Ada语言简明教程

麦中凡 译 谢竹虚 校

[美] M. J. STRATFORD-COLLINS 著

北京航空學院出版社

Ada语言简明教程

[美] M.J.STRATFORD-COLLINS 著

麦中凡 译

谢竹虚 校

北京航空學院出版社

内容简介

*Ada*程序设计语言是第四代计算机有代表性的语言之一。主要应用于大型软件系统、各种军事装备上的嵌入式计算机系统、实时系统，而且支持系统开发和分布式应用。尤其是*Ada*能在整个软件生存期中支持软件工程的各项目标，从而有效地降低软件费用。近期内，*Ada*可能成为软件产业中的主导语言，甚至可能取代*FORTRAN*和*COBOL*。

《*Ada*程序设计语言简明教程》是为已经熟悉一门高级程序设计语言的读者编写的*Ada*入门教材。本书内容简明扼要，文字通俗，例题浅显，是一本良好的自学教材。

本书适用范围，计算机软件工作者、计算机专业研究生、高年级大学生、以及从事计算的各学科科技人员、信息工作者。

*Ada*语言简明教程

[美]*M.J.STRATFORD-COLLINS* 著
秀中凡 譯　謝竹虛 校

責任編輯　　肖之中
北京航空学院出版社出版
新华书店北京发行所发行　各地新华书店经售
北京航空学院印刷厂印装
*
787×1092 1/32 印张：7.25　字数：162 千字
1985年11月第一版 1985年11月第一次印刷 印数：1—10000册
统一书号：15432·001　　定价：1.45 元

译 者 的 话

*Ada*程序设计语言最初是美国国防部为摆脱软件费用急骤增长而研制的大型通用的计算机语言。自75年开始研究以来，前后经历约十年时间至83年2月，*Ada*语言文本才正式定型。84年已有第一批商品的系统软件问世。*Ada*是第四代计算机有代表性的语言之一。主要应用于大型软件系统、嵌入式计算机系统、实时系统，而且还支持系统设计应用和分布式应用。尤其是*Ada*能在整个软件生存期（*Lifecycle*）中有效地降低软件费用。这是当今现有高级程序设计语言无法与之比拟的。因此，有人预言从现在到公元2000年将是*Ada*鼎盛时期，她会成为软件产业中的主导语言，甚至有可能在科技计算和数据处理的领域中代替*FORTRAN*和*COBOL*。当然，一个语言的成功与否不仅与它本身性能有关，而且更重要的是看她能否为使用者接受，以及权势的支持。为此，美国国防部这个软件业界最大的用户声称，自84年2月份以后，所有军用软件一律用*Ada*语言作为开发工具，否则不予承认。我们知道，60年代初期*COBOL*就是在美国国防部全力支持下才形成今日统一的通用语言的。因此，可以预期，*Ada*必将得到广泛的应用。

我国计算机科学界和军事科学界对*Ada*的研制一开始就十分关注，并进行了广泛的评论和深入的研究。目前已有几个*Ada*实验性子集问世，正酝酿开发我国正式的*Ada*。因而，一个学习，研制、应用*Ada*语言的高潮即将到来。在国

外这两年高潮已经出现。有关*Ada*的各种教程已有50余种。为了有助于我国关心并渴望及早了解*Ada*的读者，结合当前教学需要，我们选择了本书。

《*Ada*简明教程》原名为"*Ada*: *A Programmers Conversion Course* (*Ada*: 程序员的转换教程)"。为美国加州计算机高级程序设计员*M.J.Stratford-Collins*所撰写。顾名思义，本书是为已熟悉一门高级程序设计语言的人学习*Ada*而写的。内容简明扼要，篇幅仅及*Ada*语言文本之半。文字通俗、例题浅显、清晰地写出了*Ada*语言的各主要特征和*Ada*的思想风格。是一本自学入门的良好教材。

作为软件开发工具的第四代计算机语言，以与软件环境紧密相连为其主要特征。*Ada*是第一个提出环境与语言同时开发的新语言(美国国防部提出的语言需求计划是*STEEL-MAN*，环境需求计划是*STONEMAN*)。然而，本书未涉及这方面的内容。

