第3章 顺序结构程序设计

教学目标：

- 掌握赋值语句的使用方法。
- 掌握常见的输入和输出语句的使用方法。
- 掌握 end 语句的使用方法。
- 了解 stop 语句、pause 语句的使用方法。

通过前两章的学习，我们了解了 FORTRAN 语言的一些基本知识、简单的 FORTRAN 程序和上机步骤，掌握了程序中用到的一些基本要素（如常量、变量、运算符、表达式等），它们是构成 FORTRAN 语言程序的基本组成部分。从本章开始将学习如何编写程序，以及编写程序所需掌握的一些知识。下面先从一个简单的例题开始学习编写程序。

【例 3-1】 已知 a＝1，b＝2，c＝a＋b，编写程序计算 c 的值。

程序编写如下：

```
integer a, b, c                          !定义整型变量 a,b,c
a=1                                      !给 a 赋值 1
b=2                                      !给 b 赋值 2
c=a+b                                    !计算 a、b 的和
print *, "C=",a,"+",b,"=",c              !输出 c 的值
end
```

程序运行结果如图 3.1 所示。

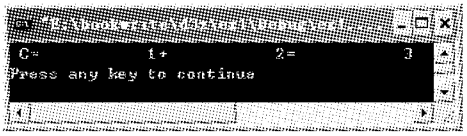

图 3.1 例 3-1 运行结果

这个实例程序很简单，程序的实际执行命令有四行（第 2～5 行）：

第 2 行到第 3 行是赋值语句，分别将原始数据赋值给变量 a、b，以待后续处理。

第 4 行也是赋值语句，是对原始数据的加工处理，即求和，求和后的结果保存到变量 c 中。

第 5 行是输出语句，将运行结果显示到屏幕上。

程序执行时按顺序依次执行每一个语句的命令，直到程序结束。将这种程序结构称

为顺序结构，它是最简单的程序设计结构。

在本章中，主要介绍实现顺序结构的基本语句：赋值语句以及输入输出语句的基本知识。

3.1 赋值语句

从上面的实例程序可以看到，赋值是一种非常重要的概念，作用是将一个确定的值赋给一个变量。

一般格式为：

```
v=e
```

v 代表一个变量名(v 是“变量”英文单词 variable 的第一个字母)，“=”称为赋值号。e 代表一个表达式(e 是“表达式”的英文单词 expression 的第一个字母)。也可以写成：

```
变量=表达式
```

例如：

```
x=1.2
y=x
z=(-b+sqrt(6**2-4*a*c))/(2*a)
```

都是正确的赋值语句。

赋值语句使用说明：

(1) 赋值语句的功能是先计算右边表达式 e 的值，而后将此结果赋给左边的变量。对变量的赋值过程是“覆盖”过程，指的是在变量对应的存储单元中用新的值去替换原有的值。例如：

```
n=n+1
```

在数学中是错误的，但在 FORTRAN 语言中却是一句正确的赋值语句。这句赋值语句作用是取出变量 n 对应存储单元中的数值，加上 1，再将新的值存入变量 n 对应的存储单元中，覆盖原来的值。如果 n 原来为 2，执行上面的赋值语句后，n 的值是 3，再执行一次，N 的值是 4，依次类推。

(2) 赋值号“=”是语句符号，执行赋值操作，不是运算符，不能去判断赋值号两端相等。

(3) 赋值语句不能连等，即赋值语句只允许出现一个赋值号，不允许有两个赋值号。例如 a=b=3 在数学上是合法的，但是非法的 FORTRAN 赋值语句。

(4) “=”两边数据类型不相同时，先对右边表达式进行计算，然后将计算结果的数据类型转换为赋值号左边变量的数据类型进行赋值。

例如：

```
integer::n=3
```

```
n=n*1.5+n/4
```

执行后，n 的值为 4。

赋值过程为：先计算表达式 n*1.5+n/4，表达式计算分 3 步：第 1 步计算 n*1.5，结果为 4.5；第 2 步计算 n/4，结果为 0，第 3 步计算 4.5+0，结果为 4.5。然后将表达式结果 4.5 转换为整型值 4，再赋值给整型变量 n。

当赋值号两侧的类型不同时，往往会产生程序设计者事先预想不到的结果。所以在编写程序时，应尽可能使赋值号两侧保持同类型。

(5) 如果是字符变量赋值语句，赋值时应遵循以下规律：

右边字符表达式长度与左边变量长度相同时，直接赋值。

右边字符表达式长度小于左边变量长度时，在表达式字符串后面补空格使其和变量等长，然后赋值，即“左对齐，右补空格”。

右边字符表达式长度大于左边变量长度时，将表达式字符串从左侧开始截取与变量长度相同的字符串，然后赋值，剩余舍去，即“左对齐，右截掉”。

【例 3-2】 字符型数据赋值练习。

```
character*5 ch1,ch2,ch3,ch4*1,ch5*11
ch1='love'
ch2='chIna'
ch3='student'
ch4=ch2(3:3)
ch5=ch4//' '//ch1//'you!'
```

执行后，ch1 为'love_'(_表示空格)

ch2 为'china'

ch3 为'stude'

ch4 为'I'

ch5 为'I _ love_ you!'

