PASCAL
程序设计

郑启华 编著

清华大学出版社

PASCAL 程序设计

郑启华 编著

清华大学出版社

内 容 简 介

本书全面地介绍了PASCAL语言的数据类型、语句及结构特点，系统地讲述了程序设计方法，特别是自顶向下逐步求精的结构化程序设计方法，并强调培养良好的程序设计风格和习惯。讲述力求理论联系实际、深入浅出、通俗易懂。

全书共分十二章。第一章介绍必要的基本知识。第二章到第五章介绍结构化程序的四种基本结构（顺序、选择、重复、函数与过程）的设计方法，以及PASCAL的标准数据类型和基本语句。第六章到十一章介绍PASCAL的各种用户定义数据类型（枚举、子界、数组、集合、记录、文件和指针）及其程序设计。第十二章介绍余留问题。每章后面均附有习题。

本书可作为高等院校计算机软、硬件专业或其它专业的程序设计教学用书，也可作为计算机应用科技人员的自学或培训教材。

PASCAL 程序设计

郑启华 编著

☆

清华大学出版社出版

北京 清华园

北京京辉印刷厂印刷

新华书店总店科技发行所发行

☆

开本：787×1092 1/16 印张：17.5 字数：436千字

1991年7月第1版 1991年7月第1次印刷

印数：00001～20000

ISBN 7-302-00822-1/TP·297

定价：4.95元

前　言

PASCAL 语言是由瑞士的沃斯（N.Wirth）教授于1971年提出来的。它的命名是为了纪念波兰数学家Pascal。

PASCAL 语言的建立基于两个主要目的：第一，提供一种能够清晰、自然地表述某些基本概念的语言，使其成为基本概念系统训练的工具，适合于程序设计教学。第二，使新定义的语言能在现有计算机上可靠、有效地加以实现。

PASCAL 语言是系统地体现由戴克斯特拉（E.W.Dijkstra）和霍尔（C.A.R.Hoare）定义的结构程序设计概念的第一个语言，因此它是程序设计语言发展史中的一个里程碑。由于它结构清晰、便于学习和有较丰富的数据类型和语句，而且编译、运行效率高，便于移植，它已广泛地用于大学程序设计语言的教学和许多应用软件、系统软件的开发中。

本书是作者在长期教学经验的基础上，参考国内外有关教材编写的。它不仅可以作为大学计算机软、硬件专业和其它专业的程序设计教学用书，还可以作为工程技术人员的自学参考书。

本书不仅全面介绍了标准 PASCAL 语言，而且通过介绍 PASCAL 语言讲述了自顶向下逐步求精的结构化程序设计的基本思想和基本方法。学习程序设计必须理论联系实际。本书在介绍基本概念的同时，列举了大量典型而有意义的例题和习题。读者通过阅读这些例题和自己动手完成习题，并尽可能地上机通过，就一定可以学好程序设计。本书还强调培养良好的程序设计风格和习惯，这对于一个优秀的程序设计者是很重要的。

全书共分十二章。第一章介绍了必要的基本知识。第二章到第五章介绍了结构化程序的四种基本结构（顺序结构、选择结构、重复结构、函数与过程）及其设计方法，此外还介绍了 PASCAL 的四种标准数据类型和一些基本语句。这四章是本书的核心，掌握了这四章的内容也就掌握了程序设计的基本方法。第六章到第十一章介绍了 PASCAL 的各种用户定义数据类型（枚举类型、子界类型、数组类型、集合类型、记录类型、文件类型、指针类型），它们的引入扩大了 PASCAL 的应用领域。通过这几章的学习，不仅能了解 PASCAL 的丰富的数据类型和处理语句，而且对前几章所讲述的程序设计基本方法是一次新的应用和提高。第十二章是一些余留问题。在每章之后都附有一些精选的习题。书末列有 5 个附录。

为了配合读者学习本书，我们另编了一本《PASCAL 程序设计习题与选解》（杨德元等编著）。该书已由清华大学出版社出版。本书如有不妥之处，希望读者批评指正。

本书由清华大学计算机系郑人杰和谢若阳两位老师审阅，他们提出了许多宝贵的意见，作者在此向他们表示感谢。

郑启华

1989年于清华大学

目　录

第一章　计算机和程序设计介绍

1.1　引　　言

本章将介绍计算机和程序设计的基本知识，包括计算机的组成、程序和程序设计语言、PASCAL 介绍等内容。通过这些内容的学习，将使我们对计算机和程序设计有一个初步的了解，为以后各章的学习打下必要的基础。

