

# 第二章 FORTRAN 语言基础

## 2.1 字符集与保留字

引子 让我们先从一个简单的示例程序展示 Fortran 语言的基本结构。

简单的 Fortran 示例小程序	
<code>program main</code>	开始语句，关键字为 <code>program</code> ，必要
<code>! example</code>	注释行，可写程序说明
<code>implicit none</code>	表示不需要程序自定义变量
<code>REAL st1,st2,st3,stave</code>	自行变量申明，定义实型变量，必要
<code>st1=8.5</code>	变量赋值
<code>st2=9.0</code>	
<code>st3=8.7</code>	
<code>stave=(st1+st2+st3)/3.0</code>	计算语句，运算式
<code>print *,stave=stave</code>	或者： <code>write(*,*)</code> 变量 或 ‘字符串’ 输出到屏幕
<code>end</code>	结束语句，必要

### 2.1.1 字符集

- 合法字符 Fortran 允许使用的字符包括（解释语句/注释中可随意使用中文等字符）：
- ① 英文字母：A-Z 与 a-z（大小写在 Fortran 中不作区分，如 REAL, real, Real 是完全一致的）
  - ② 阿拉伯数字：0-9
  - ③ 特殊符号：空格 = + - \* ( ) , . ' : " ' ! & ; < > \$ ? \_

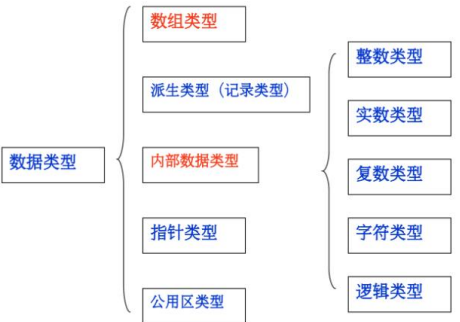
### 2.1.2 保留字

- 保留字 又称为关键字，是 Fortran 中具有特定意义的字符串，例如语句关键字、内部函数名。
- 语句关键字 有 IF, THEN（选择语句）, PROGRAM（程序开始）, INTEGER（整型）, REAL, READ, PRINT, WRITE, DO（执行循环）, END, SUBROUTINE（子程序）, FUNCTION 等。
- 内部函数名 系统内部固有的函数库及其函数名，例如 ABS, SIN, LOG 等。
- 注意 Fortran 允许保留字作为其他实体的名称（变量名、数组名、函数名、程序名等），但很不推荐。例如 `program program` 中的第一个为保留字，第二个为主程序单元名称。

## 2.2 基本数据类型

### 2.2.1 提供的数据类型

- 主要类型 数组类型、派生类型（记录类型）、内部数据类型、指针类型、公用区类型。
- 内部数据类型 整型、实型、复数类型、字符类型、逻辑类型。
- 注意 不同类型数据具有不同特性，其处理方式有所不同，取值范围也不同，在处理数据前，必须先声明数据特性。这与 Python 的某些数据自动转换不同。



目前常用的数据库（如 SQL）均为关系数据库，由表组成，第一行为字段名（表征事物基本特征的属性），其余行每行称为一条记录。一个关系数据库由很多条记录组成，读写可以以记录为单位，这就是记录类型。

## 2.2.2 数据类型的性质

- 四条性质**
- ① 每个数据类型具有唯一确定的名称。
  - ② 每个数据类型规定了一个取值范围（值的集合）。
  - ③ 每个数据类型规定了其常量数据的表示方法。
  - ④ 每个数据类型规定了一组操作。

## 2.3 常量与变量

### 2.3.1 常量

**常量的概念** 在程序运行过程中，其值不能被改变的量称为常量。它在程序中直接生成并直接用于计算和处理。

**常量的类型** 包含整型、实型、复数、字符型、逻辑型。

#### 2.3.1.1 数值型常量

**整型常量** 又称为整型常数或**整数**，包括正数、复数和零值，例如+5、-36、0等。

**实型常量** 又称为实型常数或**实数**，它具有两种形式：

- ① 小数形式（100.56）
- ② 指数形式（5.35E5 表示  $5.35 \times 10^5$ ）

当单精度实数不足以表示一个数的大小或精度时，可以使用**双精度实数**。只需要将指数部分的 E 改变为 D 即可，例如 6.85746304857D5 具有双精度。

