绿色程序设计语言参考手册

一个应铁人计划要求设计的语言

浙 江 大 学

陈增武　　金连甫 译

浙江《计算技术通讯》编辑部印

目　录

1. 概　论

本报告叙述绿色语言。根据国防部铁人计划的要求，绿色语言体现了一个新企图，即将经典语言的特征与在专用语言中出现的特征结合起来。本语言中包含异常情况的处理，并行处理，数据表示法指明，封闭定义，低级输入输出，访问与系统有关的参数等。

1.1. 设计目标

绿色语言的设计有三个优先考虑的因素：意识到程序可靠性和易维护的重要，程序设计和人的活动密切相关和有效性。

为促进可靠性和易维护性而对语言提出的要求，已经很好的建立起来。今后重点是将程序的可读性置于可写性之上。例如，绿色语言要求，程序中的变量应作显式说明，对它们的类型应予规定。自动类型转换通常是不允许的。因此，翻译程序可以保证，目标的类型能满足预期的要求。同时避免有错误倾向的概念，语言的语法避免使用编码格式，以使之更象英语的结构。最后，语言给程序单元的独立编译提供强有力的支持。

与程序员有关的事情也是设计中着重考虑的。首先，保持语言尽可能的小，回避特殊情况和细微特征，因为它们常常妨碍程序设计的进一步发展。在语言结构上使基本概念的数目最少，并企图以一致而简单的方式归拢所有的特性。若某结构的形式或意义难于用系统方式表达时，则不予采纳。

语言都不能回避有效性的问题。那些需要过分精致的翻译程序或导致低效率地使用存贮器，或运行时间效率低的语言，将迫使所有程序和在所有机器上都成为低效率。绿色语言的每一结构都是研究过现在的实现技术的。任何实现过程尚不清楚或者过分耗费机器资源的结构均不采用。

大概最重要的是，上述目标一个也不能到事后再考虑能否实现，本设计目标从一开始就控制着整个设计过程。

1.2. 语言概述

用绿色语言书写的程序是一系列的高级程序单元，各个程序单元可以独立编译。程序单元可以是子程序（它定义可执行的算法）、定义模块（它定义实体的集合）、或者是路径（它定义并发计算）。可以独立编译将使程序可以分为数个独立部分，分别进行设计，书写和调试。这种可能性对于大型程序和建立程序库是特别有用的。

子程序为表达算法的一个基本单元，子程序可以带有参数，用它们和其他程序单元建立联系，绿色语言规定了三类子程序：过程，函数和异常处理。

过程子程序是一系列动作的逻辑付本。例如，它可以读入数据，修改变量，或者产生输出。函数子程序是计算出一个值的数学函数的逻辑副本，与过程不同的是，函数不能有付作用。异常子程序是为了在程序执行期间处理可能动态发生的特殊情况，如算术溢出，断言不成立，或者由用户定义的异常情况。

定义模块也是一个基本单元，用之来联结逻辑上有关的实体。定义模块部分可以由用户隐藏起来，因此仅允许访问定义模块表示的逻辑性质，例如定义一模块可用来决定数据和类型的公共池，有关的子程序包，或在一个新的封闭类型的聚集。

路径是定义并发计算的基本单元。路径可以在多处理机上执行，也可以在单处理机上交替执行来实现。各路径之间的通讯是通过各路径中的信箱来进行，信箱可使各路径同步和传递数据。

每个程序单元通常由两部分组成：说明部分：决定在程序单元中使用的逻辑实体；语句表部分：决定程序单元的执行。

说明部分将名与说明的实体相联系。名可以表示一个类型、一个常量、或一个变量。说明部分也引入其它子程序名和参数，路径和在程序单元中使用的定义模块。

语句描述执行的动作。赋值语句将变量的当前值用一个新值来替代。子程序调用语句，在根据调用者提供的自变量与相应子程序的形式参数联系起来以后执行该子程序。

如果语句和情况语句根据语句首部的条件或表达式的值在一组语句中进行选择。断言语句指出无论何时在程序到达控制点时，必须成立的一个正确性条件。异常语句明显地提出异常子程序的动作所要求的特定状态。

