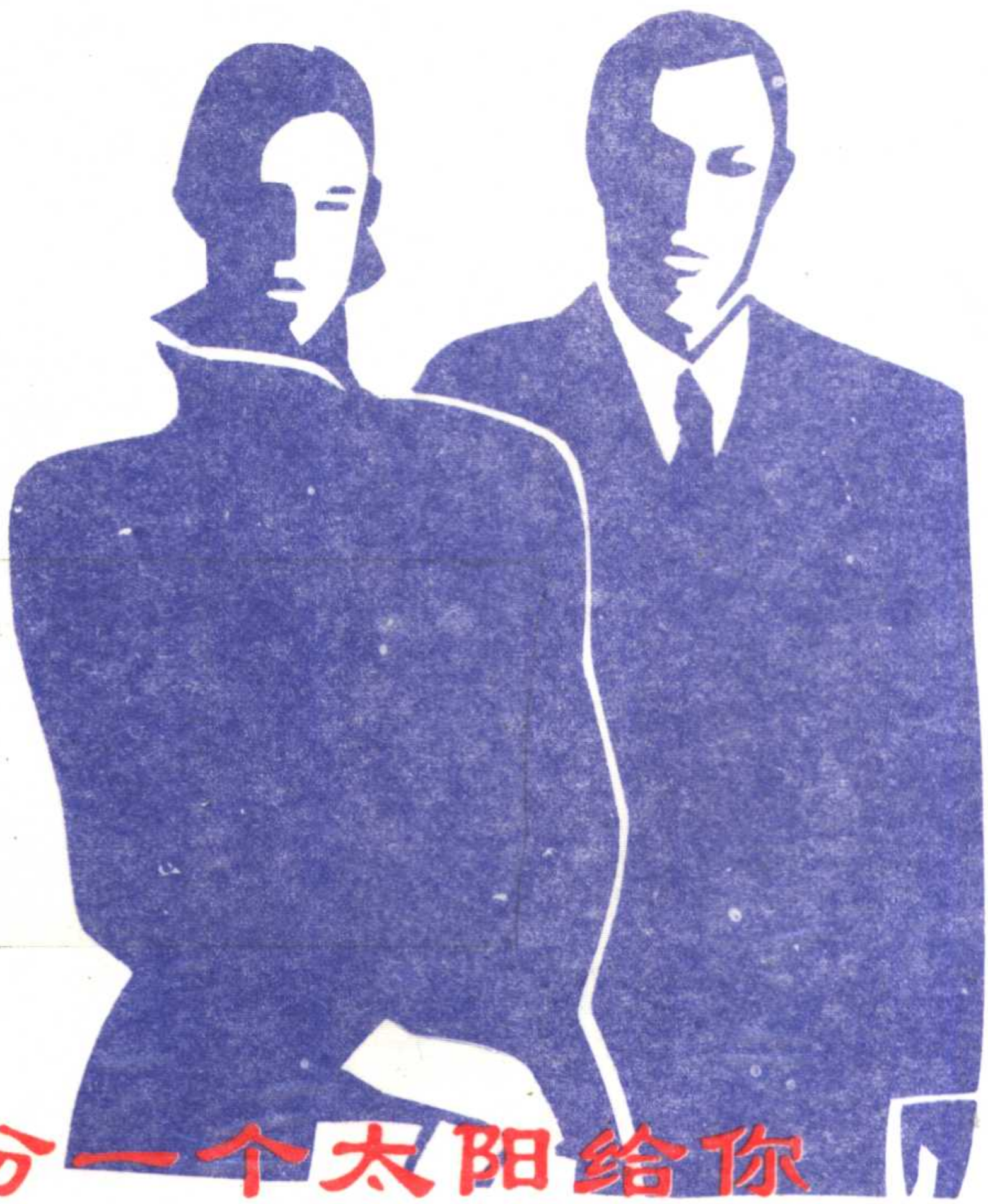
函授教育专用

PASCAL 语言



分一个太阳给你

PASCAL
语　言

覃红艳

教材编写组选编

目　录

第一章　PASCAL 语言介绍

1.1　PASCAL 语言的特点

PASCAL 语言是由瑞士的N. Wirth 教授于 1971 年提出来的。它的命名是为了纪念法国数学家 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 E. W. Dijdstra 和 C. A. Hoare 定义的结构化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，它已广泛地用于程序设计语言的教学和许多应用软件，系统软件的开发之中。

