ADA 语言

万 达　梁琨瑶　马淑雅　编译

尤尧中　校对

宇航出版社

内 容 简 介

ADA语言是美国国防部在实现一种通用高级语言进行的招标设计中开发出来的。它是一种功能极强的计算机高级语言，是在对比十六种方案基础上最后选定并由法国人以PASCAL语言为起点进行开发的。该语言可用于数值分析计算和系统程序设计，满足实时分析及并行操作要求，能在各种规律的大、中、小、微型计算机上实现。目前已受到国际上计算机的应用机构、大学和公司的高度重视，成为八十年代最有影响和最有代表性的一种计算机高级语言。

本书以法国计算机语言学专家Daniel-Jean David所著的Le Langage ADA为基础，参考其它有关资料编译而成。适合于计算机研制、应用、开发单位的科技人员及大专院校、职业高中计算机系的师生作参考。

ADA 语 言

编译者：万达 梁琨瑶 马淑雅

校 对：沈尧中

特约编辑：高敬松

☆

宇航出版社出版

新华书店北京发行所发行

各地新华书店经售

四二二九工厂印刷

☆

开本：787×1092 1/32 印张：7.75 字数：170千字

1987年7月第一版第一次印刷 印数：1—6,500册

统一书号：15244·0097 定价：2.10元

目　录

第一章　ADA语言的概貌

美国国防部的招标细则曾建议：新语言可以借鉴于Pascal，Algol 68或PL/1。一个很有意义的事实是：初选出的四家公司（CII—Honeywell—Bull，Intermetrics，SRI International及Soft—Tech.）不谋而合地都以PASCAL作为开发新语言的基础。本章将介绍ADA语言的概貌，目的在于指出：ADA语言有些概念是PASCAL的延伸，有些概念则摆脱了PASCAL的局限性。但是ADA并不是PASCAL的简单扩充，许多新的、重要的概念是ADA所特有的。

为便于阅读，本书采用如下约定：

1.ADA语言的键词用粗体大写字母；

2.标识符（无论是语言预定义的还是用户定义的）一律用普通大写字母。

1.1　结构语句

ADA语言最重要的结构语句是条件语句和循环语句。条件语句以**IF**语句为代表，循环语句以**LOOP**为代表。可举几个例子来说明ADA结构语句的特点及其源程序的书写格式：

例1.1　用A表示B的绝对值

①　用单分支**IF**

```
A : = B;
IF  B < 0  THEN
A : = - B;
END  IF;
```

② 用对称IF

```
IF  B < 0  THEN
            A : = - B;
            ELSE
            A : = B
END  IF;
```

从上面的例子中初步看到：

1.与PASCAL相同，ADA中赋值操作也由符号：＝表示。

2.在ADA语言中也采用；号，但分号的用法与PASCAL有所不同。ADA中的分号仅仅表示语句的结束，因此不必象在PASCAL中那样小心地使用。分号只要加在每一个语句的末尾即可。

3.在ADA语言中，注释是通过两条短线（双减号）引入的。注释可以放在语句之后，但在一行中，注释后面不得再接语句。例如：

```
A : = A + 1 ;  - - 将A的值加 1
```

关于循环语句，**LOOP**允许建立一个无条件循环，其格式如下：

```
LOOP
    语句列;
END  LOOP;
```

要跳出这个循环，需要在循环中引入条件，使之变为条件循环。这可以通过**EXIT**语句或**EXIT WHEN**语句来实现：

```
LOOP
    语句列;
EXIT  WHEN条件;
END  LOOP;
```

处于语句列之后的**EXIT WHEN**语句与Pascal中的**REPEAT UNTIL**（ADA不存在这一语句）语句功能相同。

通过**FOR……LOOP**结构也可以建立一个条件循环。

例1.2 求一维矢量V前十个元素之和

```
SUM : = 0;
FOR I IN 1 . . 10 LOOP
SUM : =SUM+V ( I );
END  LOOP;
```

例1.1和1.2只是ADA语言的程序段。可以由它们构成函数（**FUNCTION**）、过程（**PROCEDURE**）以及定义自身数据的分程序（**BLOC**），进而构成完整的ADA程序。