在语言中基本的机械性重复是循环语句。循环语句确定一组要重复执行的语句直到重复的条件被执行完，或在遇到循环出口语句为止。

某些语句仅用于路径中。内部语句规定一组可以开始执行的路径。连接语句规定一路径和其他路径通过信箱准备连接。局部路径请求规定一路径，它准备连接到其他路径。

无论何时局部路径请求和连接语句都要实现会合，产生任何一个规定的数据传送，局部路径和连接路径可以继续。

语言中每个元素均有一类型，确定了它的逻辑性质，也决定了可能执行的操作。有两种基本类型：纯量型和组合型。

纯量型INTEGER、BOOLEAN和CHARACTFR是预定义的。标度数型为进行精确的数值计算提供了工具。实型提供浮点计算的工具，它有必要的近似值。枚举型为用户提供了确定与离散值有关的问题的工具。

组合型可以定义有关元素的结构化的汇总。在语言中，组合型有数组结构、纪录结构和对记录结构的访问，后者是动态分配的。一族记录结构可以通过一带变体部分的记录类型予以定义。

从类型概念可以引伸出子类型概念，从而用户可以限制于类型中一组可允许的值。子类型可用来定义纯量的子域，或一定范围下标值的数组，或者特殊变体记录。

表示法的特征可以用来规定数据类型与基本机器之间的映照。例如，用户可以规定，数组可以表示成装配格式，已知类型的目标必须用规定个数的位来表示，或者记录分量用规定的存储格式来表示。

最后，语言为条件编译和属类型程序单元提供可能性。

1.3. 回顾

在最初的PASCAL报告中，Niklaus Wirth写道“在一个新的语言中舍弃一些什么实际上比包含什么更为关键。决定舍弃一些特征时，不仅需要通晓这些特征，（并需知道缺少它时如何工作），而且由于这些特征已存在于其它的现有语言中，要有由于缺少它们而面对着不可避免的批评的勇气”。

在我们设计语言时也存在这个问题，尽管铁人计划而比通常情况来的小些。这个要求经常简化设计过程以使我们集中到按规定的良好目标而进行的逻辑系统的设计中，而不是花在确定目标本身上。

我们设计工作较简单的另一原因由于在我们以前，根据类似的目标，已有数个Pascal衍生语言的成功的试验。其中有Euclid, Lis, Mesa, Modula, Sue和CS$_4$等。在这些语言中的许多关键想法和语言格式已经吸收到绿色语言中。可以说，这些以前的研究应该认为是真正研究成果，绿色语言是语言设计工程总结出的方案，它体现了当前的工艺水平。

其它一些语言，诸如ALGOL68，Simula以及新近研究的语言如Alphard和Clu也从不同方面影响到本语言，但它们比Pascal类的语言的影响少一些。

谨对本语言设计中提供帮助的我们的同事以及同事的同事表示谢意。为了防止差错，我们在这里不一一列出他们的工作，不指名的对他们表示深切的感谢。

2. 词 法 元 素

这一节定义语言的词法无素。

2.1. 字 符 集

所有的词法元素可以由*ASCII*的64个字符的子集组成。这些字符如下所述：

（*a*）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

（*b*）数字

0 1 2 3 4 5 6 7 8 9

(*c*)特殊字符

! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _

(*d*)低横线字符

_

空格符

2.2. 标 识 符

标识符由字母和数字的序列构成，必须以字母开始，低横线可以插在一个标识符的各部分之间，一个标识符必须列在一行内，所有的字符都是有效的。一个被说明的标识符通常是作为一个名，带有一个指明它的用法的前缀。

如：*variable_name*或*type_name*。

例：

COUNT *X* *LINE—COUNT* *GET—SYMBOL*

SNOBOL—4 *X1* *PAGE—COUNT* *STORE—NEXT—ITEM*

2.3. 数

数分三类：整数、标度数和实数。低横线可以插在一个数的各个部分之间，但不起作用。

整数由一系列的数字构成。

例：

12 0 1977 123456 123_456

标度数或写成整数，或者为一系列数字，中间带有小数点。

例：

12.0 0.0 123.456 10_000.1 1200

实数是将字母*E*和指数接在一个整数或标度数后面构成的。指数是一整数前缀以可任选的+或-号。

例：

12.0*E*10 0*E*0 1*E*6 3.14159_26535*E*0

以2，4，8或16为基底的非十进制数也写成一系列字后跟#和基数。对十六进制的数，(底为16)，字母*A*到*F*按习惯意义使用。

例：

011001#2 1777#8

2*FEEE*#16 0*FFF*#16

2.4. 字 符 串

字符串由一系列字符所构成，两边用引号括起来。长度为1的串也表示字符型的文字，如果串中含有引号，则该引号必须重复地写一个。每一个串必须列在一行上。如果一个串有几行，则可用连接操作符&接连起来。

