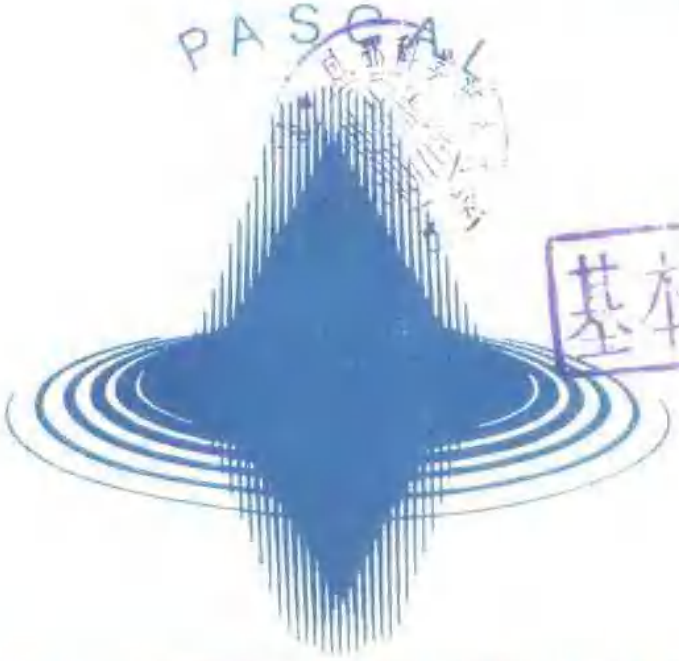
怎样使用最新 PASCAL 语言

邓晓玲　王燕霞　邓鸿斌　译
杨铭霞　校　阅



陕西电子编辑部

怎样使用最新PASCAL语言

邓晓玲　王燕霞　邓鸿斌　译
杨铭震　审校

陕西电子编辑部

前　言

MODULA—2是瑞士苏黎士联邦技术学院（ETH）的Niklaus Wirth教授创立的一种新的程序设计语言，它是在PASCAL的基础上建立的，它包括了PASCAL中的全部概念，并且在有的方面发展了PASCAL。尤其重要的是MODULA—2包含了有多道程序编程能力的“模块”的重要概念，利用灵巧的方法编制的低级语言实现的方法，总之MODULA—2可以象普通程序语言那样使用，同时又象一种系统执行语言，MODULA—2可以用到出高质量的软件的方法来改善程序，实际上对于达到程序化、可靠、清晰、可扩展、可多次使用及可移值的目的，模块化是一种很有力的工具。

本书描述了MODULA—2从开始到完成的整个过程，每个题目都用明确的例子来描述每一个概念也都被证明是合适的，该语言中的句子都可用语法图来解释。

本书虽不是MODULA—2的参考手册，但却是一本能使学生循序渐进地学会这种语言的教科书，书中对最主要的概念（即过程、模块和数据结构）都解释得非常仔细，同时讲授方法也很好。

在第一章的开头，学生可以在不同的机器（Apple—Ⅱ，IBM/PC或VAX11/7801上运行自己编的程序。这样学生可以得到一些基本的概念。

在每一章的结尾，给出一个小结，以便使读者记住所讨论问题的主要概念。

最后还有两个附录，第一个用语法图描述一些完整的句子，并发展为Backus—Naur形式，第二个则是MODULA—2的关键字的清单和标准模块。

译 者 的 话

MODULA—2是瑞士N·Wirth教授于1977年设计1980年发表的计算机程序设计语言。这种语言是根据PASCAL语言发展而来的。

如同FORTRAN，COBOL，BASIC等高级语言一样，PASCAL高级语言已广为大学生和计算机科学工作者所喜爱，因为PASCAL语言具有小巧、简洁、结构性好、移植容易等特点，尤其是它与计算机的基础理论（离散数学、数据结构等）结合密切，因此，目前全国大部分高校都将PASCAL语言作为计算机专业和相近专业的语言课。PASCAL语言在科研和生产部门也得到了广泛的应用。但是，由于受计算机领域发展形势的限制，PASCAL仍存在一些问题。主要表现在：