计算机是电子数字计算机的简称。从世界上第一台电子数字计算机诞生到现在不过几十年的时间，计算机经历了四代，现正进入第五代。计算机已应用到工业、农业、商业、文教、军事、科学和人民生活各行各业。

计算机基本上由四个部件——存储器、中央处理器、输入设备和输出设备组成。计算机只能按照人编写的程序进行工作。最基本的程序设计语言是机器语言。PASCAL语言是一种高级程序设计语言，较适合于人的使用。

1.2　计 算 机 组 成

1.2.1　引言

计算机是表示和处理信息的工具。有许多不同类型的计算机。例如从尺寸上说，有握在手里的袖珍计算器，也有占满几个房间或整幢大楼的复杂的计算机系统。从前计算机很昂贵，以致它们仅用于商业和科学计算。现在已经有了用于家庭的个人计算机。

计算机的尺寸和价格通常依赖于它在给定单位时间内所能完成的工作量的大小。大的贵重的计算机具有同时执行许多运算的能力，它们有更多的设备，以便执行一些特殊的功能。所有这些增加了它们的性能和价格。

1.2.2　计算机部件

尽管在价格、尺寸和性能上有很大的不同，现在计算机在许多方面是非常类似的。基本上，一个计算机由图 1.1 所示的四个部件组成（连接各部件的线表示可能的信息流路径，箭头指示了信息流的方向）。

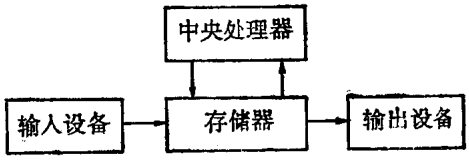

图 1.1　计算机基本部件

由计算机处理的所有信息必须首先通过输入设备送进计算机存储器。在存储器中的信息由中央处理器处理。这个处理结果也存储在计算机存储器中。在存储器中的信息可以通过输出设备输出。这几个部件和它们的相互作用将在以下几节更详细地描述。

1.2.3 计算机存储器

计算机存储器可以看成是存储单元的有序排列。每个单元与一个特定的地址相联系，地址指出了该单元在排列中的相对位置。图 1.2 描述了一个包括1000个单元，从 0 到 999 连续编号的计算机存储器。某些大型计算机的存储器可能包括一百万个单元。

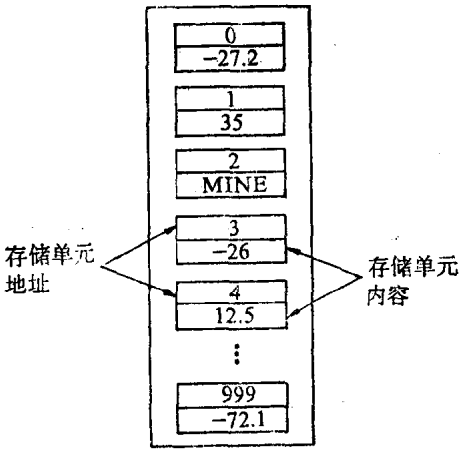

图 1.2 具有1000个单元的计算机存储器

计算机存储单元用来存放信息。各类信息——数字、名字、表甚至图都可以在计算机存储器中表示。存储在存储单元中的信息称为存储单元的内容。每个存储单元包括某个信息，没有单元是空的。而且一个单元只能包括一个数据项。当一个数据项放进某存储单元时，原来存放在该单元中的信息消失，且不可恢复。在图 1.2 中，存储单元 3 中的内容是数－26，存储单元 4 的内容是数12.5。

值得注意的是，处理数据的指令也存储在计算机存储器中。这些指令由中央处理部件检查和执行。

1.2.4 中央处理部件

计算机的信息表示能力是无关重要的，重要的是计算机的处理能力。它使得我们有可能解决那些以前因为计算量太大而无法解决的问题。

计算机处理能力的核心是中央处理器（CPU）。CPU可以从存储器取信息，这个信息可以是数据或者是处理数据的指令。CPU也可以为了以后的引用，将处理的结果回送到存储器中。

