal源程序结构

一个用Pascal语言描述的程序称为Pascal源程序。例如上一节给出的addition. 用户编写的Pascal源程序,由计算机系统中一个专门软件——Pascal编译程序将它翻译成机器能识别的代码,以便机器执行,产生结果。

源程序主要描写对数据的处理,它包括两个方面。一个是处理动作,另一个是处理对象,即数据。同很多其它高级语言一样,Pascal的动作由"语句"描述,数据由"说明"定义。

一个Pascal源程序由三部分组成,依次为程序头、程序体和程序尾。

程序头必须是Pascal源程序的第一行,自左至右依次包括保留字PROGRAM、程序名和程序的若干参数。后者指出程序中使用的文件名字,它们用一对圆括弧括起来,相互之间用逗号分隔。程序名由用户任意选定,一般采用标识符,如上节例子的程序名为addition. 所谓文件,简单地说,就是一组数据(信息),借助它可沟通程序与外界环境之间的信息传递通道。例如上例中的input和output就是常见的输入输出文件,分别对应着输入、输出设备。可见,程序总要使用文件,以存取原始信息、中间结果和最终结果。

程序体包括两部分:说明部分和执行部分。Pascal规定,说明部分在前,执行部分在后。说明部分最多可以包括五种说明,但具体在一个程序中均可有可无。如果有说明部分,则必须严格遵照下列次序书写:

标号说明；
常量说明；
类型说明；
变量说明；
子程序说明。子程序说明包括过程说明和函数说明，两者可以交叉出现。

前例只包含变量说明。Pascal 语言定义了一些标准的常量、类型、过程和函数，可以不加说明就在执行部分直接使用。执行部分以保留字 BEGIN 打头，END 押尾，中间包括一系列语句，用以描述问题算法的核心。各语句之间用分号“；”分隔。例如，前例的程序体包括了三个语句。

程序尾仅由一个句号字符“.”组成。它必须位于最后一个 END 之后。

1.4 程序设计风格

源程序的设计方法以及它的书写格式、标识符的选择等等，几乎都可因人而异。但是，人们在长期的程序实践中，已逐渐体会到“读”程序在很多方面远比“写”程序影响大，而且设计方法的优劣直接影响到程序的正确性。

程序设计风格指的是，借助好的设计方法编写结构好的程序；在编写源程序时，往往采用种种措施提高其可读性、可理解性、可修改性等，以利于程序的查错、调试、维护修改和学习交流。

当今，程序设计风格已成为程序人员的基本素养之一，对程序员进行这方面的训练得到学校和计算机软件工作者越来越多的重视。甚至一些早期语言（如 Fortran，Basic，Cobol 等）的新教科书也一改过去的传统，开始重视程序设计风格。这里，我们针对 Pascal 语言，提出几种常用的方法。

1. 缩进规则（锯齿规则）

对于编译程序来说，源程序只不过是一串字符，在书面上只要说明语句各部分的顺序不变，怎么安排都可以。我们完全可以将1.2节的整个源程序改写为：

```
PROGRAM addition (input, output); VAR a, b, s: integer; BEGIN writeln ('please input!');read(a, b); write(a,'+',b,'='); s:=a+b; writeln(s) END.
```

这丝毫不影响编译结果。但对于阅读来说，两种写法的效果就很不相同了。1.2节的程序采用了所谓缩进规则，使程序的书写格式能较好地反映出该程序的层次结构。该规则要求处于同一层的语句都从同一个字符位置开始书写。例如，1.2节程序处于最外层的程序头和程序体的第一个字符均在同一个字符位置开始。内层语句第一个字符则相对于紧外层语句向右缩进两个字符位置，如前例中变量说明和执行部分的语句均相对于外层的 PROGRAM 缩进两个字符位置。我们在以后章节将会看到，程序层次更多时，这种规则带来的优越性将会更明显。

2. 英文符号名

符号名可由用户任意选择，但最好能反映有关功能和特征，便于见名识意。例如，1.2节的程序名选为 addition 或缩写的 add，就突出了主题。存放和值的变量名选择为 sum 或

s，就比较醒目。

同样，程序中反复使用的常量，宜在常量说明中起一个有助于记忆的名字，以利阅读和修改。例如给出下列常量说明：

```
CONST
  one=1;
  delta=0.05;
```

就可在程序执行部分用 one 代替“1”，用 delta 代替“0.05”。

3. 注解

注解是一个增加可读性、可理解性的常用措施。Pascal 允许在源程序任意适当地方插入注解。办法是，用一对花括弧分别作为注解的起止符号，其间可安排合适的词句、短语或数学公式，以提示读者或供自己备忘之用。注解可位于程序头或程序尾之后，子程序头或子程序尾之后，也可插在任一语句之前或之后，占一行或连续多行。注解仅起解释性作用，编译程序不予处理。只要设备允许，可随源程序正文一起输入或输出。此外，一对花括弧可分别用“(*”和“*)”替代。

