HOPE

5.0 版

TURBO PASCAL 参考手册

（下）

H

北京希望电脑公司

TURBO PASCAL

参 考 手 册

（V5.0版）

下册

北京希望电脑公司

一九九一年七月

目　　录

第一章　词语和常量

Pascal程序中，词是最小的有意义的单位。词包括特殊符、标识符、标号、数字和字符串。

Pascal程序是由词和分隔符组成的。分隔符是空格或注释。如果词是一个保留字、标号、标识符或是一个数字，则两个相邻的词必须用一个或多个分隔符分开。除字符串常数外，分隔符不能作为词的组成部分。

1.1 特殊符和保留字

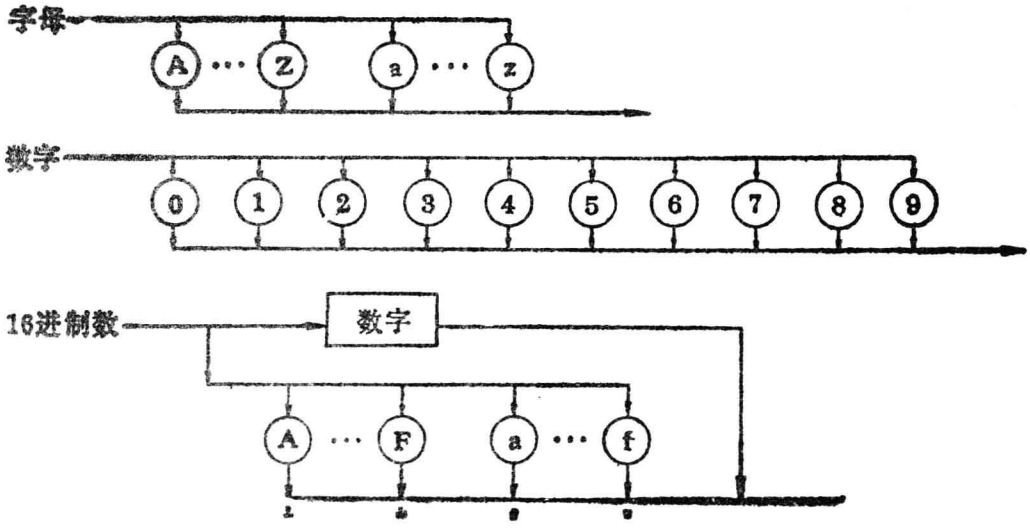
Turbo Pascal使用的是ASCII码：

(1) 字母—英文字母：A—Z和a—z；

(2) 数字—阿拉伯数字：0—9；

(3) 16进制数—阿拉伯数字0—9，字母A—F和a—f；

(4) 空格—空格符（ASCII码为32）和所有ASCII控制符（ASCII码从0到31）。其中包括行结束指示符，即回车符（其ASCII码为13）。



特殊符保留字是一个或多个有固定意义的字符，下面是特殊符：

＋－＊　／　＝　＜　＞　〔　〕　．　，　（　）　：　；　＾　@　｛　｝　$　#

下面的字符也是特殊符：

＜＝　＞＝　：＝　．．　（＊＊）　（．　．）

有些特殊符也是操作符。左括号〔与（．是等价的。同样右括号〕与．）也是等价的。

下面是Turbo Pascal的保留字：

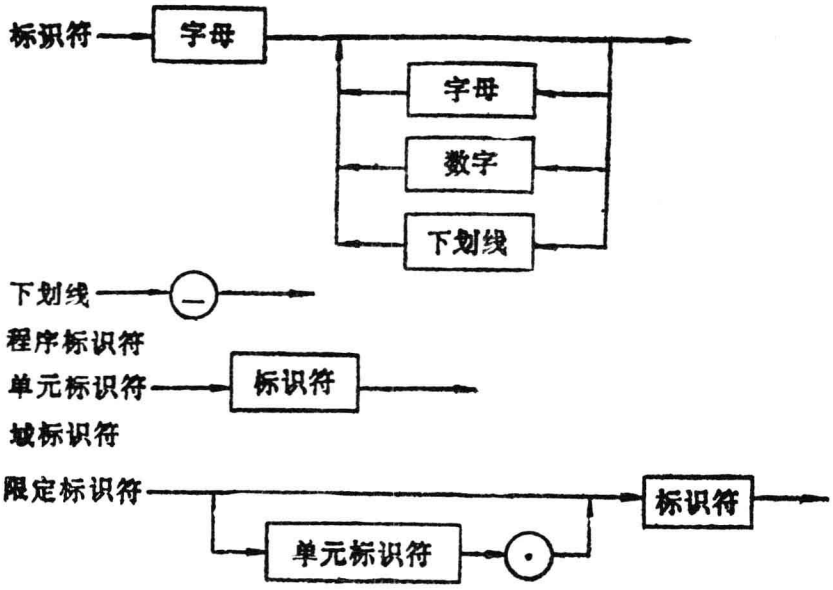
absolute	end	inline	procedure	type
and	external	interface	program	unit
array	file	interrupt	record	until
begin	for	label	repeat	uses