CPU协调计算机各部件的所有动作。它决定应该执行哪些操作，和以什么次序执行。CPU的功能是传输同步控制信号和命令。

在中央处理器中还有一个算术-逻辑部件。算术部件由执行各种算术操作(包括加、减、乘、除）的电子线路组成。一秒钟它大约可以执行 100 万次这种操作。逻辑部件由比较信息和根据比较结果作出决定的电子线路组成。计算机与普通的袖珍计算器的区别正是这个特性以及有效的存储特性。多数计算器只可用于对数字执行算术运算，而不能比较这些数字，作出判定和存储大量数据。

1.2.5 输入和输出设备

如果我们不能与计算机通信，计算机的处理能力对于我们是没有多少用处的。特别是，我们必须能够将信息输入到存储器中和显示存储在计算机存储器中的信息（通常是处理结果）。输入设备用于输入数据到计算机存储器。输出设备用于以可读的形式显示结果。

有许多类型的输入和输出设备。输入设备包括卡片读入器、纸带读入器和计算机终端。最通常的输入设备是卡片读入器。它读一迭轻的穿孔卡片。穿孔卡片可以包括至多80行信息，每行可以包括一个字符（字母、数字或标点符号等）。每个字符由一行上的特定的一组孔代

表。卡片由一个标准的卡片穿孔机穿孔。

通常的输出设备是打印机。这种设备通常每分钟能打印2000行或更多行。每行可以打印120到132个字符。穿孔卡片也可以由计算机输出产生。另一种输出介质是绘图仪，它以图形显示计算结果。以上提到的输出设备均能产生计算结果的永久性的、直接可读的记录。

还有一种既作为输入设备又作为输出设备的计算机终端。终端通常包括一个类似于打字机的键盘。计算所要求的信息通过键盘打入。计算机结果以字母或数字形式打印在终端打印纸上或显示在终端屏幕上。某些终端具有图形功能，它能将输出显示成二维图形，而不只是字母、数字行。借助于图形设备，用户还可以用光笔指示屏幕上显示的信息，与计算机通信。

计算机终端广泛用于票证预定计算机中，以便形成预定和打印票证。它们也用于百货商店结帐计算机中，以便帮助保存顾客购买商品的记录和计算输出盘存。

在许多计算机系统中，有一类输入输出设备用于提供附加的信息存储和恢复（次级存储）。这些设备可以在主计算机存储器和磁存储介质（例如磁带、磁盘或磁鼓）之间传输大量信息。可以将原来的或新得到的信息保持在二级存储设备中，以便将来恢复和使用。

有两类存储器的原因是，快的主存储器太昂贵。次级存储器较慢但是较便宜，它适于存储不经常使用的大量数据，这些数据必须传入主存储器才能被处理。

1.3 程序和程序设计语言

1.3.1 引言

信息（输入数据）可以存储在计算机存储器中，并以相当高的速度被处理，以产生结果（程序输出）。我们通过提供执行的指令清单（程序），向计算机描述一个数据处理任务。一旦指令清单提交给计算机，计算机就可以执行这些指令。

产生指令清单（写程序）的过程称为程序设计。写一个计算机程序类似于向一个不曾玩过某种游戏的人描述游戏规则，要求描述语言为通信双方所理解。例如，游戏规则必须以某种语言描述，然后阅读和执行。游戏的发明者和希望玩游戏的人都必须熟悉所用的描述语言。

人和计算机间通信所用的语言称为程序设计语言。提交给计算机的所有指令必须按照程序设计语言的语法规则描述和连接（形成一个程序）。然而程序设计语言和自然语言（例如中文、英文、法文）有一个很大的差别，即程序设计语言的规则是很严格的，没有“例外”或“含糊”。这是因为计算机不会思考，它只能严格地遵循所给的指令行事，而不能解释这些指令，以猜想出程序编者（程序员）想让它干什么。在写出的指令中的错误将改变程序的原意，引起计算机执行错误的动作。

本书将集中研究PASCAL程序设计语言。该语言是由瑞士的N.Wirth教授开发的，它具有许多先进的特性，可以简化不同应用领域中的程序编写工作。

大多数计算机不能直接执行PASCAL程序，而必须首先转换成计算机能懂的语言——机器语言。机器语言中的指令由运算符的数字代码和数据的数字地址组成。将PASCAL语言转换成机器语言是由一个专门的程序（编译程序）来执行的。如果转换是成功的，程序的