（1）PASCAL语言没有模块概念，PASCAL语言是70年代初期设计的，而模块概念是在70年代中期程序趋于大型化时发展的。

（2）PASCAL语言缺乏可用于多道程序的设施。

（3）PASCAL语言没有对机器内部存贮器进行访问的低级设施，给系统程序设计带来一定的困难。

（4）PASCAL语言没有引入过程类型。

（5）PASCAL语言没有动态数组，因为PASCAL的数组大小只在编译时决定，这就难于进行字符串操作。

（6）PASCAL语言缺少分离编译设施，有碍于程序的开发。

（7）PASCAL的文件处理功能不完全，尽管PACAL是讲授程序设计的最好语言，但对重要的随机文件访问技术不能用它讲授。

MODULA—2语言解决了以上问题，MODULA—2的模块概念的出现，使得程序易读，简明，隐蔽性好。深信读者读了这本书后，会感到MODULA—2语言基本解决了PASCAL语言存在的问题，使得计算机领域发展了，这是顺乎潮流的。

目前，国内已经有人开始对MODULA—2进行研究，但知道MODULA—2的人还不多，国内也还没有一本书对MODULA—2作系统的介绍，更没有一本MODULA—2的教材。我们现在将加拿大莱特利尔大学Daniel Thalmann先生编著的“MODULA—2导论”一书介绍给国内的广大计算机工作者和大专院校的师生，希望这本书和MODULA—2一样成为计算机领域中程序设计的好工具。

这本书对MODULA—2作了详细的分析和讲解，在国外是一本很好的教材。它使读者对MODULA—2有一个系统的完整的了解，本书所举的例子典型，易读。每一章都有一个小结，以概括本章的主要内容。附录一采用Backus—Naur法则来描述语法关系和交叉索引。使人感到掌握MODULA—2语言的句法有规律性。附录二用图表形式来描述语

法，有通俗易懂之感。

本书共21章，第1～17章由邓晓玲同志翻译，前言部分、第18、19章由王燕霞副教授翻译，第20、21章和两个附录由邓鸿斌同志翻译。杨铭震副教授审校。

由于水平有限，时间比较仓促，本书一定有不少缺点和错误，敬请有关专家、教授，计算机工作者和广大读者批评指正。

译者

1987，12，2

目　录

第一章 概 述

1.1 程序的作用和起源

计算机是一个由许多电路组成的一部电子机器。机器的实体叫硬件，这是一种模拟没有头脑的人的常用说法。实际上，只有硬件是有局限性的。为了使其更有用处，计算机必须有程序，它必须是一种能被采用和便于理解的指令，而且许多基本的指令必须事先存放在计算机内，这些基本的指令就是我们经常所说的计算机的基本软件。计算机的指令不能随机地、没有逻辑和规则地加进计算机中。事实上，计算机的程序需要使用一种语言，通常把它们叫做程序语言。

早在1801年Frenchman Jaccard发明了一种织布机，它能根据编好的码盘上的码在所织的布上编出某一种花型，这个织布机就具有一个有一组指令的程序，并能在没有人参与时正常工作。

在十九世纪初，一个叫Charles Babbage的英国人，他设计了一台机器，这就是迄今被认为的第一台计算机。这部机器叫分析引擎，它的存储能力为1000个数字，程序由编码而产生，为这个分析引擎而编写程序的第一个人是Ada Augusta，Countess of Lovelace和诗人Byron的女儿，实际上，Babbage的实验因缺少经费而失败了，但是，他的思想正如1884年Herman Hollerith建立的Census机器一样是以下两种计算机发展的雏型，Harvard Mark I 和ENIAC 这两种机器由于输入一个程序时必须重新布线，因而很难于使用

在计算机的发展中最重要的一点无疑是存储程序的概念，这个概念是两个著名的数学家Alan Turing和John Louis von Neumann的工作所建立的。主要思想是：由于一个程序能够进行数字编码，所以它也可以象数据一样保存在内存中。在存储器中主要存储的是数据，直到今天，计算机仍然是建立在存储程序的概念上，因而必须研究程序语言设计中的发展。

1.2 程序语言的发展

最初的计算机是用二进制语句来编程的，计算机具有会话功能，自然工作十分艰苦，误差又很明显，为了改善这种状况，在1954年John Backus和他的设计组设计了一种符号程序语言，称为FORTRAN（FORmula TRANslation），这种语言主要适于科学计算，因为计算机只能理解（二进制的）机器语言，因而就必须研究从FORTRAN到机器语言的翻译语言，这就是所谓的编译程序，它在计算机科学中起了非常重要的作用。FORTRAN在科学家中获得了很成功的应用，而且直到今天虽然它已经了七次重大的修改，仍是一种十分重要的语言。1958年，计算机翻译成功，它被称为计算机的“第二个伟大成就”，从此计算机科学工作者开始将“硬件和软件”的术语引了进来，计算机开始和主要由编译程序和操作系统等组成的软件一块销售。1959年在商务部门的要求下，美国政府使