例：

```
"A"  "*""ASMALL STRING"

"FIRST PART OF A STRING THAT"&
"CONTINUES ON ANOTHER LINE"
```

2.5. 注 解 和 语 用

注解可放在程序之内。一个单独的注解以字符——开始，以一行的末端作结束。注解不能出现在一个表达式或一个语句的内部。一个嵌入的注解左右用方括号括起。嵌入的注解不能跨行。注解在编译时总是略去的，设置注解的唯一目的是为了使人便于阅读程序。为了便于阅读本手册，注解将用大字母和小写字母写出。

语用（对语用学来说）用来将信息送至翻译程序。它们以关键字 *Pragmat* 作为开始，以一行的末端作结束。语用不可以出现在简单语句或说明内部。

注解例：

```
end{GET—SYMBOL};
——a stand alone comment
——and its  continuation
```

语用例：

```
pragmat  NO—LIST                    禁止列表
pragmat  LIST                       恢复列表
pragmat  OPTIMIZE TIME              优化指明
pragmat  INCLUDE COMMON—TEXT        包含正文文件
pragmat  DEBUG                      建立排除故障状态
```

2.6. 属 性 词

属性词表示程序结构的属性，属性词是标识符前缀以一个或多个字符' 构成，他们用来描述相应的语言结构。因为属性词总包含一个字符' ，故他们的标识符不必为关键字。

例：

```
DATE'SIZE    REAL'PRECISION
```

2.7. 保 留 关 健 字

abs	*declare*	*if*	*or*	*scale*
access	*definition*	*import*	*others*	*separate*
algorithm	*delay*	*in*	*out*	*select*
alignment	*div*	*inline*		*send*
all		*inner*	*packing*	*subtype*
and		*interrupt*	*parameter*	
array	*else*	*is*	*path*	
assert	*elsif*		*pragmat*	*then*
at	*end*	*loop*	*precision*	*type*
	exception		*private*	
	exit	*mod*	*procedure*	
begin				*until*
bits		*new*	*raise*	*use*
box	*for*	*none*	*range*	
	function	*not*	*receive*	
		null	*record*	*when*
case			*repeat*	*while*
connect	*generic*	*of*	*return*	
constant	*goto*	*only*	*reverse*	***xor***

2.8. 空 白 的 约 定

空格可以随意地插在词法元素之间，只有注解和语用，将一行的结束看成一个空格。两个标识符（保留的或不保留的）之间必须至少有一个空格，如果不是用其他规定的符号分隔开的话。

2.9. 语 法 记 号

在下面的章节中，用一个*Backus—Naur* 范式简单变形来描述上下文无关的语言的语法，特别有：

（*a*）中间可能包含有低横线的小写字表示语法范畴，如*adding—operator*

（*b*）黑体字表示语言中的关键字，注① 如 *array*

注①译者注，以后我们用下面加横线来表示黑体字。

(c)方括弧中包含的是可选项，如

return〔*expresssion*〕

(d)花括号括起的项表示重复数次，也可以是0次，例如，一个标识符表表示成

identifier—list::=*identifier*{ *identifier* }

3. 说 明 和 类 型

这一节叙述语言的类型以及对常量与变量说明的规定。

3.1. 说 明

说明连接一个名与一个语言结构，有几类说明：

说明::=

元素说明 | 类型说明
| 子类型说明 | 访问类型说明
| 子程序说明 | 路经说明
| 定义说明 | 属类范例
| 变体部分 | *null*;

*null*说明不引出新的名字，它可以用于诸如定义一个无分量的记录变体的情况。对于元素，类型，子类型和存取类型等说明在这一节中叙述，其它的说明则放在以后各节中。

3.2. 元 素 说 明

元素说明引入常量和变量

元素说明::=变量说明 | 别名说明
| 常量说明 | 受限常量说明

变量说明::=标识符表：类型〔：=表达式〕;

别名说明::=标识符：类型==变量;

常量说明::=标识符：*constant*〔类型〕=表达式;

受限常量说明::=标识符：*constant*类型;