翻译过程中，对*Ada*新概念的译名我们主要参照了中国科学院数学研究所袁崇义、徐译同同志的译法。有些地方，在行文中感到译名有些不切，因而还参照了南大徐家福、船舶公司709所王振宇等先生在有关文献上的译法。也有个别杜撰之处。由于本人水平有限，不仅译名，译文之中定有许多不妥之处，欢迎读者批评指正。

本书出版过程中，得到北京航空学院出版社大力支持。北航计算机系付主任金茂忠同志百忙中审阅了全稿，在此表示感谢。

麦中凡

1985.2.1于北航

作 者 序

*Ada*程序设计语言是竭尽全力工作的硕果。它最初是由合众国国防部高级语言工作小组于1975年为该部嵌入式系统提供一个单一的程序设计语言而研制的。

本书的目的在于给专业程序员提供一个学习*Ada*基础的方便工具。因此，假定读者已具有至少一种高级程序设计语言的知识(最好是一种象*Pascal*的块结构语言)。本书不是程序设计的启蒙教本，也不象参考手册那样涉及语言的每一细节。而是集中论述多数程序员最常使用的一些特征。

头五章包括基本的语言元素，这和多数现代过程语言没有什么不同。第一章是导论，第二章涉及整个程序流程的控制，第三和第四章是数据类型和数据结构，第五章是过程和函数。第六、七、八、九章分别包括程序包概念、类属概念、对异常处理的支援以及支持任务的语言结构。第十章专门讨论程序结构问题、名字的作用域和可见性。最后一章涉及*Ada*提供的输入和输出设施。为便于参考，附录*A*提供了语言的语法定义(按字母字符序*)，附录*B*列出了*Ada*的保留字表。

我乐于藉此机会向我的家人和朋友们致谢，感谢他们在我整理手稿和成稿期间对我的鼓励和指导。我特别要感谢我的妻子，没有她热诚的支持，本书是不可能写成的。

---

* 为便于我国读者查询，按汉字笔画序—译者

目　　录

第五章　子程序

第六章　程序包

第七章　类属子程序和类属程序包

第一章　基　础

用任何一种语言写的程序，除了最不重要的以外，都是用来处理数据的。于是，如何把那些可能遇见到的各种不同数据对象表示出来并告诉计算机就是十分重要的问题。本章中的材料将给读者提供能被*Ada*程序处理的基本元素（标识符、简单数据类型、字面量和常量）的知识。在结尾的一节中将介绍“作用域”和“可见性”的概念，以及它们和程序结构的关系。

1.1 起　点

我们从分析一个简单的程序段开始，该程序段的目的是（从普通的输入装置）读入一个用华氏温标表示的读数，把它转换为摄氏温标，并（在普通的输出装置上）写出转换结果。

```
1    declare
2         Celsius, Fahrenheit : float;
3    begin
4         get(Fahrenheit);
5         Celsius:=(Fahrenheit-32)*5/9;
6         put(Celsius);
7    end;--程序段完
```

第一行的'**declare**'(声明)标志着这个程序段或程序块的开始，其目的在于通知编译本程序块中使用的各个变量声明将在下面出现。第三行'*begin*'(开始)标志着各个可执行语

句的开始，并与第七行终止它的 '**end**'（结束）成对出现。**begin-end** 可以看作是括着语句序列的一对括号。

第二行是"声明"语句。它的功能是建立名为"*Celsius*"和"*Fahrenheit*"的变量。它们的值当前都是不知道的。它们的"类型"包含在每一变量声明之中。类型的概念和许多现代语言(如*Algol*和*Pascal*)是一样的，并且是*Ada*语言最重要的特征之一。此外，它允许编译检查程序员是否给变量赋了一个不同类型的值，比如把一个'整型'数赋给了'字符'变量。

第四、五和六行是进行实际计算的行。第四行引用一个称之为 '*get*' 的通用系统模块，它读入一个值到变量*Fahrenheit*之中，*Fahrenheit*读到一个值之后，我们就按照某个公式将它转换为摄氏温标的一个数，再将其结果放在名为 *Celsius* 的变量之中。这是靠第五行简单"赋值"语句得到的。这条语句先求出赋值号（即":="）右边的表达式的值并"赋给"左边的变量，在此处是*Celsius*。新的值将冲掉原先存放在此处的任何值。

现在剩下的事是在第六行打印*Celsius*的值。第七行标志着由第一行'**declare**'所开始的程序段到此结束。注意，此行有一个注解。注解字符串从一对字符"--"开始，随本行结束而终止。注解被编译略去。