计算机用户和制造者组织起来设计商用程序语言，这种语言于1961年问世，被人们称为COBOL语言（COmmon Business Oriented Language）。COBOL没有FORTRAN的那种数学计算的能力，但它改善了分层记录的概念，加到已在FORTRAN中使用过的数组之中。控制语句也更加结构化（IF…ELSE，PERFORM…UNTIL）并且文件处理也迅速地得到发展。

FORTRAN和COBOL不能完全满足要精确计算的计算机科学工作者的要求，他们于1960年设计了一种新的语言叫ALGOL，这是现代所有分块语言的雏型，在语言中引入了一些重要概念，诸如递归、参数传送方式和局部变量等等。但非常遗憾，ALGOL没有得到广泛的应用。

在另一方面，麻省理工学院的JohnMcCarthy在文件处理的基础上设计了一种叫LISP的新语言，这种语言开始具有人工智能的功能，直到今天也还广为流行，尤其是在专家系统中，它给予别的语言很大的影响，如被日本人所选中的在他们的第五代计算机研制中所使用的PROLOG语言。

世界上最大的计算机制造商IBM公司还成功的设计了两种语言：APL和PL/1语言。APL是一种应用非常紧凑的符号语言。PL/1语言意图在集FORTRAN、ALGOL和COBOL之大成，另外还它有两种新的概念：免除了修改的设备和多道程序的设备，PL/1只在IBM的机器上使用。

PL/1是非常完善但复杂的语言，用与此相反的目的，Dartmouth学院的John Kemeny和Thomas Kurtz提出一种非常简单但受一定限制的BASIC语言，这种语言没提出任何一点新的语言的概念，可以说这是第一种真正的交互式语言。

在六十年代的后期，有些计算机工作者力求改善程序。结构语言的思想非常流行，主要因为在软件的发展中出现了危机，程序迅速地都编得很大，同时也越难读和错误越多。这时发展的主要思想是强调数据结构，top—down程序和设计更好的控制结构，特别是Dijkstra着重指出goto语句的问题所在，在这种思想指导下设计出了一些新的语言，其中最好的三种是：ALGOL—68，SIMULA—67和PASCAL语言。

ALGOL—68是一种非常突出的语言，它由一个非常完善的报告定义。一些诸如“正交性”等有趣的概念也引了进来，它意味着语言包含有可以分段了解和与别的特点在相交处独立出来的一些基本特点，由于ALGOL—68过于复杂而使用得很少。

SIMULA—67是一种与ALGOL相类似的语言，设计开始是为了用于仿真。SIMULA—67的最重要特点是分类概念，这是数据分离的雏型，分类就使程序可做简略的说明，而过程又是在一个简单的组织中。

Niklaus Wirth设计的PASCAL提出一种出色的结构，由用户确定数据类型和正交性。因为语言相对简单，编译系统发展很快，PASCAL是当今在大学中非常流行的语言，而且适于在大多 微型机和个人计算机上使用。

在七十年代，程序语言主要沿着两个方向发展：数据分离和并行。

数据分离的思想用来设计数据结构及在其上的运算，但实现它们需要隐含码。SIMULA的“分类”提供了设计数据分离的一种方法，这个概念也参考了诸如EUCLID、CLU

和GYPSY等一些其它语言。另外这种概念是通用型的对象组织语言，如SMALLTALK就是这样设计的。在SMALLTALK中，每件事都是一个对象和别的对象仅仅在息信上的通讯。

在Dijkstra和Hoare工作的基础上，新的程序语言被设计成具有并行处理的能力和实时的程序。用监控的概念可介决问题处理间的同步问题，这个概念是Brinch Hansen在并发PASCAL和Wirth在MODULA中提出来的。

能在一种语言中同时包含现代的几乎所有的近代思想的语言就是ADA语言（取名以表示对Ada Augusta, Ceuntess of Lovelace的敬意），这种语言是美国国防部研制的，ADA是建立在PASCAL的基础上的，它包括易于数值计算、并行处理、数据结构和优良的管理功能。

现在发展成MODULA—2，它是Wirth于1977年设计的，它包含了PASCAL中所有的概念，并在它的基础上作了某些改进，例如加强了结构化、多道程序能力以及用一灵巧的方法设计的低级语言所能完成的功能等概念。MODULA—2也和其它一般用途的语言和系统工具语言一样是能易于调整的语言。