1.2　PASCAL 语言介绍

1.2.1 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 3.13,0.005,—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(例明 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 CONSTR，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，INTEGRER 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）。标准识符也在附录 A 中给出。

1.2.2 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，需要在以上几节中介绍几个 PASCAL 语句。

1.2.3 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 1.1 说明。

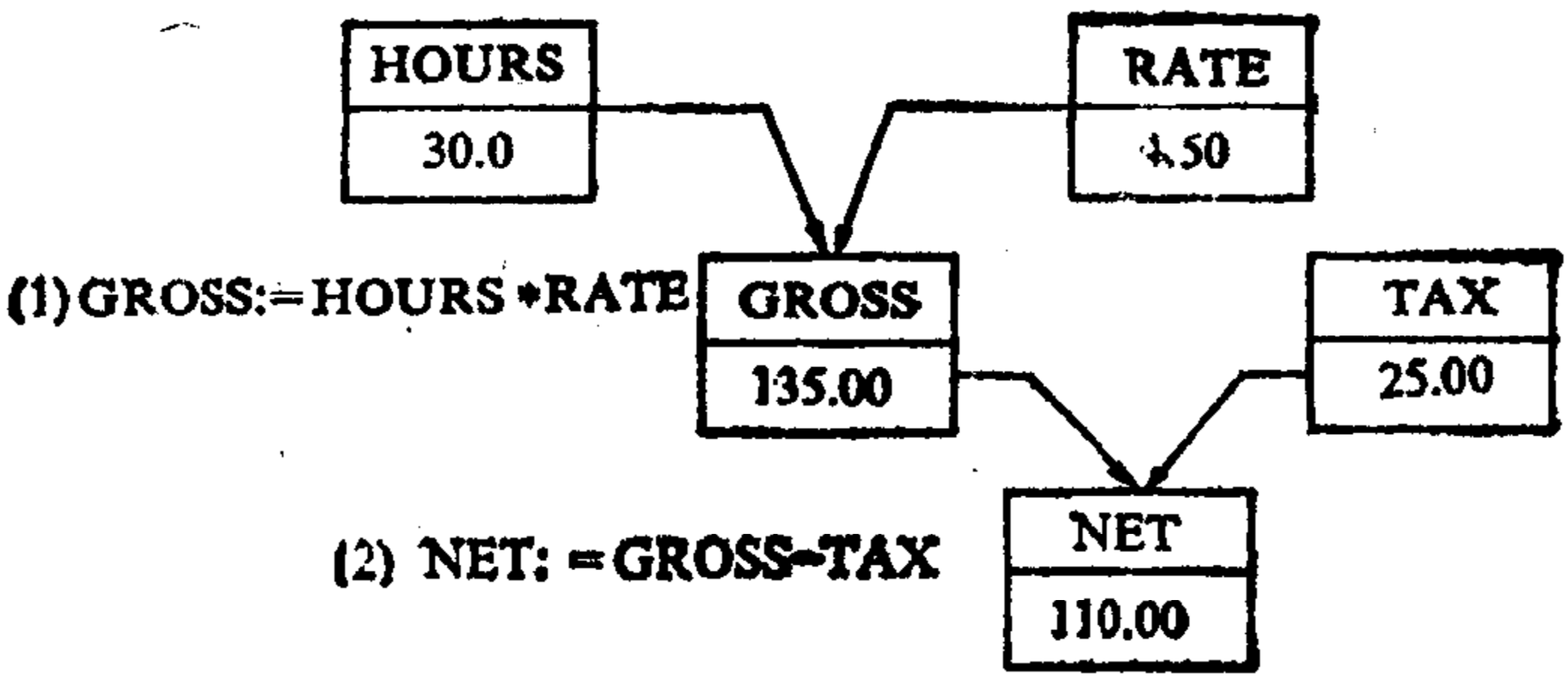


图 1.1　赋值语句的效果

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，写如下形式的赋句语句完全允许的：

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。该语句指标计算机，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 1.2 所示。