例如，我们可对1.2节的源程序加上两条注解：

```
PROGRAM addition(input,output);
  VAR a, b, s: integer;
BEGIN{this program is used for s=a+b.}
  writeln('please input!');
  read(a,b);
  write(a,'+',b,'=');
  s :=a+b;
  writeln(s)
END. {of addition}
```

4. 大小写英文字母

除了引号间的字母外，编译程序并不区分英文字母的大小写。程序员完全可以自行规定用大小写字母表示不同特点的对象，以求清晰。今后，我们在本书中用大写字母表示保留字，而用小写字母表示其它对象。addition 程序就是按照这个规定书写的。

5. 输出信息

利用输出语句尽量将输出信息组织得直观清晰，布局合理，使之形象化、表格化、页面化、自动成文，便于他人看懂和存档。addition 程序的输出就考虑了这点。

6. 程序设计方法

自顶向下逐步求精法是比较适用于 Pascal 的一种结构程序设计方法。本书就是力图紧扣结构程序设计思想进行讲述的，并且在第十四章作了总结和提高。

程序设计风格还可表现在很多方面，我们将结合后续章节陆续介绍。

*1.5 BNF 与语法图

Pascal 语言的语法规则，可以采用多种方式来精确描述。常见的是巴科斯范式(BNF)和语法图。

1.5.1 BNF 及其派生规则

任一语言都含有大量成分(如标识符、表达式)。每一语言成分的语法结构均可用 BNF 的一条产生规则表示。

先介绍两个概念：终结符和非终结符。终结符指语言字符集的基本字符。非终结符代表某个语法成分，并具有确定的语法概念，如标识符、无符号整数、算术表达式、赋值语句、过程等等。

BNF 规则只用到两种基本符号：一个为"::="，表示"定义为"或"可以是"，另一个为"|"，表示"或"。每条规则均包含且仅包含一个"::="，它将规则分为两部分：其左边是一个拟定义的非终结符，右边是由非终结符和终结符或其中之一组成的一个任意符号串。凡非终结符均用一对尖括号括住。例如：

〈字母〉::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

〈数字〉::=0|1

定义了两个语法成分。〈字母〉这个语法成分定义为 A 或 B 或 C……或 Z，亦即26个大写英文字母之一。〈数字〉可以是0或1，即定义为二进制数字。如果要同时定义二进制数字和十进制数字，那么可以这样定义：