1.3 “模块”（MODULE）概念

任何一种程序语言最终的目的要有以下三个性质：模块化、机器兼容性和可靠性。

模块化是指能将一个大的程序分为许多部分，它们易于在不同的程序中并接，这表示必须很好地定义各个不同部分间的接口。

MODULA—2为了实现模块化，用很强的工具来改善程序，象在SIMULA—67中一样，对象和过程可以装在一个模块中，有时一个模块又分为两个部分：一个定义部分和一个实现部分，定义部分正如满足它与其它模块连接的接口的需要，而实现部分具有在单个程序中的可实现性和可靠性。

机器兼容性是一种适应能力，任何一个程序在某一机器和某一操作系统支持下会具有它自己的特征而难于转到别的机器上，在模块的实现部分使用了排除机器的不可兼容性的概念，则MODULA—2可使机器的不可兼容性明显地下降。

MODULA—2是一种已定义得很好的语言，因而程序经常是能扩充的，依所写的一些模块，在没有改变它的基本结构的情况下，很容易扩充它。

1.4 词法、句法和语义分析

MODULA—2是一种高级程序语言，虽然它允许含有低级语言，但它对主计算机的机器语言来说是完全不同的。这就意味着任意一个MODULA—2的程序必须借助叫做编译程序将它翻译为机器语言。编译程序应完成以下任务：

（1）它读入程序并发现词法错误，例如它辨认PROCEDURE这个字时，可以认定它是在MODULA—2的词典中的，但PROZEDURE却不然。

（2）它要检查程序的句法是否正确。为此它必须检验是否违反了语言中的任何一句法规则。例如 任何一个表达式都是左括号在先，后面必须有右括号。

（3）它建立了在程序中使用的符号的表并检查算子的应用对几个相同类型的对象是否兼容，以便于应用。例如一个整数和一字符串的加法运算是无意义的。这样的分析叫词意分析。

（4）它产生的机器码和原始程序（核心码）相对应，并且这个机器码应是优化了的。

为了实现上述的分析，（词法、句法和语义）编译程序必须知道所有的语言规则，用户也必须学习这些规则以避免出错。这就意味着程序语言的任何一个描述都要强调词法、语法和词意的规则及定义。对于MODULA—2我们将非正式的叙述它的语义规则，因为在描述语言的词意时没有一种简单的方法，词法规则非常有限，我们将在下一节中讨论，在.6中将讨论句法规则。

1

1.5 MODULA—2词汇

在MODULA—2词汇中，有六种符号：

（1）标识符

一个标识符是由字母和数字组成的，不包括其它字符（如短线、空格等），第一个字符必须是一个字母，大写、小写均可，有如下区别：

合法的例子：Vanessa　MyDaughter　James Bond007

非法的例子：7Dwarfs　New York

一个标识符可以受另外一个标识符限制：在这种情况下，标识符间要用园点分开

例：Person.age　　Cow.Tail

（2）数字

可以有两种数字类型，它们将在2.1中讲到。

（3）串

一个串是多个字符的串连，用双引号或单引号括起来。

例。"Hello"　　'Peace or war?'

一个被双引号括起来的字符串，其中不应再含有别的双引号，同样单引号内也不允许再含有单引号，例如，下面几个是正确的字符串：

"It's you"

'He said: "you are stupid"'

（4）专用字符

专用字符用来指示操作和所受的限制，它们可以包括一个字符（例如：+ * ( ) 或者可以包括没有空格的两个字符（例如：〈＝ : = ）

（5）关键字

MODULA—2包括一些确定的关键字，它们不能用作标识符，它们都用大写字母书写，并且在语言中有专门的作用，图1.1是MODULA—2所用的关键字清单

（6）注释

注释是由字符组成的，用括号（*和*）将注释包括在内，它们仅仅用来帮助程序员理解自己的程序（或使别人易于理解而加的符号）。

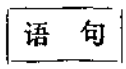
例：（*这个注释是专为你设置的*）

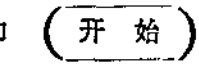
AND	ELSIF	LOOP	REPEAT
ARRAY	END	MOD	RETURN
BEGIN	EXIT	MODULE	SET
BY	EXPORT	NOT	THEN
CASE	FOR	OF	TO
CONST	FROM	OR	TYPE
DEFINITION	IF	POINTER	UNTIL
DIV	IMPLEMENTATION	PROCEDURE	VAR
DO	IMPORT	QUALIFIED	WHILE
ELSE	IN	RECORD	WITH