变量说明将一个或多个标识符与某一类型的几个新变量联结起来，说明中可以规定变量的初值。置初值等价于在说明后立即执行一个赋值语句。

别名说明连接一个局部名与一个变量，局部名可作为变量的速记，例如一个数组或

一个记录结构。

常量说明一个名和一个表达式规定的值，当常量说明详细说明后，即将其值算出。当值是文字且它的类型已知时，常量的类型可以略去。

受限常量说明规定常量的名和类型，它的值在规定限制的上下文中计算。

例子：

```
ITEM－1、ITEM－2 : INTEGER;
SORT—COMPLETED : BOOLEAN := FALSE;
OPTION——TABLE : array (1 ·· N) of OPTION;
ANCESTOR : PERSON == JOHN · FATHER · MOTHER;
ACCURACY : constant = 1E－30;
LIMIT : constant INTEGER = 10__100;
NULL—ENTRY : constant ENTRY
```

3.3. 类型和子类型说明

类型规定该类型中的元素的一组性质。类型说明连接一个名和一个类型。

类型说明::＝*type*标识符＝类型定义；

类型定义::＝类型 | *private*〔*parameter*〕

类型::＝简单类型定义〔约束范围〕| 数组类型 | 记录类型

简单类型定义::＝纯量类型 | 类型记号

类型记号::＝类型名 | 子类型名 | 属性

约束范围::＝纯量约束 | 数组约束 | 记录约束

子类型说明::＝*subtype*标识符＝类型记号〔约束范围〕；

子类型说明连接一个名和一个母体类型，它的性质可以用某个约束范围来限制它。作为类型定义的*private*用法在定义模块那一节中说明。

3.4. 纯 量 类 型

纯量类型用来描述离散值和实数值。离散型可以被用来附标。所谓*INTEGER*，*BOOLEAN*和*CHARACTER*的纯量型是预先定义的离散型，其它类型可以由用户说明之。

纯量类型::＝离散型 | 实型 | (值域)

离散型::＝标度数型 | 枚举型

纯量约束::＝*range*(值域)

值域::＝简单表达式··简单表达式

纯量约束规定为一个值域，它是母体类型的子集。值域*L*··*R*说明了一个子集，包含*L*到*R*的各值。值域作为纯量类型时，等价于确定该值域的母体类型，但取值受值域约束。

对于有序的所有离散型，函数*SUCC*和*PRED*是预定义的。它们取该类型的值域中的后一个或前一个值。此外，对于一个有序的离散子类型或类型T，属性$T'FIRST$和$T'LAST$表示类型的极小和极大值。

3.4.1 整　型

预定义类型*INTEGER*表示全体数的一个子集。整数的值域则根据各个实现过程隐含地限制着。引伸出来的类型可以用值域约束来获得。

例子：

```
type  PAGE—NUM=INTEGER;
type  LINE—SIZE=(1..MAX—LINE—SIZE);
Subtype  SMALL—INT=INTEGER  range(—10..10);
Subtype  COLUMN—PTR=LINE—SIZE  range(1..10);
```

3.4.2 标 度 数 型

标度数型为非整数值的精确计算提供工具。对应着每一个标度数型，有一个常数比例因子。该类型的所有量都是该比例因子的整数倍。比例因子在类型说明中给出，其值或为整数或为整数的倒数。

标度数型::=*scale*　简单表达式

决定比例因子的表达式的值必须在编译时知道。在类型的作用域内，标度数型T的比例因子可以用属性$T'SCALE$访问它。

例子；

```
type  TICK=Scale  1 // 60  range(0..3600);
type  VOLT=Scale 1 // 1000  range(0..1.5);
type  JOULE=Scale  1000  range(0..1__000__000);
```

3.4.3 实　型

实型为执行具有必要的近似值的浮点计算提供工具。与实型数有关的精度在类型说明中规定，它只限于浮点所提供的误差。

实型::=*precision*　简单表达式

确定精度的表达式的值必须在编译时知道。在实型T的作用域内，类型精度可以用属性$T'PRECISION$进行访向。

例子：

```
type  LONG—REAL=pricision  1E—40
type  COEFFIDIENT=pricision  1E—10range(—1E0..1EO)
```

3.4.4 枚 举 型

枚举型定义一组值，这些值用列表方式列出。如果使用分格符！，则这些值是无序的；如果使用分格符＜，则按递增顺序枚举列出。

枚举型::=(枚举型值{！枚举型值})|(枚举型值{＜枚举型值})

枚举型值::=标识符 | 字符

例子：

```
type  SUIT=(CLUBS<DIAMONDS<HEARTS<SPADES);
type  HEX—LETTER=("A"! "B"! "C"! "D"! "E"! "F");
type  DAY=(MON! TUE! WED! THU! FPI! SAT! SUN);
Suptype  WEEK—DAY=DAYrange(MON··FRI);
Subtype  REST—DAY=DAYrange(SAT··SUN);
```