**复型常量** 又称为复型常数或**复数**，例如(1.0,1.0)表示  $1.0 + 1.0i$ ，(2.1,-4.5)表示  $2.1 - 4.5i$ 。

#### 2.3.1.2 字符型常量

**字符型常量** 又称为**字符串**，使用一对单引号或双引号括起来的数个**非空**字符串。例如'a'、'A'、'x+y'等。

- 注意**
- ① 与变量名不同，字符串内部字母区分大小写。
  - ② 字符串中间若带有撇号，需要使用转义符或双引号字符串，例如 'I'm a boy.' 或 "I'm a boy".
  - ③ 字符串长度包含空格，"" 为空字符串，'' 长度为 1。

#### 2.3.1.3 逻辑型常量

**逻辑型常量** 在 Fortran 中，逻辑常量有且仅有两个：**.TRUE.** 和 **.FALSE.**，两侧有两个小点。

**注意** 对于逻辑值.TRUE.，其在存储单元字节内每位为 1，可视为整数-1；对于.FALSE.，则为 0，它们均能够直接参与到整数运算。例如： $7 + \text{FALSE} = 7$ ,  $1 + \text{TRUE} = 0$ 。

#### 2.3.1.4 符号常量

**符号常量** 例如圆周率 $\pi$ ，重力加速度 g 等数据，关键字为 **PARAMETER:**。

##### 定义方式

**REAL** pi, x, y, z

此时为多个变量

**PARAMETER**(pi=3.1415926, x=1, y=2, z=3)

此时变为多个常量，其值不可再次更改

### 2.3.2 变量

**变量的概念** 变量在程序运行期间值可以变化，系统为程序中的每一个变量开辟一个内存空间，用来存放值。因此，使用变量前必须进行定义，否则数据将无处可放。

**变量的命名** Fortran 中规定，变量必须以**字母开头**，随后可接多达 **30 个字母、数字或下划线**。

例如，Sum\average\student\_name 是合法变量名，\_total\M.D.John 都是非法变量名。

**变量的类型** 总体分为三类五种，即数值型变量（包含整型、实型、复型）、字符型变量与逻辑型变量。

### 2.3.2.1 整型变量

**变量定义** 下面为合法声明整型变量的语句，关键字为 **INTEGER**：

代码定义	
<b>INTEGER</b> (KIND=2) a,b,c,d	批量声明长度为 2 的 4 个整型变量
<b>INTEGER</b> (1) e	声明长度为 1 的 1 个整型变量
<b>INTEGER</b> f	声明长度为 4（缺省，由计算机位数决定）的 1 个整型变量
<b>INTEGER</b> ::g=123	声明长度为 4 的 1 个整型，且初始值为 123

**注意** 符号::在声明中可有可无，若有则可赋初值，否则不可赋初值。如 **INTEGER f=123** 是非法语句。

### 2.3.2.2 实型变量

**变量定义** 下面为合法声明实型变量的语句，关键字为 **REAL**：

代码定义	
<b>REAL</b> (KIND=4) a,b,c,d	声明长度为 4 的 4 个实型变量
<b>REAL</b> (8) e	声明长度为 8 的 1 个实型变量
<b>REAL</b> f	声明长度为 4（缺省）的 1 个实型变量
<b>REAL</b> ::g=1.23	声明长度为 4 的 1 个实型，且初始值为 1.23

**注意** KIND 值为 8 的实型变量为双精度变量，可由 **DOUBLE PRECISION** 声明取代。

### 2.3.2.3 复型变量

**变量定义** 下面为合法声明复型变量的语句，关键字为 **COMPLEX**：

代码定义	
<b>COMPLEX</b> (KIND=4) a,b,c,d	声明长度为 4 的 4 个复型变量
<b>COMPLEX</b> (8) e	声明长度为 8 的 1 个复型变量
<b>COMPLEX</b> f	声明长度为 4（缺省）的 1 个复型变量
<b>COMPLEX</b> ::g=(3,4)	声明长度为 4 的 1 个复型，且初始值为 $3 + 4i$

### 2.3.2.4 字符型变量

**变量定义** 下面为合法声明字符串的语句，关键字为 **CHARACTER**：

代码定义	
<b>CHARACTER</b> a	声明长度为 1（缺省默认值）的 1 个字符型变量
<b>CHARACTER</b> (8) b,c	声明长度为 8 的 2 个字符型变量
<b>CHARACTER</b> (len=4) e,f,g	声明长度为 4 的 3 个字符型变量
<b>CHARACTER</b> *6 h	声明长度为 6 的 1 个字符型变量
<b>CHARACTER</b> ::a='a'	声明初值为 'a' 的字符型变量
<b>CHARACTER</b> (7) ::b='Fortran',c	字符串 b 的初值为 Fortran，c 初值为 7 个空格