需要注意的是：

① 例题中出现的字符连接符//，作用是将两个字符型数据连接起来，组成一个新字符型数据，如本例中 ch5。它是唯一的一个字符运算符。

② ch2(3：3)表示 ch2 的一个子串，即一个字符串的一部分称为该字符串的子串。通常表示为：

```
字符变量名(m：n)
```

其中 m 和 n 是整数或整型表达式，用来表示子串在字符串中的起止位置，取值范围为：字符串长度≥n≥m≥1。

(6) 复型变量的赋值语句中如果右边表达式中实部或虚部含有变量，应该用 complex 函数将实部和虚部组成复型数据再赋给复型变量。

例如：

```
integer c
```

```
complex a,b
a=(2.5,3.0)
b=cmplx(2.5*c,3.0)
```

3.2 输入和输出语句

【例 3-3】 问题：在例 3-1 中，我们通过赋值语句将需要计算的原始数据 1 和 2 分别保存到变量 a 和 b 中，那么除了通过赋值语句还有其他方法可以将数据 1 和 2 保存到变量 a 和 b 中吗？如果要用这个程序求 3 和 4 的和值，或者其他两个整数的和值时，应该如何修改程序最简单呢？

对于上述问题，可以通过引入输入语句解决。

程序修改如下：

```
integer a, b, c                          !定义整型变量 a、b、c
read *,a, b                              !从键盘输入数据到变量 a 和 b
c=a+b                                    !计算 a、b 的和
print *, "C=",a,"+",b,"=",c              !输出 c 的值
end
```

程序运行结果如图 3.2 所示。

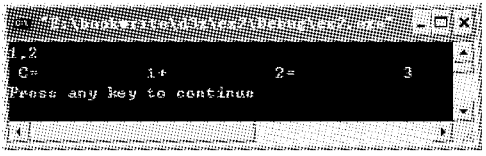

图 3.2 例 3-3 运行结果一

再一次运行程序结果如图 3.3 所示。

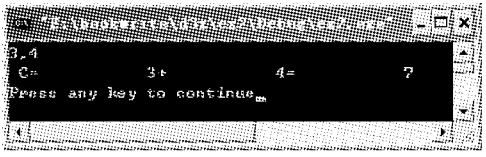

图 3.3 例 3-3 运行结果二

通过执行输入语句，每次运行程序时从键盘上输入两个整型数据到变量 a、b 中，增加了程序的灵活性，程序可以计算任意两个由用户从键盘输入的数据之和。

从上面例题可以看出，一个计算机程序通常包含三个部分，即输入、处理和输出。输入、输出就是把要加工的原始数据通过某种外部设备（如键盘、磁盘文件等）输入计算机的存储器中，且把处理结果输出到指定设备（如显示器、打印机或磁盘等），把数据以完整、有效的方式提供给用户。

输入输出数据时，需要程序告诉计算机三种信息：

一是输入输出哪些数据。

二是用何种格式输入输出(每个数据如何表示，占多少字符位，数据间如何分隔等)。

三是从什么设备上输入输出。

有了这些信息，计算机就可以对输入的数据，进行加工处理，然后将结果输出。

FORTRAN 语言提供了输入语句(read 语句)和输出语句(write 语句和 print 语句)，以实现上述数据传送的功能，输入输出的格式分为三种：

(1) 按系统隐含的格式输入输出(即表控格式输入输出、自由格式输入输出)。

(2) 按用户指定的格式输入输出(即有格式输入输出)。

(3) 无格式的输入输出。它是以二进制形式输入和输出数据，只适用于计算机内存与磁盘、磁带等之间的数据交换。

本节只介绍前两种输入输出格式。

3.2.1 表控输出输入

1. 表控输出

表控输出是最简单的输出方法。其输出格式不必用户自己说明，而是由系统做了隐含的规定，故也称为固定格式输出。

表控输出语句一般格式为：

```
print *,输出表
```

* 表示从系统隐含指定的输出设备(一般为显示器)上，按系统隐含规定的格式输出数据。

输出表由若干输出项组成，输出项可以是常量、变量、表达式，各输出项之间用逗号间隔。

例如 print * ,a,35.0,a * 2 是合法输出语句。执行此输出语句时，计算机按输出系统隐含规定的格式在显示器上输出 3 个实型数据。

说明：

(1) * 后面可以为空，即 print * 是合法输入语句，执行该语句，输出一空白行，相当于一个换行语句。

(2) 系统隐含规定的输出格式非常简单，数据按规定的输出宽度及显示形式输出，数据之间不添加分隔符。

(3) 输出语句中输出项如果是字符串，字符串中内容原样显示输出。

例如：

```
x=1.5;y=2.5;z=6.5
print *,"平均值=",(x+y+z)/3.0
end
```

输出结果如图 3.4 所示。

(4) 如果有多个输出语句时,每个 print 语句都从新的一行开始输出数据。

例如执行下面语句:

```
i=12;j=-25;a=12.345;b=245.5e2
print*,i,j
print*,a
print*,b
end
```