3.4.5 布尔型与字符型

称作*BOOLEAN*的枚举型是预定义的。它含有两个无序值*TRUE*和*FALSE*。

称作*CHARACTER*的枚举型是一标准库，它所允许的字符集及其顺序由给定的实现过程所决定。

3.5. 数 组 类 型

一个数组汇集了同类型的数个元素。数组中各元素用下标来区分。

数组类型::=*array*(下标{，下标})*of*类型

下标::=值域记号 | *

值域记号::=值域 | 类型记号

数组约束::=(值域记号{，值域记号})

数组类型由下标号和元素的类型所决定。下标有规定的值域，它不是数组类型的一部分。下标*表示任何离散型的任意值域。

当在类型说明中已定义了一个数组类型名后，则数组约束可以将名连结到一个特定的下标实在值域。

对数组类型*T*，属性*T'RANGE*，*T'FIRST*，和*T'LAST*分别表示第一个下标的值域，下界和上界。类似地，属性*T"RANGE*，*T"FIRST*，和*T"LAST*表示第二下标的值域，下界和上界，其他依此类推。

例子：

```
typeT=array(*, *)ofBOOlEAN;
A:   T(1··10, 1··100);
B   array(1··10, 1··100)ofBOOLEAN;
A'  FIRST  〔值是1〕
A"  LAST  〔值是100〕
```

3.5.1 动 态 数 组

数组的每一下标的值域必须在数组分配时知道。如果下标的值域在编译时不能算出，就必须作为动态数组考虑。动态数组也不能出现在用访问类型表示的记录中。

3.5.2 数 组 量 和 串

一个数组量表示数组的一个值，即一个数组目标结构。下标用选取表示，元素值用

表达式表示。

数组量::=字符串|〔类型名〕(元素指明{，元素指明})

元素指明::=选取：表达式

选取::=被选值{|被选值}

被选值::=数|枚举值|值域记号|*others*

一个选取规定了一组离散型的若干不同的值。在选取中值域记号可看作该值域中的所有值。关键字*others*表示在上述选取中没有规定的所有其他元素。选取也用在情况语句和记录变体中。

字符串当作数组量考虑。N个字符($N\geqslant 1$)的串是字符型数组。他的值域是$1\cdot\cdot N$。

多维数组量作为数组的数组来处理。

例子：

```
type  TABLE=array(1..10)ofINTEGER;
type  LINE=array(1..MAX-LINE-SIZE)ofCHARACTER;
A:  TABLE:=(1: 2: 1, others: 0);
BLANK-LINE: Constant LINE=(I..MAX-LINE-SIZE:" ");
```

3.5.3 集　合

名为*SET*的预定义类型表示一维布尔数组。

```
typeSET=array(*)ofBOOLEAN
```

布尔量的操作符也可应用于布尔向量，即一维布尔数组中。这些操作符逐个元素执行相应的操作。数组量可用来表示集合的值。

例子：

```
typeWEEK=SET(DAY);                〔集合类型〕
(TUE!THU: TRUE;others: FALSE)     〔二天的集合〕
(X and Y)=X                       〔测试X是否为Y的子集〕
X(E)=TRUE                         〔测试E是否是X的元素〕
```

3.6. 纪 录 类 型

记录类型定义数个分量的结构。分量的名和类型在分量表的元素说明中给出。记录类型可以包含一个变体部分因而定义一族结构。

记录类型::=*record*分量表*end record*

分量表::={元素说明}〔变体部分〕

变体部分::=*case* 判别式*of*{变体} *end case*;

判别式::=变量名

变体::=*when* 选取⇒分量表

记录分量中定义的元素说明可以给出分量的初值。

例子：

```
type  DATE=
      record
        DAY : ( 1 ·· 31 )
        MONTH : MONTH-NAME ;
        YEAR : ( 0 ·· 2000) ;
      end record ;
```

3.6. 1 常量分量，不可赋值的分量和变体部分

说明为常量的记录分量用来表示常值的分量，该类型的所有记录中它保持相同的值。

说明为受限常量的记录分量是一个不可赋值的分量。它的值仅可以用完整的记录赋值来建立。

带变体的记录类型规定一族记录结构。变体部分用上面说明过的所谓判别式（或标记场）的分量来加以判别。每个变体确定对应的判别式的值的分量。判别式必须说明为受限常量，因而是不可赋值的。