case	forward	mod	set	var
const	function	nil	shl	while
div	goto	not	shr	with
do	if	of	string	xor
downto	implementation	or	then	
else	in	packed	to	

1.2 标识符

标识符可表示常量、类型、变量、过程、函数、单元、程序及记录的域，标识符的长度不限，但只有前63个字符有意义。

下面一些框图是Turbo Pascal的语法图。从箭头开始读框图，从左边开始到右边以一个箭头结束的路径是合法的。

方框中的名字表示实际的结构。圆圈中的保留字、操作符及标点符号是程序中实际使用的项。



标识符必须用字母打头，中间不允许有空格。字母后可跟数字、字母和下划线的任意组合，和保留字一样，大小写字母作用相同。

当同一标识符出现多次时，必须用一个标识符来限定此标识符的含义。例如：用单元标识符 UnitName来限定标识符Ident，可以写UnitName.Ident。这类标识符叫做限定标识

符。

下面给出一些标识符的例子：

WriteIn

Exit

Real2String

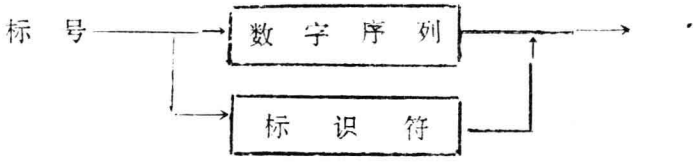
System.MemAvail

Dos.Exec

Crt.Window

1.3 标号

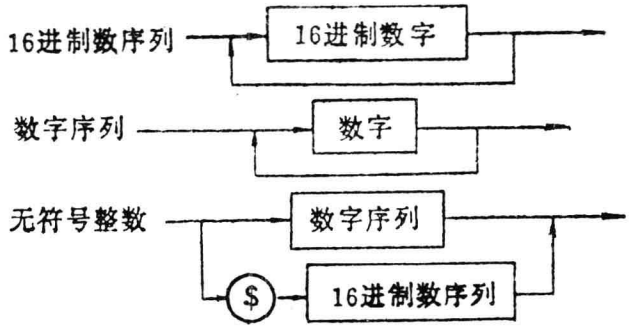
标号是取值为0至9999的数字序列。标号中的开始0无意义。标号和goto语句一起使用：



在Turbo Pascal中可用标识符做标号。

1.4 数字

十进制可用作整型和实型常数。16进制的整型常数以$做前缀。实型采用工程表示法，E或e后跟冥数。例如：7E−2代表7×10^{-2}，12.25e+6或12.25e6都代表12.25×10^{6}。下面是数字的语法图：