机器语言版本就存储在存储器中，并被执行。图 1.3 说明了这个过程。

图 1.3 PASCAL程序的编译

以PASCAL语言而不是以机器语言进行程序设计有两个主要的优点。首先，PASCAL语言比机器语言更接近于自然语言，因此写PASCAL程序更加容易。其次，PASCAL程序是高度可移植的，为一种计算机写的PASCAL程序常常可以在各种计算机上执行；而为一种计算机写的机器语言程序通常不能在不同类型的计算机上执行。

1.3.2 执行一个程序

为了执行一个程序，计算机控制器从第一条指令开始，检查在存储器中的每一条程序指令，发出适合于执行这个指令的命令信号。通常，指令按顺序执行。然而，后面我们将会看到，计算机也可以跳过某些指令或多次执行某些指令。

在执行指令期间，数据可以进入计算机存储器，也可以显示对这个数据执行的处理结果。自然，仅当程序包括计算机输入或显示的相应指令时，这些事才会发生。

图1.4显示了一个计算工资的程序和它的输入、输出之间的关系，也指出了在程序执行期间通过计算机的信息流。由程序处理的数据（职工工作小时卡片）必须首先输入计算机存储器（图1.4的第1步）。按程序指令的指示，中央处理器处理存储器中的数据，把计算结果回送到存储器（第2—4步）。当计算处理完成后，最后结果从计算机存储器中输出（第5步）。

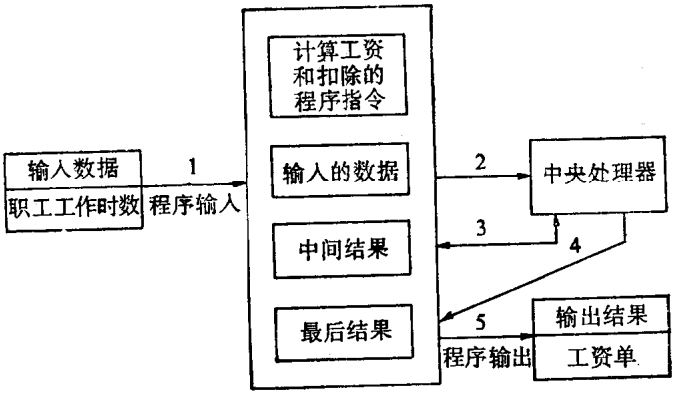

图 1.4 通过计算机的信息流

1.4 PASCAL 介 绍

1.4.1 在PASCAL中符号名的使用

PASCAL的一个最重要的特性是，它允许通过使用描述性的符号名（称为变量名或简称为变量），而不是存储地址，来引用存储在存储器中的数据。编译程序为在程序中使用的每一个变量分配一个存储单元。我们不必关心它的地址，而只需简单地告诉编译程序，我们所希望使用的每个变量名，让编译程序决定与该变量相联系的地址。

只有合法的PASCAL标识符才可以用作变量名。标识符必须以字母（A—Z）开始，后面可以跟任意个字母或数字（0—9）。

可以直接用标识符描述每个变量代表的内容。对于工资问题，可以使用变量HOURS（工作小时）、RATE（小时工资率）、GROSS（总工资）和NET（净工资）。

在PASCAL中用到的每个变量名都必须在变量说明中加以说明，以便由编译程序分配存储单元。例如

```
VAR
 HOURS, RATE, GROSS, NET:REAL;
```

其中REAL表明以上变量是实数类型。包括一个小数点的数称为实数（例如3.14，0.005，-35.0，0.0）。不包括小数点的数称为整数（例如-99，35，0，1）。整数类型以INTEGER标识。

变量说明的一般形式是：

```
VAR
 〈变量表〉:〈类型〉;
```

解释：变量表为一些用逗号分开的变量标识符。编译程序将为变量表中的每个变量标识符分配一个存储单元。存储在每个变量中的数据类型由冒号后的类型指明（例如REAL，INTEGER等）。