例子：

```
type  PERIPHERAL=
    record
      STATUS : ( OPEN ! CLOSED ) ;
      UNIT : Constant ( PRINTER ! DISK ! DRUM ) ;
      Case UNIT of
          when  PRINTER⇒LINE-COUNT : ( 1 ·· PAGE-SIZE ) ;
          when  others⇒
                CYLINDER : CYLINDER-INDEX ;
                TRACK : TRACK-NUMBER ;
      end Case ;
    end record ;
```

3.6. 2 记录量和记录约束

一个记录量表示一个记录的值，即一个记录结构。其值用给出它的分量的值得到。

记录量::=〔类型名〕（分量指明{，分量指明}）

分量指明::=分量名{!分量名}：表达式

记录约束::=记录量

如果记录类型中包含变体，则被选的分量名必须对应于判别式的规定值。

如果上面说明记录类型包含几个变体，则记录约束可用来约束记录变量或子类型于规定的变体。记录约束规定被选取到的变体的值。当值由判别式提供时则表示成记录量的形式。记录量之例：

```
( DAY : 4 , MONTH : JULY , YEAR : 1776 )
( STATUS : CLOSED, UNIT : DISK, CYLINDER : 9 , TRACK : 1 )
```

记录约束之例：

```
subtype DISK—DEVICE=PERIPHERAL(UNIT:DISK);
```

3.7. 访 问 类 型

在程序中说明的正常的记录变量可用它的标识符进行访问。在说明部分活动期间内它一直存在，它是局部的，并称之为静态的。反之，访问类型的变量则用来规定进行动态分配的记录。

访问类型说明::=*access type*标识符==类型;

动态记录的访问由访问变量来实现，它可以用分配语句来建立，也可以用赋以另一个访问变量来实现。没有指明动态记录的访问变量的值用*none*表示。

每一访问类型隐含地定义着一组动态分配的记录，并可用该访问类型的变量作标识。一个记录可以用访问型的数个变量来指定。访问型记录的分量可以属于相同的访问类型。

表示法指明（见第10节）可以规定一个存贮空间，留作为(静态地)存放与一个访问类型有关的诸记录之用。

例子：

```
access type PERSON==
    record
        NAME:STRING;
        AGE:INTEGER;
        MOTHER:PERSON;
        FATHER:PERSON;
    end record;
access typeLIST-ITEM==
    record
        VALUE:INTEGER;
        SUCC:LIST-ITEM;
        PRED:LIST-ITEM;
    end record;
```

3.8. 类 型 的 相 容 性

每一类型，子类型，变量和常量均有一基本类型，用它来检验类型相容性。

不同的类型名的说明总表示不同的基本类型，即使他们的定义相同亦如此。类型约束不改变基本类型。子类型的基本类型是它的母体类型。变量或常量的基本类型是在说明中所出现的类型。

未命名的类型说明遵守下述规则：

（*a*）如果类型是枚举型或记录型的，则其基本类型不同于任何其它的枚举型或记录型，即使它们的定义在构造上相同也是如此。

（*b*）如果类型是值域，则基本类型是确定该值域的表达式；

（*c*）两个实型或标度数型有同一基本类型，如果它们的精度或比例因子相同。

（*d*）两个数组型有相同的基本类型，如果它们的维数相同且元素的基本类型和约束也相同。

如果类型*A*用另一类型*B*来定义，即

type $A=B$；

则*A*和*B*为两个不同的类型，具有同一逻辑特性，但不必具有相同的表示法。象*A*和*B*这样有关的类型之间显式转换是允许的，但必须要用类型表达式写出来。

3.9. 说 明 部 分

每一程序单元可以包含一个说明部分，用来规定它的说明以及其它局部信息。

说明部分::=〔移入子句〕{说明}{表示指明}{体}

体::=子程序体 | 定义模块体 | 路径体

在程序单元内说明的标识符有一作用域，由该程序单元以及未用相同标识符重新说明的内层单元所组成，标识符对说明它的单元来说是“局部”的，而对未用同一标识符重新说明的内层单元来说是“全局”的。移入子句用于定义模块的移入标识符，表示法指明定义特定的类型的表示法。说明表中的子程序体，定义模块体和路径体放在说明部分的尾部。它们的结构在以后章节中给出。

4. 变 量 和 表 达 式

4.1. 变 量

变量表示一给定类型的存储值。它可以表示一纯量值的名，数组名或记录名。此外，它还可以表示数组元素，数组片或记录分量。

变量::=变量名 | 数组元素 | 数组片 | 记录分量

数组元素::=变量（表达式{，表达式}）

数组片::=变量（值域记号）