**注意** 形如声明语句 **CHARACTER \*7 h='student'** 为非法语句（没有双冒号）。

### 2.3.2.5 逻辑型变量

**变量定义** 下面为合法声明逻辑型变量的语句，关键字为 **LOGICAL**：

代码定义	
<b>LOGICAL</b> (KIND=4) a	声明长度为 4 的 1 个逻辑型变量
<b>LOGICAL</b> (4) a	声明长度为 4 的 1 个逻辑型变量
<b>LOGICAL</b> a	声明长度为 4（缺省）的 1 个逻辑型变量
<b>LOGICAL</b> ::a=.True.	声明长度为 4 的 1 个逻辑型，且初始值为真

## 2.3.3 变量的声明

- 声明方式** 在 Fortran 中，变量类型需要通过类型声明语句来定义，且有两类形式：**显式声明**和**隐式声明**。显式声明即为上文各类型变量代码定义中的声明规则，我们来重点介绍隐式声明（隐含约定）。
- I-N 规则** 在程序中，凡是变量名用 **I, J, K, L, M, N, i, j, k, l, m, n** 开头的变量，均被默认为整型变量，以其他字母开头的变量均被默认为实型变量。例如 **id** 为整型，**total** 为实型。
- 这种隐式声明的方式在 Fortran 90/95 中不被提倡使用，建议在变量声明前使用 **IMPLICIT** 取消该规则。
- IMPLICIT** 这种语句可以禁止 I-N 规则或重新定义 I-N 规则，它的具体使用方式如下：

IMPLICIT 使用方法	
<b>IMPLICIT NONE</b>	关闭默认类型功能，任何变量都需要事先声明
<b>IMPLICIT INTEGER(a,b,c)</b>	a,b,c 开头的变量默认为整型
<b>IMPLICIT REAL(m-p)</b>	从 m 到 p 开头的变量都认为是实型

- 注意**
- ① 在所有变量声明方法中，类型显式声明语句优先级最高，IMPLICIT 语句次之，I-N 规则最低。
  - ② 类型说明语句和 IMPLICIT 语句都是非执行语句。
  - ③ 类型说明只在本程序单位内有效。
  - ④ **IMPLICIT 命令必须置于 PROGRAM 命令的下一行**，不能把它放在其他位置。
- 初始化** 直接把数值写在声明的变量后面，使用该方法时，**不能省略定义语句中间的冒号 ::**。
- 或者在声明后，单起一行，例如 **real a; a=1**
- 批量初始化** 使用 DATA 命令批量按顺序设置：**DATA a, b, c, string/1, 2.0, (1.0,2.0), "FORTRAN"/**

## 2.4 运算符和表达式

- 一般概述** 运算符包括**算术运算**、**字符运算**（**//连接运算**）、**关系运算**和**逻辑运算**。

### 2.4.1 算术运算符及其表达式

- 运算符** **+正号、-负号、\*乘号、/除号、\*\*乘方**，不同运算符有优先级顺序。例如：**(a-b)/c\*\*2+sin(x+y)**。

- 注意**
- ① 由于用 / 号作为除号，因此在写除法运算式时应加上必要的括号。
  - ② 乘号不能省略。如 **asinx**，必须写成 **a\*sin(x)**。
  - ③ FORTRAN 中无大、中、小括号之分，一律用小括号。
  - ④ 乘方按**先右后左**原则处理。
  - ⑤ 对单项运算符（±）相当于在它前面有一个运算量 0，如 **-a\*\*2** 相当于 **0-a\*\*2**，而不是 **(-a)\*\*2**。

- 求值运算**
- ① **同类型的操作数**之间运算的结果仍保持原类型。特别要注意：**两个整数相除的商也是整数**。例如，**5/2** 的值是 2 而不等于 2.5，**4\*(-1)** 等于 0，应写为 **5\*\*(1./3.)** 而不是 **5\*\*(1/3)**。
  - ② 如果参加运算的**两个操作数为不同类型**，则编译系统会自动将它们转换成同一类型后进行运算。转换的规律是：将低级类型转换成高级类型。类型的转换时**从左向右**进行的，在遇到不同类型的操作数时才进行转换。例如，**1/2\*1.0** 等于 0，而 **1./2\*1** 等于 0.5。