标识符的长度没有特殊的限制（有的PASCAL系统对标识符长度有限制）。但是，某些保留字对编译程序具有特殊的意义，不能用作标识符。字VAR是保留字。以后我们还会看到其它保留字，如CONST，PROGRAM，BEGIN，END等（附录A提供了PASCAL的全部保留字）。

在PASCAL中还有一些标准标识符，这些标识符是在PASCAL中预先定义的，具有专门的意义。例如REAL，INTEGER等。但是，它们不是保留字，可由程序员重新定义（尽管我们不主张这样做）。标准标识符也在附录A中给出。

1.4.2 计算机的几种操作

每一种计算机都具有它可以执行的特殊的操作集合。这些操作通常可分成三类：（1）输入输出操作；（2）数据处理和比较操作；（3）控制操作。

每一类又包括许多操作，其中某些操作对于多数计算机是共同的。例如：

（1）输入输出操作：读，写。

（2）数据处理和比较操作：加，减，乘，除，反号，拷贝，比较。

（3）控制操作：转移控制，条件执行。

下面将通过工资处理的例子说明在PASCAL中如何使用这些操作。

例1.1 给出职员的小时工资率、工作小时和税金扣除量，计算一个公司职员的总工资和净工资。

为了解决这个例子，需要在以下几节中介绍几个PASCAL语句。

1.4.3 简单的数据处理——赋值语句

我们选择HOURS和RATE变量分别表示工作小时和小时工资率，用变量GROSS和

NET 分别表示计算的总工资和净工资，TAX 表示从总工资中扣除的税金。为简单起见，现规定扣除税金总数是25元，而不管职员的工资是多少（更现实的税金方案应根据职员总收入的多少，采用不同的比率来扣除）。

这个例子要求执行以下两个计算：

（1）由工作小时与小时工资率的乘积计算总工资。

（2）从总工资减去税金，求出净工资。

通过使用 PASCAL 的赋值语句，可以让计算机执行这些计算：

```
GROSS:=HOURS*RATE;
NET:=GROSS-TAX
```

它们之所以叫作赋值语句，是因为它们将某个值赋给了变量。上列第一个语句将HOURS与RATE 的乘积值赋给变量 GROSS。赋值语句的右端称为表达式。这两个赋值语句的效果可用图 1.5 说明。

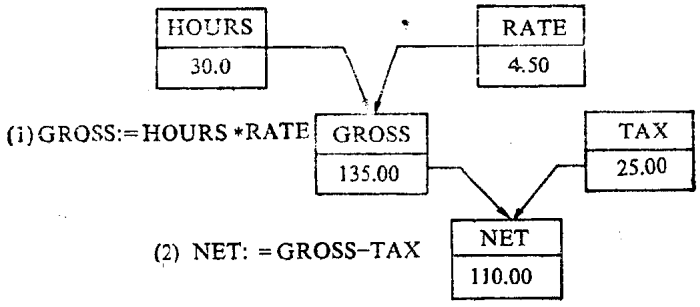

图 1.5　赋值语句的效果

第一个赋值语句的结果是，变量GROSS 的值由变量 HOURS 和RATE 值的乘积(135.00) 代替。第二个语句的结果是，变量NET 的值由变量GROSS 和 TAX 值的差(110.00) 代替。自然，我们假定在变量 HOURS、RATE 和 TAX 中已经存在了有意义的数据项。由这个算术运算序列改变的仅仅是 GROSS 和 NET 的内容。

在PASCAL中，写如下形式的赋值语句是完全允许的：

```
SUM:=SUM+ITEM
```

其中变量 SUM 出现在赋值操作（:=）的两端。这显然不是一个数学等式。该语句指示计算机，将变量 SUM 与 ITEM 的值相加，并将结果作为变量 SUM 的新值赋给它。在这个处理中，SUM 的原来值将消失，如图 1.6 所示。

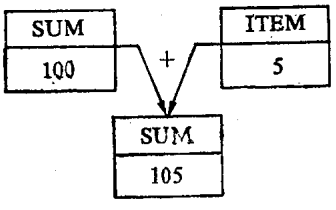

图 1.6　赋值语句的效果

赋值语句也可以包括单个运算对象。语句

```
NEWX:=X
```

指示计算机，将变量X的值存到 NEWX 中。而语句

```
NEWX:=-X
```

指示计算机，将变量X的值变号，其结果存到NEWX中。这些语句均不影响变量X的值。将一个数变号等于将它乘以－1。因此，如果变量X包括值－3.5，则语句 NEWX:=－X 将使 3.5 存到变量 NEWX 中。以后我们还将看到使用多个运算符和多个运算对象的更复杂的赋值语句。现将已讨论过的赋值语句归纳如下：