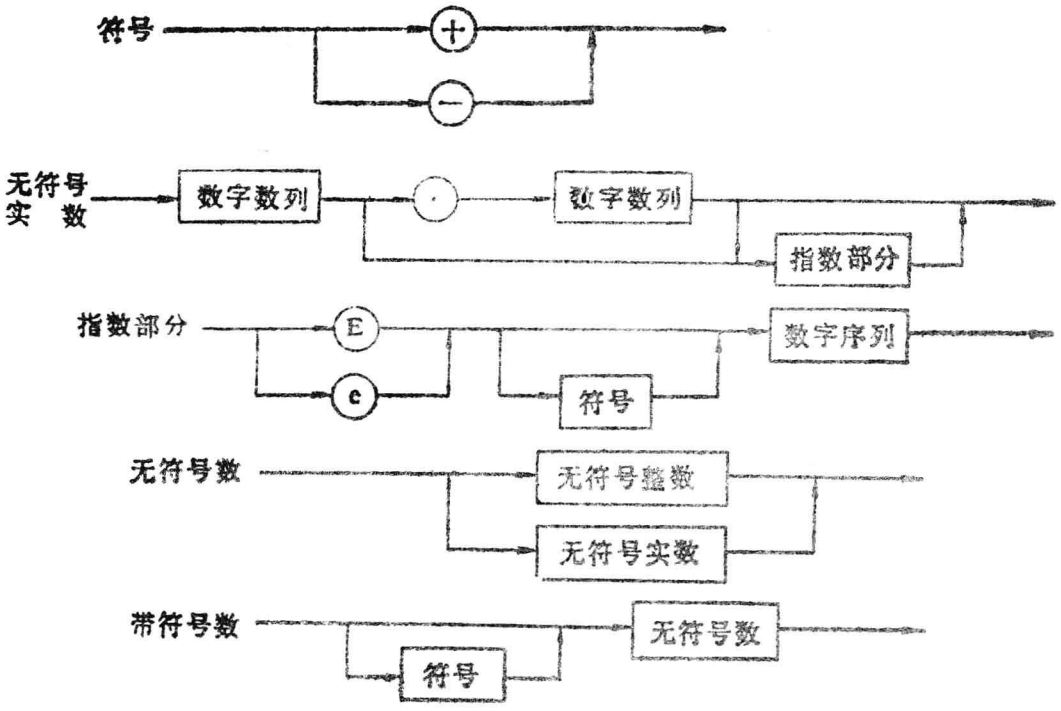


含小数点或指数的数表示实型数。别的十进进制数表示整型数，整型范围从-2147483648至2147483647。

用16进制数可表示整型常数。范围从$00000000至$FFFFFFFF。

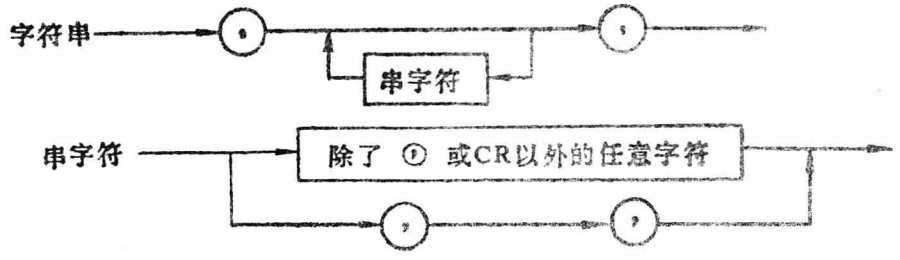
1.5 字符串

字符串由扩展的ASCII字符组成，可以是空串（串长为0）。字符串应在程序的一行内



并用单引号'括住。若两个'之间没有任何字符，则称之为空串（null string）。字符串的长度就是两个单引号之间的字符数。字符串中的单引号用两个连续单引号表示：''。

Turbo Pascal允许在字符串中出现控制字符。'#'符号后跟一个范围在0至255之间的一个无符号整常数就表示一个字符，该字符的ASCII码值就等于这个数。'#'和整数之间不能有空格之类的分隔符。字符串中有多个控制符时，注意不能把分隔符也放于其间。