- 优先级** **COMPLEX > REAL > INTEGER**，同一类中长度长的高于长度短的。

### 2.4.2 关系运算符及其表达式

- 运算符** **.LT. < .LE. <= .EQ. == .NE. /= .GT. > .GE. >=**

- 格式** 表达式 1 **关系运算符** 表达式 2

- 注意**
- ① 如果两个表达式都为算术表达式，则进行关系运算前将其转换成同一类型。
  - ② 如果两个表达式都为字符表达式，则进行关系运算前将其转换成等长字符串，不足末尾补足空格。
  - ③ 复数的关系运算只有两种：等于和不等于。
  - ④ 对算术表达式进行关系运算，根据它们值的大小决定运算结果。
  - ⑤ 对字符表达式进行关系运算，**依次比较**两字符串相应位置字符的 ASCII 码值大小决定运算结果。

例如	
12>34	结果为 .FALSE.
(4+5*2).LE.10	结果为 .FALSE. (14<10 为假)
(4.2,7.3).NE.(7.3,4.2)	结果为 .TRUE.
MOD(4,2).EQ.0	4 除以 2 的余数是否等于 0。结果为.TURE.
'banana'<='apple'	结果为 .FALSE.
'is a pen.'<='is a pencil.'	字符 . 的 ASCII 为 46，而 c 的 ASCII 为 99，结果为.True.

### 2.4.3 逻辑运算符及其表达式

运算符	.NOT.	非	.AND.	与	.OR.	或
	.XOR.	异或	.EQV.	相同	.NEQV.	不相同
一般形式	逻辑值 1	逻辑运算符	逻辑值 2	(逻辑值通常是判断条件的表达式)		

运算含义	逻辑变量	逻辑非	逻辑与	逻辑或	逻辑异或	逻辑相等	逻辑不等
	a b	.NOT.a	a.AND.b	a.OR.b	a.XOR.b	a.EQV.b	a.NEQV.b
	T T	F	T	T	F	T	F
	T F	F	F	T	T	F	T
	F T	T	F	T	T	F	T
	F F	T	F	F	F	T	F

#### 示例-计算饱和水汽压（马格努斯经验公式）

公式形式： $E = E_0 \cdot 10^{\frac{7.45t}{237.3+t}}$  或  $E = E_0 \cdot e^{\frac{17.67t}{243.5+t}}$ ，其中  $E_0 = 6.11hPa$ 。  
 则其表达式为： $E = E0*10^{((7.45*t)/(237.3+t))}$  或  $E=E0*\exp(17.67*t/(243.5+t))$

## 2.5 语句

### 2.5.1 赋值语句

**语句类型** 三种赋值语句：算术赋值语句、逻辑赋值语句、字符赋值语句

**语法描述** 变量名 = 表达式

#### 2.5.1.1 算术赋值语句

**描述** 在一个赋值表达式中，如果变量名与表达式均为数值类型(整型、实型或复型)，则称为算术赋值语句。  
**注意** 如果右边表达式类型与左边变量类型不一致时，将表达式计算后的结果强制转换为左边变量类型，并将转换后的值赋予左边变量。例如，INTEGER k; k=4.5\*3.5 k 实际赋值为 15

#### 2.5.1.2 逻辑赋值语句

**描述** 赋值号左边变量和右边表达式类型均为逻辑型。

语句合法性判断	
LOGICAL flag1,flag2,flag3,flag4	声明逻辑值
flag1=.TRUE.	合法
flag2='China'	非法，是字符型。
flag3=1.5	非法，是实型。
flag4=flag1.AND.i>100	两侧都是逻辑值，计算结果为逻辑值，合法。

#### 2.5.1.3 字符赋值语句

**描述** 赋值号左边变量和右边表达式类型均为字符型。

语句合法性判断	
CHARACTER*7 str	下面语句是合法语句：（声明长度为 7 的字符型）



str='student'	标准赋值。
str='He is a '//student'	连接运算符，实际赋值为'He is a'（保留前 7 位）
CHARACTER*7 str	下面语句是非法语句：
str='student'+125	125 是整数不能与字符串进行加法运算
str=125+3*20	右边不能为算术运算表达式
str=a<100.AND.p	右边不能为逻辑运算表达式