1.2 注、说明与可执行语句

通常在高级语言中把语句分为可执行语句与不可执行语句（或说明语句）。这种分类方法同样存在于汇编语言中。汇编语言中不可执行的语句是指导汇编器工作的命令（伪指令）。

由于Pascal语言有丰富的数据类型，说明语句有了很大的发展。这一点在ADA中被进一步强化。对于数据的定义十

分细致，以致从编译器的角度来看，一条说明的出现隐含着一种真实的处理。我们把这种处理过程叫做**加工（ELABORATION）**。对于说明的加工，一部分在编译时进行，另一部分则在程序运行时进行。

除可执行语句、说明语句之外，ADA中还包括给编译器提供的简单指令，称为**参注（PRAGMA）**。由于参注的存在，程序员可以对时空比例作灵活的选择，例如

```
PRAGMA OPTIMIZE (SPACE);
```

要求对生成码实行优化以节省存贮空间。

这样，在ADA语言中，语句就有三个等级：参注；说明语句与可执行语句。

1.3 数据类型管理

首先研究一个例子。我们要处理一周的各天，若采用Basic、Fortran，或PL/I就不得不用数码去一一对应。例如：1对应于星期一，2对应于星期二……。采用这种对应关系，由于汇编器无法去检验数值的有效性，赋值总是可以进行的。

Pascal第一个用定义数据类型的方式向数据抽象化的概念迈出了一步。它允许用户根据所要求的特性定义数据，而不是根据某种内部数字表达方式定义数据。例如，在Pascal中我们可以定义一个类型DAY：

```
TYPE  DAY = (MON, TUE, WED, THU,FRI,
      SAT, SUN);
VAR   TODAY : DAY;
```

变量TODAY属于DAY类型，所以下述语句都是非法的：

```
TODAY := 1;
TODAY := JAN;
```

因为编译器要检验赋值语句两边的一致性。除赋值语句外，在Pascal中还可以有：

```
IF TODAY = WED THEN……;
```

或

```
FOR TODAY := MON TO FRI DO……;
```

等语句。

遗憾的是Pascal就此止步，从而大大限制了类型定义的用途。譬如：

```
TYPE DAY = (MON, TUE, WED, THU,
            FRI, SAT, SUN);
     WEEKEND = (SAT, SUN);
```

在PASCAL中是非法的。二个类型不相容，因为SAT和SUN同时属于这两个类型。另外PASCAL中的子界和实数处理也受到一定的限制。

ADA在类型管理方面比PASCAL更加强化和完善。一方面每一个数据具有唯一的类型，决不能对某种类型的变量赋予其他类型的值。另一方面，我们将看到，在ADA中使用派生型、子类型以及**重载**（**OVER LOAD**）弥补了PASCAL语言的缺陷。

1.3.1 派生型

派生型是由下面的格式定义的：

```
TYPE T IS NEW S;
```

它表示由类型S派生T。S叫做父型（**PARENT**），T称为S的派生型。T具有S所有的字面值（**LITERAL**）和对S所定义的一切运算。然而父型和派生型是不同的类型，赋值时必须进行类型转换。在ADA中类型转换是使用类型名实现的。例如：

```
X : T;
Y : S;
```

在上述变量说明中，X为T类型，Y为派生型S，因此

```
X : = Y;
Y : = X;
```

都是非法的。然而

```
X : = T (Y) ;
Y : = S (X) ;
```

是合法的。

使用派生型，可以避免逻辑上属于不同种类的对象的运算。例如我们要清点苹果和桔子的个数，同时又要避免把二者混淆，这时用户可以定义两个整型的派生型APPLE—NB和ORANGE—NB：

```
TYPE  APPLE—NB  IS  NEW  INTEGER;
TYPE  ORANGE—NB  IS  NEW INTEGER;
A :     APPLE—NB;
B :     ORANGE—NB;
I :     INTEGER;
```

那么：

```
A : = 0 ; --是正确的，苹果数为零
A : = A + A; --是正确的
```

I：＝A＋B；--是非法的

I：＝INTEGER（A）＋INTEGER（B）；

最后这个算术表达式是正确的。在这里我们可以看到：类型名（如上式中的INTEGER）可以象函数名一样加以使用，将某一变量转换成为指定类型。

使用派生型，还可以实现区间约束。例如：

```
TYPE  INT  IS  NEW  INTEGER
              RANGE—1000 . . 1000;
```

也可以简化为：

```
TYPE  INT IS NEW RANGE—1000 . . 1000;
```

我们可以定义许多派生型，它们具有相重迭的区间或相同的数值，这也是**重载**的含义之一。例如：

```
TYPE  COLOR  IS(GREEN, ORANGE, RED);
```