长度为0的字符串（空串）只能和字符串类型兼容，长度为1的字符串可以和任意字符及字符串类型兼容。长度n（n≥2）的字符串和任意字符串类型及n个字符的紧缩数组兼容。

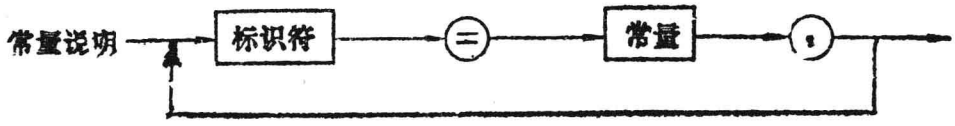
下面是一些字符串的例子：

'TURBO'　'You''re'　'I see'　''''　';'　' '

'Line 1'#13'Line2'　#7#7'Wake up!'#7#7

1.6 常量说明

常量说明把一个标识符定义为一个常量。常量标识符不能出现在它的说明中的右边。



1.7 注释

编译程序不理睬注释。下面的例子是注释行：

{Any text not containig right brace}

(*Any text not containing star/right paren*)

在一个注释中，若符号{或（*后紧跟的是一个$符号，则表示这是一个编译指令。有关编译指令的详细介绍见附录C。

1.8 程序行

Turbo Pascal程序行的最大长度为12。

1.8.1 常量表达式

作为标准Pascal的扩展，Turbo Pascal允许使用常量表达式。常量表达式是由编译器运算，没有实际执行代码的表达式。如：

```
100   'A'   256-1   (2.5+1)/(2.5-1)   'Turbo'+''+'Pascal'
chr(32)      Ord('z')-Ord('A')+1
```

简单常量表达式是单个常量如100或'A'；标准Pascal只允许使用简单常量，Turbo Pascal允许使用常量表达式。

由于编译时就完成常量运算，因此下面结构在常量表达式中是不允许的：

* 引用变量和类型变量
* 函数调用
* 地址操作符　除了这些限制，常量表达式和普通表达式语法差不多（见第六章“表达式”）

下面的标准函数允许在常量表达式中出现：

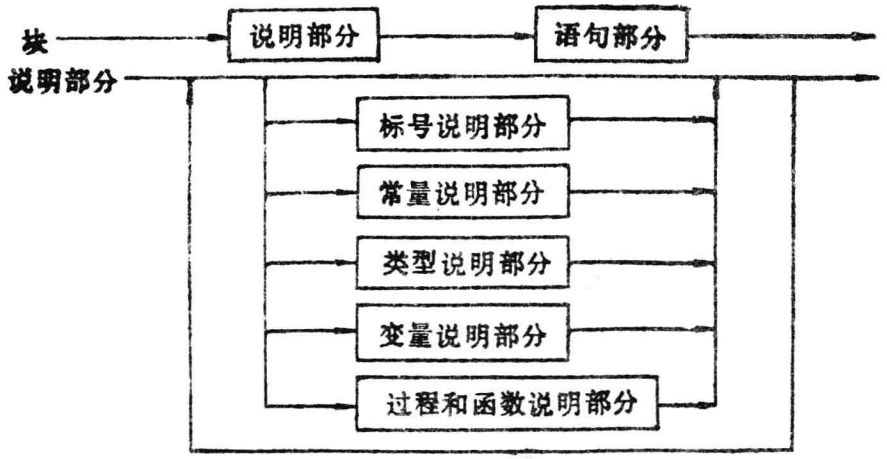
Abs	chr	Hi	Length	Lo	Odd	Ord
Pred	Ptr	Round	Sizeof	Succ	Swap	Trunc