〈二进制数字〉::=0|1

〈十进制数字〉::=0|1|2|3|4|5|6|7|8|9

右边含"|"的产生规则，实际上是左边具有同样非终结符的若干条产生规则的合成写法。例如，〈二进制数字〉可以写成两条等价的规则：

〈二进制数字〉::=0

〈二进制数字〉::=1

但通常采用含"|"的形式，显得更为简洁。

产生规则的右边不仅可以出现终结符，而且还可出现其它已定义的非终结符，例如

〈数字〉::=〈二进制数字〉|〈十进制数字〉

产生规则的右边甚至还可以用正在定义的左边非终结符自身来定义，称为产生规则的递归定义。例如

〈标识符〉::=〈字母〉|〈标识符〉〈字母〉|〈标识符〉〈数字〉

将"标识符"定义为以字母打头的字母数字串。又如

〈无符号数字〉::=〈数字〉|〈无符号整数〉〈数字〉

将"无符号整数"定义为一串数字。

自从 ALGOL—60 语言文本首次采用 BNF 规则以来，已得到软件界的广泛注意和推广使用。同时又出现了 BNF 的若干派生规则。这里介绍其中常见的一种。

此派生规则比传统的BNF规则又增加了一种符号：

{ }表示花括弧中的终结符和非终结符在语法成分中出现n次(n≥0)。

例如：

〈标识符〉::=〈字母〉{〈字母〉|〈数字〉}

〈无符号整数〉::=〈数字〉{〈数字〉}

可见，用派生规则定义语法成分有时比传统的BNF更为清晰简洁。

1.5.2 语法图

对于产生规则右边各种可能的选择，还可通过图形形象地描绘。这种图形称为语法图。例如，“标识符”、“无符号整数”和“无符号常量”的语法图，如图1.2所示。

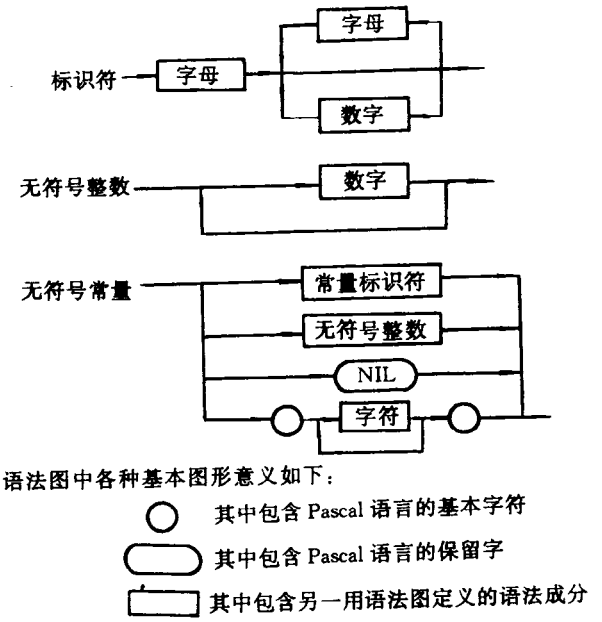

图1.2 语法图示例

练 习

模仿1.2节例子，编写下列诸题的程序，并分析源程序结构，和指出在哪些地方讲究了程序设计风格。

1. 输入两个实数作为长和宽，计算所组成之长方形的面积。

2. 输入一个实数作为半径，计算相应圆周长和圆面积。

3. 输入一个实数作为某正方体的棱，计算其体积、表面积、侧面积和对角线。

4. 将1.2节例子作下列修改：

(1) 将加法改为乘法；

(2) 当做完一次乘法运算之后，程序还能继续执行。

5. 本章1.2节程序可看作是一个最简单的计算机辅助教学程序：用户为老师(提出原始数据)，计算机为学生(给出答案)。现在，反过来，计算机为老师：出题，接收用户输入之答案并判断正误，打分和给出评语，而用户为学生(给出答案)。程序中应考虑哪些问题？

第二章　类型、常量与变量

2.1　基 本 元 素

2.1.1　基本字符集

Pascal 语言的基本字符集共包含56个字符，其中：

·英文字母共26个；

·阿拉伯数字共10个；

·其它字符共20个，它们是＋、－、＊、/、＝、＜、＞、(、)、[、]、{、}、'、，、.、；、：、↑、␣

2.1.2　基本词汇(单词、符号)

基本词汇可以由单个基本字符，也可以由两个或两个以上的字符组成。除上述字母和数字外，基本词汇还有60个特殊符号。其中：

·算术运算符共4个，它们是＋、－、＊、/

·关系运算符共6个，它们是＝、＜＞、＜＝、＜、＞＝、＞

·括号共7个，它们是(、)、[、]、{、}、'

·标点符号共5个，它们是，、.、；、：、‥

·赋值号1个，即：＝

·指针号1个，即↑(或^)

·空格符1个，即␣(打印时不印出)

保留字共35个，如 BEGIN，END 等等。

2.1.3　标识符

以字母打头的字母、数字串称为标识符。在语法定义中其字符个数不限。但在具体机器上实现时总有个数限制，例如8个、12个等等。标识符主要用作变量、常量、类型、函数、过程、程序、参数等的符号名。如 addition，s 等。

除了用户自定义标识符外，Pascal 语言自身还规定了两种有固定意义的标识符。

一种是保留字，均用英文字表示。例如 PROGRAM，END 等。语言赋予每一个保留字以特定的语法意义，不许用户挪作他用。

另一种为标准标识符，也有特定的意义，在程序中不经说明便可使用。但允许用户重新定义，另作他用。此时，该标识符将丧失原标准标识符的意义。为减少混淆，建议尽量不要这样使用。标准标识符共40个，在以后章节中将陆续介绍。其中：

·常量标准标识符共3个。如 maxint 表示计算机所能表示的最大整数，该值依赖于具体机器的实现。

·类型标准标识符共5个。如 integer 表示整型。

·文件标准标识符共2个。如 input 表示输入文件。

·函数标准标识符共17个。如 sin 表示正弦函数。

·过程标准标识符共13个。如 read 表示输入过程。

2.1.4 分隔符

在任意两个相邻的标识符、数或单词之间，至少必须嵌入一个分隔符，即空格符、行结束符或注解。注意，如果在一个标识符、数或单词的内部出现分隔符，则导致错误。

2.2 数据及其类型

2.2.1 数据

计算机是一个信息(数据)处理系统。我们把能输入到计算机并由计算机进行处理的对象称为数据。面向裸机(没有任何软件的计算机)的数据，一般以二进制数字表示，而面向虚拟机(配上系统软件，如 Pascal 编译程序的计算机)的数据就可人为地(如按照 Pascal 语言的语法规则)作一些约定，以适应各种应用。