简单赋值语句

〈变量〉:=〈运算对象1〉〈算术运算符〉〈运算对象2〉

〈变量〉:=〈运算对象〉

〈变量〉:=-〈运算对象〉

解释：运算对象1和运算对象2代表正被处理的量。算术运算符指示正在执行的处理。运算对象可以是变量名或数。算术运算符是表1.1中给出的任一符号。运算的结果将作为新值赋给左端的变量，左端变量的原来值消失。运算对象的值是不变的。

表 1.1 PASCAL算术运算符

算术运算符	意　义
+	加
-	减
*	乘
/	除

1.4.4 在存储器中存储数据

信息必须首先存放到存储器中，计算机才能进行处理。有两种存放初始数据的方法：(1)使用常量定义语句；(2)在执行程序时读数据进入存储器。通常，对于程序常量(从程序的第一次使用到以后的使用不改变其值)的数据项采用第一种方法。对于可能改变其值的数据项则使用第二种方法。在工资问题中，所有职工应扣除的税金都是25.00，因此这个值可以通过使用常量定义语句与一个常量标识符相联系：

```
CONST
TAX=25.00;
```

常量定义语句的一般形式是：

```
CONST
  〈常量标识符〉=〈常量〉;
```

解释：将指定的常量（值）赋给常量标识符。常量标识符的值不得由后面的程序语句改变。

常量标识的类型与定义它的常量（值）的类型相同。因此，如果值是包括小数点的数，常量就是实型；如果值是没有小数点的数，常量就是整型。

下一节将描述，在一个程序的每次执行中，变化的数据如何输进计算机存储器。

1.4.5 READ 语句

由于一个公司的每个职员可能有不同的周工作小时数和小时工资率，HOURS 和 RATE 的值常常是改变的。因此，它们的值应当在程序执行时输入存储器，而且应当在执行所要求的计算以前完成。

在 PASCAL 中，用下列读语句指示计算机，在程序执行时读数据到变量 HOURS 和 RATE 中：

```
READ (HOURS, RATE)
```

执行这个数据输入过程后，变量 HOURS 和 RATE 的原来值就消失了。

程序和数据需要单独存放，对于不同的计算机系统，在程序执行时，数据项或者由终端输入，或者由程序从预先存有数据的次级存储器读入。

READ 语句如图 1.7 所示。

READ 语句的一般形式：

READ (〈输入表〉)

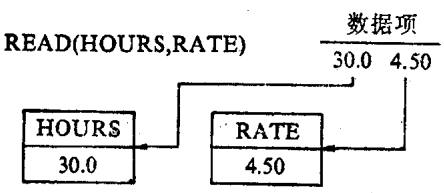

图 1.7 READ语句示意图

解释：该语句读入在输入表中给出的每个变量的值。输入表是一些由逗号分开的变量名。数据项被单独提供，项间具有空格。

为了证明已输入的值是否正确，可显示或回打存储的每个变量的值。打印也提供由程序处理的数据记录，这个记录对于程序员以及需要分析程序输出的人是相当有用的。

1.4.6 WRITE 语句

前面我们已经讨论了输入职员工作小时数、工资率以及计算总工资、净工资所需要的PASCAL 语句。计算结果已分别存于变量 GROSS 和 NET 中。但是，计算机所做的所有这些工作，对于我们仍无多大用处。因为我们不能实际查看存储在存储单元中的值是多少。因此必须有一个方法，以指示计算机显示或打印输出一个变量的值。PASCAL 的输出语句是：

```
WRITE (GROSS, NET)
```

它将变量 GROSS 和 NET 的值打印在一个输出行上。这个操作不改变 GROSS 和 NET 的值。WRITE 语句如图 1.8所示。

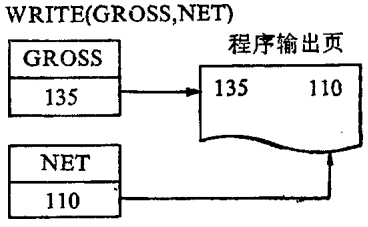

图 1.8 WRITE语句示意图

WRITE 的一般形式：

WRITE (〈输出表〉)