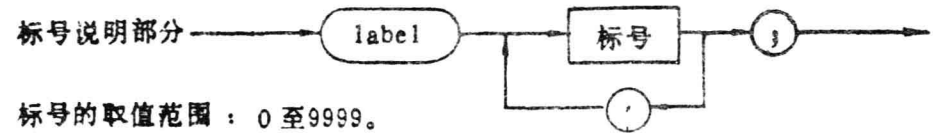
第二章　块、位置和作用域

块由说明和语句组成。其中说明的次序可以任意。块是过程说明、函数说明、程序或单元的组成部分。所有在说明部分说明的标识符和标号是该块的局部变量。

2.1 语法

块的格式如下：

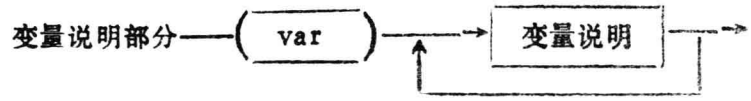




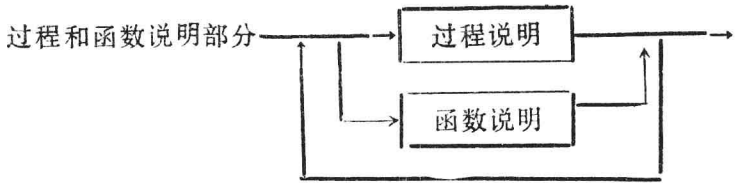
标号的取值范围：0至9999。

常量说明部分包含对块的局部常量的说明。

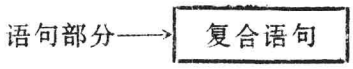
类型说明部分包括本块的所有类型说明。



变量说明部分包括对块内局部变量的说明。



过程和函数说明部分包括块内的局部过程和局部函数的说明。



语句部分定义语句或该块要执行的算术式。

2.2 标识符作用域规则

一个说明语句中的标识符或标号必须出现在说明的作用域内。标识符或标号的作用域是从其说明开始，到所在块结束为止。但可以有以下的例外：

（1）内部块中的再说明

假如块Exterior包含子块Interior，而Exterior和Interior同时有一个同名的标识符（比如：j）这时，Exterior和Interior只能存取各自说明的j。

（2）块内的说明位置

不能在标识符和标号被说明之前引用它们。指针类型的基类型可以是一个没被说明的标识符，但它必须在其指针说明块中被说明。

（3）块内再说明

一个标识符或标号只能在一个块的外层说明一次，然而可以在该块的一个子块中或在记录的域表中再说明。

记录域标识符是在记录类型中说明的，只有和该记录类型变量一起引用才有意义。因此可在同一块中，同一记录类型的不同层次，用同一个名字说明一个域标识符。在一个块中，一个标识符可被再说明为一个域识符。

2.3 接口和标准标识符的作用域

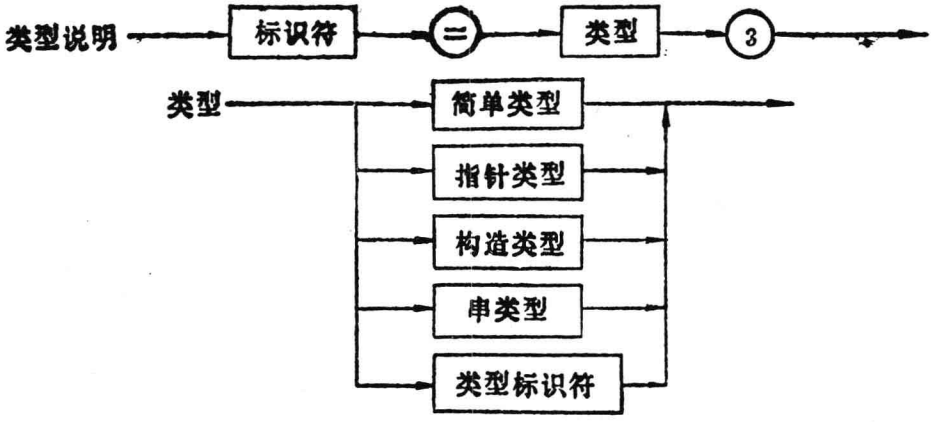
包含uses子句的程序和单元可访问在uses子句中出现的单元中的接口部分的标识符。

uses子句中的每个单元有一个新的作用域，该域包含其余各单元及整个程序。uses子句中的第一单元表示最外层的作用域，最后一个单元表示最内层的作用域。这意味着：如果两个以上的单元说明了同一个标识符。那么对该标识符的非限制引用将选择uses子句中最后一个单元中说明的那个标识符。但限定标识符可以选择任意一层上的标识符。