按照其值在程序运行过程中能否变化，可将数据分为两种：常量和变量。其值在源程序中明确指定，在程序运行之前已经知道，且在程序运行过程中不再变化的量称为常量。在程序运行过程中，其值可在一定范围内变化的量称为变量。

2.2.2 类型的概念

类型表示值的集合和施加于这些值上的运算的集合。例如，整型定义了全部整数，以及施加于整数值上的四则运算。Pascal 语言每一个常量、变量、函数或表达式之值的类型都是唯一的。

常量、变量、函数或表达式之值必是其类型所定义值集合中的一个。例如，整型常量"1"为整型所定义的一个值。变量的类型确定了该变量的取值范围，它在程序运行过程中可获得其类型定义的任一值。

一般说来，不同类型所定义的运算是各不相同的。但对大多数类型(个别类型除外)来说均有两种运算。它们是：

(1) 赋值运算；

(2) 等同比较运算。

类型定义的运算，还可以通过定义函数和过程来加以扩充。除了标准函数和标准过程这些 Pascal 语言预定义的运算外，还允许用户在程序中根据需要自行定义该类型的其它函数和过程。程序中调用标准函数的一般形式为：

〈标准函数名〉(〈自变量〉)

其中，标准函数名采用规定的标准标识符。自变量只允许一个，它可以是变量、常量、函数甚至是一般表达式。每个标准函数及其自变量的类型和取值范围均有一定的限制。

凡是其值能按照一定顺序排列的类型称为有序类型。有序类型的每个值各对应一个序号，此序号用整数表示。整型、布尔型、字符型、枚举类型和子界类型均为有序类型，而实型则不是有序类型。

2.2.3 类型的分类

Pascal 语言有丰富的数据类型。共分三类：

1. 非构造类型

非构造类型又称为简单类型、纯量类型或标量类型。它的构成不再取决于其它类型。分为两种：

(1) 标准类型

整型(类型名为 integer)；

实型(类型名为 real)；

布尔型(类型名为 boolean)；

字符型(类型名为 char)。

(2) 用户自定义类型

枚举类型；

子界类型。

2. 构造类型

构造类型一般由其它数据类型构造而成。分为两种：

(1) 标准类型

标准类型只有一种，即正文文件类型。类型名为 text.

(2) 用户自定义类型

集合类型；

数组类型；

记录类型；

文件类型。

3. 动态类型

动态类型只有指针类型一种，它属于简单类型。借助于它定义的变量是动态变量，即可在程序运行过程中生成和释放的变量。

本章只讨论非构造类型中的四种标准类型。其余在后续章节中分别讲述。

2.3 整　型

2.3.1 整数

整型的数据——整数由带符号“＋”或“－”(其中“＋”号可省略)的十进制数字串或零组成。数学中的整数有无限多个。Pascal 中的整数数目依赖于具体计算机，是数学整数的一个子集。标准常量 maxint 是一台计算机能表示的最大整数。譬如，对于用16位二进制补码表示的带符号整数，maxint＝32767. 整数取值范围为：

[－maxint－1，maxint]

程序运行中，若整数大于 maxint，则产生溢出错误。如阶乘、乘方等运算时就很容易溢出。

整型为有序类型，每个整数的序号就是它自己。

2.3.2 运算

整型定义了五种算术运算，即整数加(＋)，整数减(－)，整数乘(＊)，整数除(DIV)和模运算(MOD)。其中 DIV 和 MOD 均为保留字。“＋”、“－”可用作一元或二元运算符，其余三种只能用作二元运算符。“＋”、“－”、“＊”均有公认的意义。DIV 表示整除后的商，结果为不经舍入的整数。MOD 表示整除后的余数。

设 a 和 b 均为整型数据，MOD 的定义为：

a MOD b＝a－(a DIV b)＊b

且余数的符号与被除数的相同。例如：

1 DIV 2＝0

7 DIV 2＝3

(－8) MOD 3＝－2

(－8) MOD (－3)＝－2

8 MOD (－3)＝2

六种关系运算符均适用于整数，但要求两个操作数均是整型的。运算结果为一布尔值。如

5＞2　　结果为 true

－1＜－2　结果为 false

2.3.3 标准函数

自变量和函数值都是整数的标准函数，共有五个：

1. 前导函数 pred(predecessor)

函数值 pred(X)表示比自变量 X 的值小1的整数(如果存在的话)，即

pred(X)＝X－1

例如　pred(－3)＝－4