可以对十字路口的交通灯定义成COLOR的派生型：

```
TYPE  LIGHT—OF—CROSS—ROAD  IS
          NEW COLOR;
```

以上两个枚举型具有相同的元素。我们可以使用类型名区别不同类型中的相同元素。例如：COLOR′（RED）表示颜色中的红色，而LIGHT—OF—CROSS—ROAD′(RED)表示交通灯中的红色，两者没有任何含义上的模糊性。

1.3.2 子类型

子类型也称子界，它是在预定义的类型上加一个约束（如区间约束）。子类型的元素保持其父型的一切特性，并且在赋值时没有必要进行类型转换。也就是说，定义子类型并没有引进新的类型。例如：

SUBTYPE POSITIF **IS** INTEGER

RANGE 1 . .INTEGER′ LAST;

这时，POSITIF是整型，其区间为1至INTEGER′ LAST。INTEGER′ LAST是类型属性。每一个类型T，它的第一个元素叫做T′FIRST，最后一个元素叫做T′LAST。在类型T中的一个元素X有一个后续值T′SUCC (X) 和一个前导值T′PRED (X)。

1.4 实数处理

我们知道在Pascal中仅有的标准数值类型是整型和实型。对于Pascal的分析表明，不能调整数的精度是该语言不可移植的一个重要因素。

其他的语言提供了一些解决办法。如FORT-RAN允许用户选择单精度和双精度，PL/1允许用户规定有效数字的位数。ADA综合了这两种解决办法。对于整型和实型，ADA具有标准型、长型和短型，即：

INTEGER：标准整型，

SHORT—INTEGER：短整型，

LONG—INTEGER：长整型，

FLOAT：标准浮点型，

SHOPT—FLOAT：短浮点型，

LONG—FLOAT：长浮点型。

但我们建议用户不要直接使用这些标准型，而是使用用户定义的派生型。例如可以写成：

TYPE REAL **IS NEW** FLOAT;

如果所使用的机器的 FLOAT 类型精度不够，则可以写成：

```
TYPE REAL IS NEW LONG—FLOAT;
```

ADA提供的第二种解决办法是由用户规定有效数字的位数。例如：

```
TYPE REAL IS DIGITS 10;
```

这时由编译器选择适当的类型（标准型，短型或长型）以保证十位数字的精度。

另外，ADA允许定义实数子界（在Pascal中是不允许的），同时要给出所希望的步长。例如：

```
TYPE NUMBER IS DELTA 0.01 RANGE 10.0 . .
        100.0;
```

定义了由10到100步长为0.01的实数。但是实数子界既不能作为数组的下标也不能作为循环变量。

1.5 复合类型

关于复合类型，ADA同Pascal一样具有数组和记录型。但ADA对复合类型的处理有许多重要的发展。

1.5.1 数组

在Pascal中，数组的维长是固定的，没有可变维长的数组。ADA弥补了这一缺陷。其数组的说明可以是带约束的，也可以是不带约束的。例如：

```
TYPE VECTOR IS ARRAY (INTEGER RANGE
        1 . . 3) OF FLOAT;
```

```
TYPE MATRIX IS ARRAY (INTEGER
        RANGE < >, INTEGER RANGE
        < > ) OF FLOAT;
```

VECTOR的维长是带约束的，而MATRIX的维长是不带约束的。属于MATRIX类型的对象，其维长可以按照以下方法确定：

① 如果MATRIX是一个子程序的形式参数，在调用此子程序时，由实际参数确定它的维长。

② 在用MATRIX这一类型来描述一个下标带约束的对象的同时确定它的维长。或者根据这个对象的初始值推断出它的维长。例如：

```
M:  MATRIX (1 . . 2, 1 . . 3);
N:  CONSTANT MATRIX := ((0.0, 0.0),
        (0.0, 0.0));
```

M为二维，维长分别为2和3。而N为二维，维长为2。

③ 引入MATRIX子类型，确定这个子类型的维长。然后用子类型描述对象。例如：

```
SUBTYPE MATRIX_2_2 IS MATRIX
        (1 . . 2, 1 . . 2);
N: MATRIX_2_2;
```

④ 解决可变维长的另一种办法来源于ADA语言的高度参数化（在带参记录型中还将研究这一问题）。即对于带约束的类型，仍可以写成：

```
TYPE TABLE IS ARRAY (1 . . N) OF
        FLOAT;
```

其中N是一个变量，当然在对说明加工时，它必须有确定的