图1.1 关键字的清单

1.6 语法图

为了描述MODULA-2语法，我们将使用语法图，一个语法图采用框图方式绘出，有两种框图：

（1）矩形框：即 

（2）椭园形框：即 

矩形框代表非终了的符号，椭园框则是终端的符号，终端符号作为一个基本的符号而被确定，它没有专门的语法，终端符号对应于前述的专门的字符和关键字，非终端符号则是可以用语法规则来描述的符号，对任意一个非终端符号，均可用语法图来描述。

箭头的作用是指出符号间是怎样连接的，例如，符号可以是任意的或可重复使用的。为了了解语法图是如何构成的，可看下面的简单例子，设我们有一种语言，它仅允许我们从下列清单中使用一个或几个句子：

```
Melanie is nice
Vanessa is nice
Melanie is funny
Vanessa is funny
```

这些句子可以任意次重复地使用，这个非常有限的语言的语法规则可用语法图描述如图1.2所示。

请注意，在这些图中，有一个是顺序图（句子），一个是重复的图（族），和两个局部的图（girl，what），改变它们的组成是完全可能的，如图1.3所示。MODULA-2的全部语法将在本书中均用语法图给出。

1.7 MODULA—2 程序結构

一个MODULA-2 程序，实际上是一个模块，它是用主模块和用名字或标 识 符来说明的，图1.4给出了一个模块的语法图。

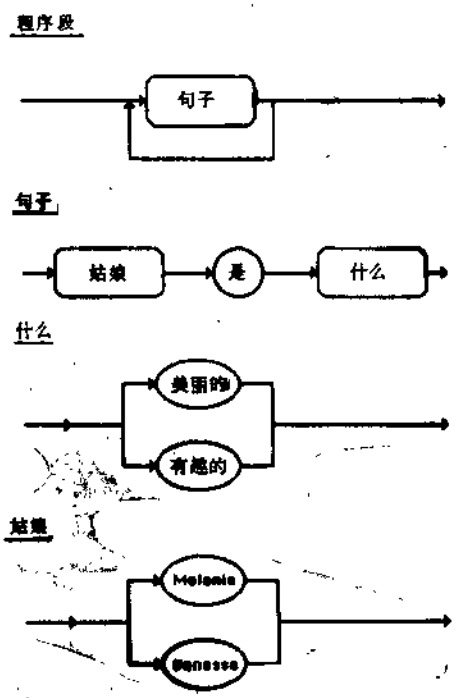


图1.2 最简单语句的语法图

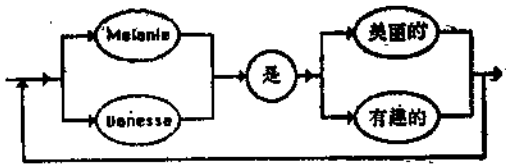


图1.3 一个比较复杂的语法框图

图1.4 一个模块的语法图

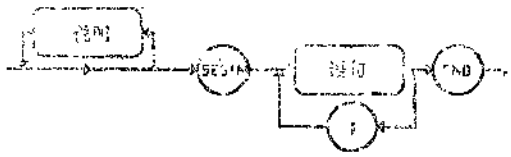


图1.5 一个分程序的语法图

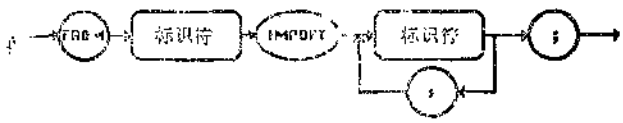


图1.6 输入子句的语法图

```
MODULE Welcome;
    FROM InOut IMPORT WriteString;
BEGIN
    WriteString( "Welcome to MODULA—2" )
END Welcome.
```

图1.7 第一个程序：欢迎模块

模块的名字必须在模块的结尾重复，并且主模块必须要用句号（。）来结束。当我们在下一部分需要它时，另外的模块可以用import命令把它们取出来。基本的模块包括一个分程序（见图1.5），包括说明和语句程序段，语句程序段用关键字BEGIN和END括起来。说明和语句将在以后的章节中讨论。

需要注意的是：语句和其它子句用分号分开，在第一次准确地理解一个分号时，可能比较困难，但是，语法框图总可给出问题的解答。

1.8 第一个程序—如何写一个字符串

作为第一个程序例子，我们将编写一个程序以在终端上写下面的文本：

Welcome to MODULA-2

我们称这个模块为Welcome，同时需要了解称为串的字符序列。这样，我们的串是：

"Welcome to MODULA-2"

现在需要一个操作，它能使我们在终端上写出这个串。这种操作没有定义在MODULA-2语言中，但是一个被称作InOut的标准模块包括这种操作，并且MODULA-2语言可以用从模块InOut产生的输入操作来扩展它。这样的操作称为WriteString时，输入