解释：在输出表中每个变量的值被顺序打印。逗号用于分开输出表中的项。

READ 和 WRITE 操作是 PASCAL 中的预定义过程。出现在输入、输出表中的变量称为过程参数。第五章将讨论过程与参数。

1.4.7 工资程序

我们现在可以汇集已经讨论过的所有语句，以产生例 1.1 的一个完全的 PASCAL 程序(程序1.1)：

程序1.1 PASCAL工资程序

```
PROGRAM PAYROLL (INPUT, OUTPUT);
  CONST
    TAX=25.00;
  VAR
    HOURS, RATE, GROSS, NET:REAL;
  BEGIN
    READ (HOURS, RATE);
    WRITE (HOURS, RATE);
    GROSS:=HOURS*RATE;
```

```
  NET:=GROSS-TAX;
  WRITE (GROSS, NET)
END.
```

程序 1.1 的第一个语句用于指示程序的名字PAYROLL，每一个 PASCAL 程序必须以下列程序语句开始：

```
PROGRAM (程序名) (INPUT, OUTPUT);
```

解释：程序名由合法的 PASCAL 标识符表示。INPUT 和 OUTPUT 指出存储输入数据的地方（系统文件 INPUT）和输出结果的地方（系统文件 OUTPUT）。

接下去的两个语句是 PASCAL 程序的说明部分。说明部分包括常量定义语句（如果需要）和变量说明语句。在 CONST 和 VAR 下面可以包括多个常量定义语句和变量说明语句，每个语句之间以分号分开。CONST 和 VAR 一般写在一个单独的行上。

保留字 BEGIN 指出了程序体（执行实际数据处理的一些语句）的开始。在这个程序体中前两个语句是输入变量 HOURS 和 RATE 的值并回打出它们。下面两个语句计算 GROSS 和 NET 的值。最后一个语句输出计算结果（GROSS 和 NET）。以保留字 END 跟一个句号作为程序体的结束。

在 PASCAL 中，分号用于分开每个语句。跟在每个语句后的分号用于将该语句与下一个语句分开。但在 BEGIN 后或 END 前不用分号。

如果程序 1.1 的数据部分包括两个数 30.0 和 4.50，程序可以产生下列输出行：

```
3.00000000E+01  4.50000000E+00  1.35000000E+02  1.10000000E+02
```

前两个数代表输入数据，后两个数是为 GROSS 和 NET 计算的值。

这些数据被打印成由字母 E 指示的科学表示形式（或称指数形式）。跟在 E 后面的两位十进制数指出小数点应移的位数。E 后面的符号指示移位的方向（+ 表示左移，- 表示右移）。上面打印的数等价于实数 30.0，4.5，135.0 和 110.0。

注意：在程序设计中使用空格是一个很重要的考虑因素。“看起来好”的程序比“差”的程序更容易阅读和理解。大多数程序将在某些时候由某些人检查或研究。如果一个程序书写整齐、意义明确，对每个人来说都是一个优点。

仔细地考虑和使用空格，可以显著地提高程序的可读性。在程序的字之间要求一个空格（例如程序语句中的 PROGRAM 和 PAYROLL 之间），然而在 PASCAL 中的额外空格可以被编译程序忽略。可按需要插入空格，以改善程序的风格和外貌。如程序 1.1 所示，我们总是在逗号以后和运算符（例如 +，-，：=）前后留一个空格，把保留字 CONST、VAR、BEGIN 和 END 写在单独的行上，以使它们突出。最后，我们将使用空格行把说明部分与程序的其余部分分开。

所有这些手段都是为了改善风格，提高程序的明晰性。

1.4.8 程序的一般形式

PASCAL 程序的一般形式显示在程序 1.2 中。在程序中所用的每个标识符必须在程序说明部分精确地说明一次，除非它是保留字或标准标识符。这意味着同一个标识符不能既用作常量名，又用作变量名。所有常量定义在 CONST 之后，所有变量说明在 VAR 之后。可以有多个常量定义或变量说明语句。在说明部分或在程序体中的每个语句都必须用一个分号与