- 注意** 当右边表达式长度与左边变量长度不同时：
- ① 当右边表达式长度小于左边变量长度，将表达式运算后的结果长度**强制转换为左边变量长度**，**不足补空格**（所以声明时字符型长度不能太长），并将转换后的字符串赋予左边变量；
  - ② 当右边表达式长度大于左边变量长度，将表达式运算后的结果左侧部分赋予变量，**多余截去**。

#### 案例

```
CHARACTER*5 str1
CHARACTER*3 str2
str1='is'
str2='china'
```

执行以上语句后，str1 值是“is□□□”，str2 值是“chi”。

#### 2.5.1.4 DATA 赋值语句

**一般形式** DATA 语句给数组赋初值的一般形式为：DATA 变量列表/初值表/,变量列表/初值表, ... 可分多部份其中，**初值表中只允许出现常量，不允许出现表达式**。例如：DATA a,b,i/3.0,-3.1,8/

**规定** 对 DATA 语句为变量赋初值，Fortran 作如下规定：

- ① 在初值表中如果有几个**连续相同的变量**可以简写为：**n\*常量**。例如：DATA a,b,c,i,k/3\*1.0,2\*3/
- ② 变量列表中的变量与初值表中的常量必须个数相同，类型一一对应。  
例如：DATA a,b,c,d/3.0,2\*2.0/i/3.0/ 这个赋值是错误的，第一个变量列表无法对应。
- ③ 在一个程序单位中有多个 DATA 语句给同一个变量赋初值，以**最后一个 DATA 中所赋的初值为准**。  
例如，在一个程序单元中有以下 DATA 语句：DATA a,b,c,d/1.0,3.0,2\*0.0/; DATA x,y,c/4.0,2.0,7.5/。  
其中变量 c 分别在两个 DATA 语句中出现，并且赋的值不同，结果 c 的值应为 7.5。

#### 2.5.2 程序控制语句

**PROGRAM** Fortran 允许编程人员为自己的程序定义一个名字，其语句格式是：PROGRAM 程序名  
PROGRAM 语句可以省略，如果不省略必须放在该程序块的第一个语句位置。

**END** END 语句在 Fortran 中是可执行语句，它有三个功能：

- ① 作为一个程序块的结束标志。
- ② 主程序中 END 语句表示整个程序的终止执行语句。
- ③ 子程序中执行 END 语句，作用与返回语句 RETURN 语句作用相同。

**每个程序单元必须有一个 END 语句在该程序单元的最后一行。**

**STOP** 作为一个程序块的结束标志。STOP 语句的一般形式为：STOP [n]

其中，n 为在执行 STOP 语句时所输出的信息—整数或字符串，一般为行号，便于调试。如果 STOP 出现在主程序中，则直接结束主程序。

**PAUSE** 暂停语句，用于暂停程序的运行，但不结束程序的运行，当需要从暂停处恢复运行时，按一个回车键即可。PAUSE 语句的一般形式为：PAUSE [n]

#### 2.5.3 其他内部函数(部分)

<b>ABS(x)</b> 求x的绝对值 $ x $	<b>EXP(x)</b> 求指数函数 $e^x$	<b>SIN(x)</b> 求正弦函数 $\sin x$ ，单位为弧度
<b>COS(x)</b> 求余弦函数 $\cos x$	<b>ASIN(x)</b> 求反正弦函数 $\arcsin x$	<b>ACOS(x)</b> 求反余弦函数 $\arccos x$
<b>TAN(x)</b> 求正切函数 $\tan x$	<b>ATAN(x)</b> 求反正切函数 $\arctan x$	<b>ATAN2(x,y)</b> 求反正切函数 $\arctan y/x$
<b>LOG(x)</b> 求自然对数 $\ln x$ 或 $\log_e x$	<b>LOG10(x)</b> 求常用对数 $\log_{10} x$	<b>INT(x)</b> 取x的整数部分，不四舍五入
<b>REAL(x)</b> 把整形量x转换为实型	<b>MAX(x1,x2,...)</b> 求最大值	<b>SQRT(x)</b> 求平方根
<b>MOD(x1,x2)</b> 求 $x_1$ 除以 $x_2$ 的余数，即求 $x_1 - \text{int}(x_1/x_2) * x_2$	<b>SIGN(x1,x2)</b> 若 $x_2 > 0$ ，则 $ x_1 $ ；若 $x_2 < 0$ ，则 $- x_1 $	