赋值语句也可以包括单个运算对象。语句

EEWX:=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。

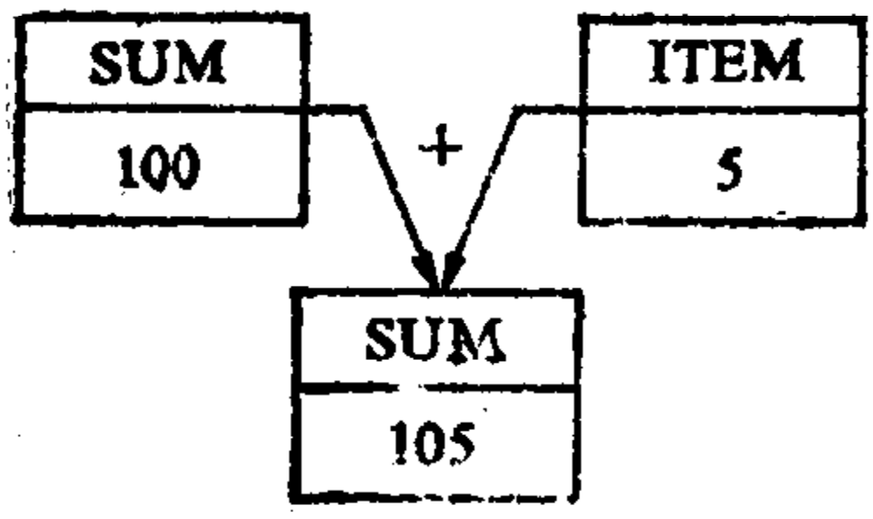


图 1.2　赋值语句的效果

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 NEWY 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。对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

1.2.4 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。常量标识的值得由后面的程序语句改变。

常量标识符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。

1.2.5 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，而用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：

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，数据项或者由终端力量输入，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 1.3 所示。

READ 语句的一般形式：

READ(〈输入表〉)