Turbo Pascal预定义的常量、类型、变量、过程和函数的标识符对包含有单元的块和整个程序都是有效的。实际上，这些标识符是在System单元中定义的。任何单元或程序都能隐含地调用System单元。单元或程序可以重新定义标准标识符，但仍能通过限定标识符来使用原来的标识符。如：System.Integer和System.Writeln。

第三章 类 型

说明变量时必须给出它的类型。变量的类型限制了它的值域和其上的操作。类型说明指明了一个标识符的类型。



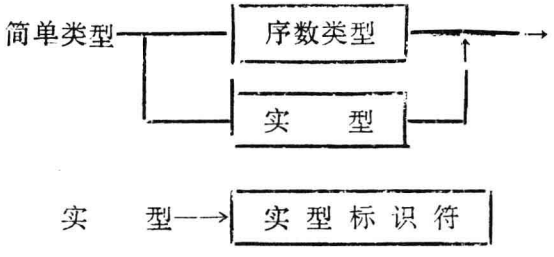
当一个标识符出现在类型说明的左边时，它被说明为一个类型标识符。

除了指针类型，一个类型标识符的作用域不包括它本身。下面是标识符的五种类型：

- 简单类型
- 字符串类型
- 结构类型
- 指针类型
- 过程类型

3.1 简单类型

简单类型定义了值的有序集合。



实型标识符可以是下列标准标识符：基本实型real，单精度single，双精度double，扩展型extended和装配十进制comp。（参阅第一章中的“数字”和字符串常数”两节）。

3.1.1 序数类型

序数类型是简单类型的一个子集。除了实型，所有的简单类型都是序数类型。可以用下述四个特征来进行区分：

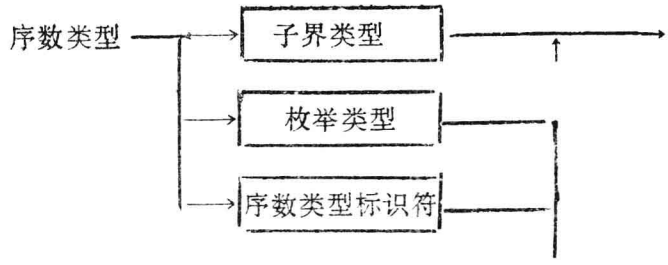
(1) 一个给定的序数类型的所有值是一有序集合。每个值在集合中都具有一个次序，此次序值是一整数。

(2) 标准函数Ord可以返回序数类型值的次序值。

(3) 标准函数Pred可以返回序数类型值的前驱，但当Pred的参数是序数类型中的第一个值时出错。

(4) 标准函数Succ可以返回序数类型值的后继，但当Succ的参数是序数类型中的最后一个值时出错。

下面是序数类型的语法图：



Turbo Pascal有七个预定义序数类型：整型，短整型，长整型，字节、字、布尔型和字符型。还有两个用户定义的序数类型：枚举型和子界型。

1. 整型

有五个预定义的整型：短整型，整型，长整型，字节和字。每个类型的取值范围如下：

表3.1整数类型

类　型	范　围	格　式
shortint	−128 .. 127	带符号8位
integer	−32768 .. 32767	带符号16位
longint	−2147483648 .. 2147483647	带符号32位
byte	0 .. 255	无符号8位
word	0 .. 65535	无符号16位