## 2.6 输入与输出

- 总体概述**
- ① **表控格式**输入输出。Fortran 输入输出中最简单的一种方式（**键盘输入，显示器输出**），是按系统隐含的标准格式输入输出，并不常用，最常用的是有格式输入输出。
  - ② **有格式**输入输出。按用户规定的数据格式输入输出，故也称**可控格式**（或有格式）的输入、输出。
  - ③ **无格式**的输入输出。以**二进制形式**输入输出数据，适用于计算机内存与磁盘之间的数据交换。本节只讲①②表控输入/出，③将在文件一章讲解。

### 2.6.1 表控输入语句

**描述** 表控输入不必指定输入数据的格式，所以又称为自由格式输入，其一般形式为：**READ \*,输入表** 其中，“\*”号表示表控输入；而输入表（如不同类型的变量，中间用逗号分隔开）则用来控制数据的输入，即要求输入表和输入的数据有严格的对应关系。

#### 案例

a,b 为实型, m,n 为整型 **READ \*,a,b,m,n** 从键盘输入以下数据:  
3.7, -1.8, 24, 10 ✓（数据间用逗号分隔）或： 3.7 -1.8 24 10 ✓（数据间用空格分隔）  
则 a=3.7, b=-1.8, m=24, n=10。

实际更为常用的形式是：**READ (\*,\*)**，第一个\*代表输入位置，第二个\*代表格式控制，当都是\*时，默认为系统输入设备，即键盘（若定义为 1，则为文件）。

- 注意**
- ① 应保证从输入设备上输入数据的个数与 READ 语句输入表中变量的个数相同，各数据类型与相应变量的类型一致，否则执行错误。
  - ② 输入数据可分为多行输入，直到输入全部数据。
  - ③ 输入数据个数要求不少于输入表中变量个数。如果少于变量个数，则程序将等待用户输入后续数据。**如果多于变量个数，则多余的数据不起作用。**

#### 案例

上述 READ 语句执行时输入以下数据： 3.7, -1.8, 24, 10, 75, 34, 2.4 ✓  
后 3 个数是无效的，将被忽略。READ 语句读取前 4 个数后，程序将继续执行下一条语句。

- ④ 使用多个 READ 语句时，每个 READ 语句都是从一个新的输入行开始读数的。

#### 案例

**READ \*, a,b; READ \*,m,n**

如果输入数据为：3.7, -1.8, 24, 10 ✓ 第一个 READ 语句读入前两个数，即 a=3.7, b=-1.8，而**第二个 READ 语句并不会从这一输入行剩余的数据中读数**，所以 m, n 未被赋值。想要正确赋值，应改为两个输入行：  
3.7, -1.8 ✓      24, 10 ✓

- ⑤ 输入数据时，**可以用符号斜杠 “/” 结束输入**，未被输入数据的变量保持原值不变。

#### 案例

**READ \*,a,b,m,n** 输入数据为：3.7, -1.8/ 24, 10 ✓  
执行结果为 a=3.7, b=-1.8, m, n 均未被赋值。

- ⑥ 如果 READ 语句中**有几个连续地变量要赋以相同的值**，则可用**重复因子 r**，r 表示某一数据重复出现的次数。

#### 案例

输入语句 **READ \*,i,j,k,a,b,c,d,str1,str2** 执行时输入以下数据：3\*12, 4\*125.45, 2\* 'student' ✓  
将 12 赋予 i, j, k, 将 125.45 赋予 a, b, c, d, 将 “student” 赋予 str1 和 str2。

⑦ 在一个数之间不能插入空格。因为空格也是两个数据间的分隔符。

#### 案例

m=123, n=456, 输入语句为: **READ \*, m,n** 如果输入数据为: 1□23, 456 ✓  
键入数据时不小心输入了一个空格, 则执行结果为 m=1, n=23, 显然不是想要的数。

⑧ 当变量为整型, 而输入的数据为实型时, 按出错处理。若变量为实型, 而输入数据为整型, 则系统自动将输入数据转换为实型再赋值给实型变量。

#### 案例

**real a,b; READ \*, a,b** 输入: 12, 34 执行结果为 a=12.0, b=34.0。

## 2.6.2 表控输出语句

### 2.6.2.1 PRINT 输出语句

**描述** PRINT 语句是只能以计算机系统隐含指定的打印机(或显示器)为设备进行打印输出。  
**形式** **PRINT \*** [,输出表]

### 2.6.2.2 WRITE 输出语句

**描述** WRITE 语句是可以指定以什么设备作为输出的对象(打印机、显示器、驱动器等)。  
**形式** **WRITE(\*,\* )** [,输出表]