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。

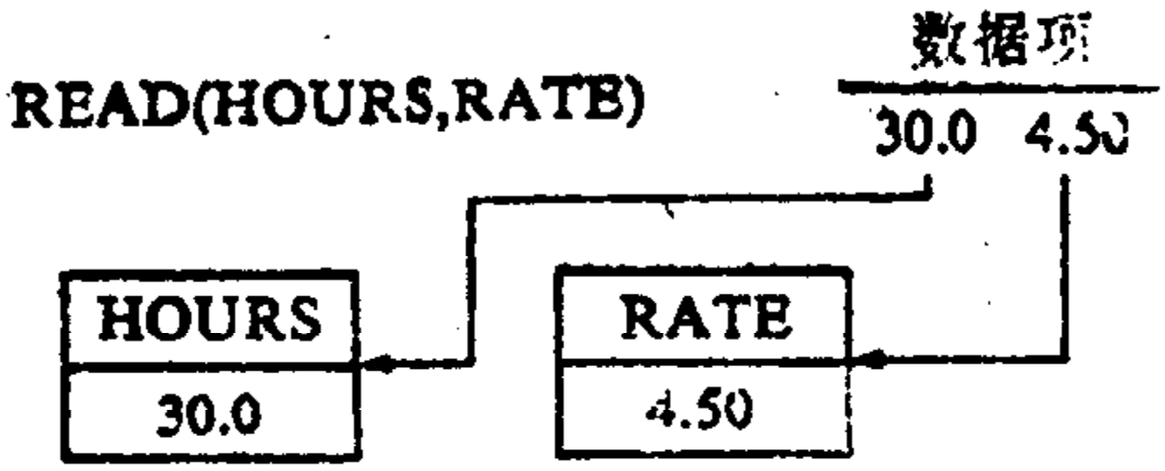


图 1.3 READ 语句示意图

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。

1.2.6 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存储单元的值是多少。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 1.4 所示。

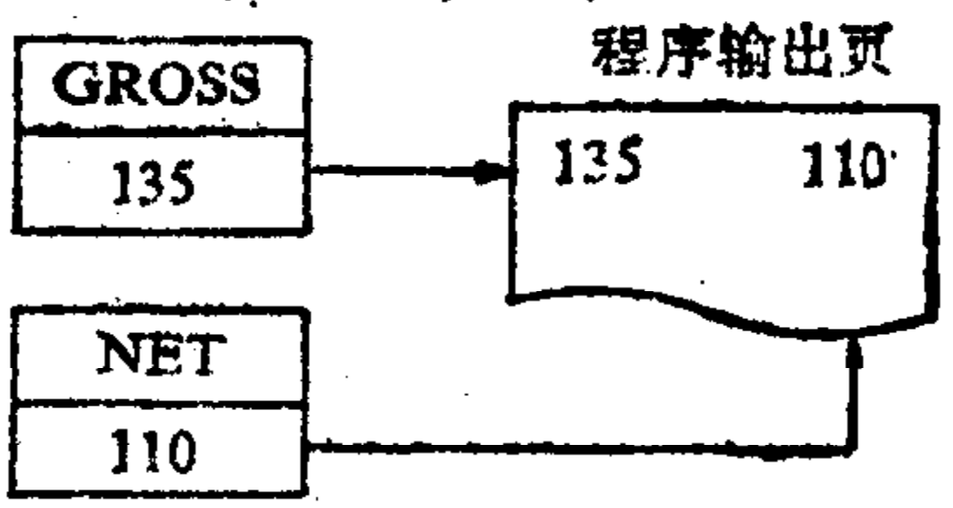


图 1.4 WRITE 语句示意图

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。

1.2.7 工资程序

我们现以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 (GROSE,NET)
  END
```

程序1.1的第一个语句用于指示程序的名字PAY-ROLL，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:

```
3.0000000 + 01     4.50000000 + 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、VSR、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。

1.2.8 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其后继语句分开。

在一个表达式中处理变量或打印变量以前，每个变量必须在程序体中通过读或赋值语句初始化。

程序 1.2　程序的一般形式

```
PROGRAM 程序名 (INPUT,OUTPUT);
  CONST                          ┐
    常量定义;                    │
  ⋮                              │
    常量定义;                    ├ 说明部分
  VAR                            │
    变量说明;                    │
  ⋮                              │
    变量说明;                    ┘
```

BEGIN
    程序语句；
    ⋮
程序语句
END，

程序体

程序1.2给出的只是PASCAL的一个较简单的形式。一个完全的PASCAL程序将包括更多的内容。这些内容将在以后各章中分别介绍。在程序1.3中给出了完全的PASCAL程序的一般形式。当然，对于一个具体的程序，不一定包括全部说明内容。但是，如果它们出现，必须按这里所指定的前后次序编写。而程序体部分是不可少的。注意，程序体以END。结束，最后一个句号不要漏掉。

程序1.3　完全的PASCAL程序

PROGRAM 程序名(程序参数表)；
LABEL
    标号说明；
CONST
    常量说明；
TYPE
    类型说明；
VAR
    变量说明；
FUNCTION
    函数说明；
PROCEDURE
    过程说明；

说明部分