按下列规则，对于整型数的算术运算可使用8位，16位或32位的精度：

·一个整型常量的值是预定义的整型中包括该值的最小整型。

·对于一个二元操作符，在操作前，两个操作数均需转换成它们公有的类型。所谓公有类型是能包含有两个操作数类型的所有值的最小整型。例如：整型和字节的公有类型是整型，而整型和字的公有类型是长整型。操作时用的是公有类型的精度，得出结果也为公有类型，

·赋值语句右边表达式的值与左边变量的类型无关。

·字节型操作数，在作算术运算前，被转换为字型。

2. 布尔型

布尔值是由预定义的常量标识符False和True来表示的。因为布尔型为一枚举型，所以有下述关系：

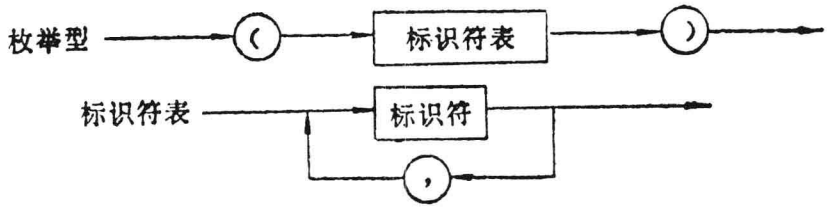
* False<True
* Ord(False)=0
* Ord(True)=1
* Succ(False)=True
* Pred(True)=False

3. 字符型

字符型值是按照扩展ASCII码来排序的字符集合。函数Ord(ch)返回ch的次序值。长度为1的字符串常数代表一个字符型常数值。字符类型的所有值都可由标准函数Chr产生。

4. 枚举型

枚举型是通过列出所有值的标识符来定义的有序集合。这些值的次序与枚举类型说明中的标识符次序是一致的。



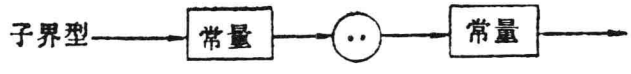
枚举型的标识符数中的标识符是作为本块中的常量来说明的，这个常量的类型就是正在说明的枚举型。第一个被枚举的常量的次序值为0，下面是枚举型的例子：

```
suit= (club, diamond, heart, spade)
```

根据这个说明，diamond是类型为suit的一个常数。函数Ord给出枚举型常量的次序值。例如：Ord (club) =0, Ord (diamond) =1, 等。

5. 子界型

子界型是序数类型的一个子域。这个序数类型称之为宿主类型。指定子界型中的最小、最大值就定义了该子界型。



两个常量必须具有同一序数类型，a..b要求a≤b。举例如下：

```
0..99
-128..127
club..heart
```

子界类型的变量具有宿主类型变量的所有特征，但是取值范围必须在指定的界限内。

3.1.2 实型

实型是实数的一个子集。它可用由固定位数的数字组成的浮点数表示。浮点数由3个值组成：m，b，e，即$n=m\times b^e$，其中$b=2$，m和e是该实型范围内的整数值。一共有五种实型：基本实型，单精度，双精度，扩展型和装配十进制数。实型的精度和范围见下表：

3.2 实数类型

类　型	范　围	有效位数	字节数
real	2.9E−39～1.7E+38	11—12	6
single	1.5E−45～3.4E+38	7—8	4
double	5.0E−324～1.7E+308	15—16	8
extended	1.9E−4951～1.1E+4932	19—20	10
comp	$-2^{63}+1\sim 2^{63}-1$	19—20	

装配十进制数只用于整数，其值范围大概相当于-9.2×10^{18}到9.2×10^{18}。

在执行实型运算时，Turbo Pascal支持两种代码生成代码生成方式：软浮点和8087浮点。用编译指令$N可选择合适的方式。没有8087时，编译指令{$E}可用软件仿真8087。