关于此例还有一个基本要点值得重复。*Ada*要求程序员用声明语句定义他将要使用的**每一个**常量和变量。所要求的信息是名字，或更正确地说是'标识符'，以及'类型'。类型指示编译如何去表示这种新的数据对象。*Ada*不支持也不允许象 *Fortran* 中所用的那种缺省声明(*default declaration*)。在使用常量和变量之前一定要程序员去定义它们所有

的属性，这样可以消除许多产生错误的潜在可能性，同时，编译时刻(*compile-time*)和运行时刻(*run-time*) 的核查也可以在较大范围内进行。

1.2 常 量

在第二、三行我们建立了可以对它们进行读、写的变量。在某些程序中，我们可能希望定义一个数据对象，它的值是知道的，而且在任何情况下该值都不改变。然而，一旦需要；在以后的日子里我们又可以很容易地改变它。库存品管理程序中就有这样的例子，该程序当库存品少于10件时自动打印订购一定数量货物的订货单。在此程序中，我们可以定义一个常量如下：

reorder_elephants : **constant** *integer*:=10;

此后就把再订货的数量叫做"*reorder_elephants*."如果将来某个时侯，我们希望手头至少保有15件，则我们只需改变一项，即在声明中再指定该常量的值，并把程序重新编译一次。与此对比，若从几千行程序正文的扫描中找数字"10"，并判定它们是否与再订货限量有关，然后将有关的改为"15"，这样，你就会看到使用常量的好处。要注意变量和常量的一个主要的差别，即一旦常量得到定义则永不再改写它（它们从不出现在赋值号的左边)。这再一次消除了出错的一个可能来源，因为若把一个语句写成：

reorder_elephants:=*first_of_month*;

而程序员的本意是写：

reorder_elephants_date:=*first_of_month*;

则编译就指出语句出错，因为被定义的常量其值是不能更改的。

尽管我们讲了许多使用常量的好处，然而也存在别的情况，在那种情况下我们绝对无误地知道一个值不会改变。比如，1公里就是1000米，从华氏转为摄氏的转换因子总是个定数。因而，将它们声明为常量就没有什么特别的优越性，我们只要在需用它们的地方把它们直接写到程序中就可以了，我们的常识会确保我们不会再去改变它们。按这种方式使用的值称为'字面'量，比如，在我们的例子中"32"，"5"、"9"都是字面量。在随后的章节中讨论各种数据类型时，我们将会更详细地看到如何使用每种类型的字面量。

1.3 字符集

对于一个特定的*Ada*编译，其允许使用的字符集要取决于实现，不过语言定义要求所有的实现至少都要提供如下字符集：

```
A B C..X Y Z
0 1 2..7 8 9
'#&'()* +, - ./ : ; <=>_|
空白字符
```

我们假定本书中*ASCII*字符集的其它字符也可用，则在我们的集合中还要加上：

```
a b c..x y z
! $ % ? @ [\] ^ ' { } ~
```

最后，我们还假定字符集的顺序是：

```
0<1..<9<A<B..<Z<a<b..<z
```

虽然此处给出的顺序是最常用的一种，但现存系统各有差异。如果你有怀疑，請参考你们那里的编译实现指南，查看所使用的字符顺序。同样，如果你对你们那里能用的字符集不清楚，請参阅编译文挡（在名叫*STANDARD*的程序包之中）。

1.4 标识符

标识符是一个字符序列，用以表示被声明的程序元素（如常量、变量、子程序等等）。标识符按以下规则生成：

（*a*）标识符必须从字母开始。

（*b*）后续字符可以是字母、十进制数字或下横线符。

（*c*）标识符可由任意多个字符组成。

（*d*）若允许用小写字母，则大小写字母沒有什么区别，作为同一字符对待。

为了描述这些规则，用一种巴库斯_瑙（*Backus-Naur*）范式的变体作为速记的方便形式，如下所示：

标识符::＝字母{[下横綫]字母_数字}
字母_数字::＝字母｜数字
字母::＝大写_字母｜小写_字母

小写字母所写的词（可能包括下横线符）表示的语法项目和大写_字母的一样。花括号{ }表示括在其內的部份可以重复任意次，包括零次。方括号[ ]表示括在其內的实体是可选的。下面的例子用来说明这些规则：

today　　--正确
tomorrow　　--也正确