(1) 软浮点

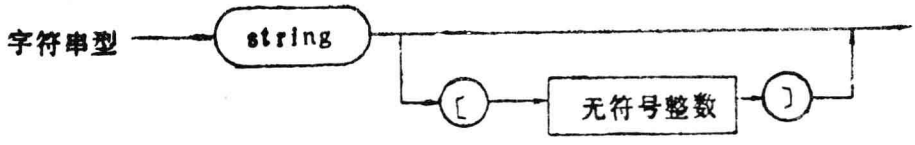
在{$N−}状态（缺省状态）下，生成的代码调用软件包中的库例程来进行所有的实型运算。此时只能使用基本实型，否则会出错。

(2) 8087浮点

在{$N+}状态下，生成的代码用8087协处理器进行所有的实型运算。此时可以使用所有五种实型。编译指令$E用于确定是否需要软件仿真8087。参阅14章。

3.2 字符串型

字符串型的值是具有可变短度的一个字符序列（长度决定于程序运行时的实际字符个数）。字符串常量的长度应在1至255之间。字符串型说明中没有指定长度时，取缺省值255。标准函数length返回字符串的实际长度。

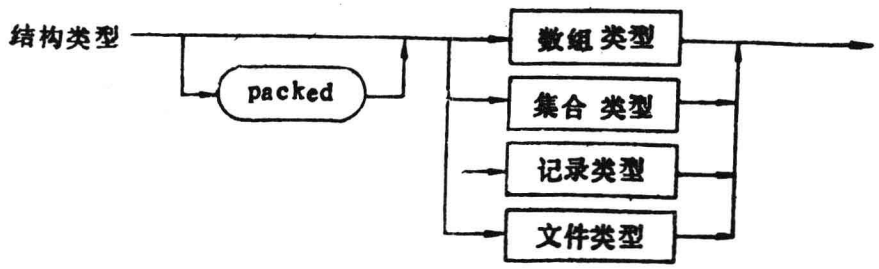


任意两个字符串值的次序是由相应位置的字符值的次序来确定的。空串只能和其它的空串相等，它们具有最小值。

字符串中的字符可以作为一个数组的元素来存取。详见第四章中的“数组、字符串和下标”一节。

3.3 结构类型

结构类型是由它的构造方法和它的元素类型决定的，它可存放多个值于一身。如果它的元素也具有结构类型，就形成了多重结构。结构类型对构造的层数没有限制。

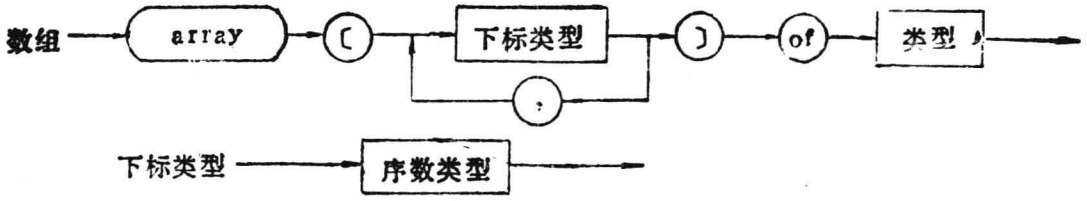


在结构类型说明中，关键字packed通知编译程序压缩数据存贮。在Turbo Pascal中，packed不起作用，在任何时候，只要有可能就自动压缩存贮。

注意：Turbo Pascal的类型只能有65520字节以下的大小。

3.3.1 数组类型

数组由有固定数目的同一类型的元素组成。



下标类型表示数组中元素的个数，每个下标类型对应数组的一维，数组的维数没有限制。下面是一个数组的例子：

```
array [1..100] of real
```

如果一个数组元素的类型是另一个多维数组。例如：

```
array [boolean] of array [1..100] of array [Size] of real
```

等价于：

```
array [boolean, 1..100, Size] of real
```

通过数组标识符和一个以上的在括号中的下标来访问数组元素（见第四章）。

3.3.2 记录类型

记录类型中包含一些不同类型的元素或域。记录类型说明指出了该类型中各个域及域的类型。其语法图如下：

记录类型说明中不可缺少的部分是一个域列表，给出每个域名及类型。下面是一个记录类型的例子：

```
record
```