输出结果如图 3.5 所示。

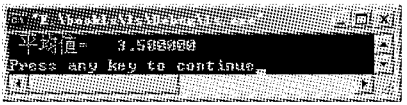

图 3.4 输出项为字符串时运行结果

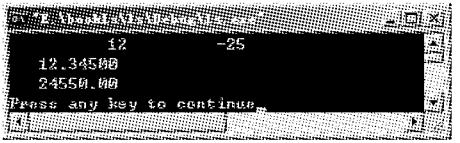

图 3.5 多个输出项时运行结果

三个 print 语句,分三行输出。

(5) 表控输出也可以写为以下形式:

```
write(*,*)输出表
```

其中,第一个 * 表示系统隐含指定的输出设备(显示器),第二个 * 表示表控输出。

2. 表控输入

使用表控输入语句,用户在输入数据时只要按照系统隐含规定的标准格式输入数据即可。

表控输入语句一般格式为:

```
read *,输入表
```

* 表示从系统隐含指定的输入设备(一般为键盘)上按系统隐含规定的格式输入数据。

例如 read*,a,b,c 是合法输入语句。执行此输入语句要求用户从键盘输入 3 个实型数据分别给 a、b、c 变量。

说明:

(1) * 后面可以为空,即 read * 是合法输入语句,执行该语句,等待用户按 Enter 键。

(2) 输入表必须由变量组成,可以有一个或多个变量,变量之间用逗号间隔,且可以是多个不同类型的变量。

下面语句是合法的变量定义语句和输入语句:

```
integer  i,j
real a,b
```

```
character*8 str1,str2*5
logical log1,log2
read*,str1,i,j,str2,a,log1,b,log2
```

(3) 输入数据时,数据按合法形式表示,输入数据的次序和类型要与输入表中各变量的次序和类型相一致。如果只输入一个数据,直接输入后按 Enter 键确定。如果输入多个数据,数据之间用逗号、空格或回车符间隔。例如:

```
        read*,a
输入方式: 12.5↙  (↙表示回车,下同)
        read*,a,b,c
输入方式: 12.5,2.6,31.4↙
或      12.5  2.6  31.4↙
或      12.5↙
        2.6↙
        31.4↙
```

(4) 如果输入时,输入数据个数少于输入表中变量个数则计算机将等待用户继续输入(光标闪烁),如果输入数据多余输入表中变量个数,多余数据不起作用。

(5) 如果有多个输入语句时,每个 read 语句都从新的一行开始读数据。

例如:

```
read*,i,j
read*,a
read*,b
```

执行上面语句时,应按以下方式输入 4 个数据:

```
12,25↙
25.5↙
3.6
```

第一个 read 语句依次读 12 和 25,赋予 i 和 j。第二个 read 语句从第二行开始读数,读入 25.5 赋予 a。第三个 read 语句从第三行开始读数,读入 3.6 赋予 b。

(6) 表控输入也可以写为以下形式:

```
read(*,*)输入表
```

其中,第一个 * 表示系统隐含指定的输入设备(键盘),第二个 * 表示表控输入。

3.2.2 格式化输出输入

在前面的例题中使用 print 和 read 命令时,都采用的是表控输入输出,这些程序的显示结果并不是很“清晰”和“美观”。格式化输出的目的就是要把数据有规划地进行版面设计后再显示,从而使输出结果更为清晰和整齐;格式化输入的目的是为了在某些情况下读取数据时,通过设置恰当的输入格式使之能够得到正确的结果。

下面对格式化输入输出作进一步的说明。

1. 格式化输出

格式化输出的一般格式为：

```
print  语句标号,输出项
语句标号  format(格式说明)
```

或

```
write (*,语句标号) 输出项
语句标号  format(格式说明)
```

由上可见，格式输出需要语句标号连接的输出语句和 Format 语句配合使用来实现。

Format 语句是“格式说明语句”，它是一个非执行语句，本身不产生任何操作，只是提供输出的格式。在 Format 语句中用相关编辑符指定输出格式，输出语句通过语句标号引用该 Format 语句，按照其指定的格式进行输出。

Format 语句可以出现在程序中程序单元说明语句之后和 end 语句之前的任何地方。

【例 3-4】 对例 3-3 的计算结果进行格式化输出。

程序编写如下：

```
integer a, b, c                          !定义整型变量 a、b、c
read *,a, b                              !输入数据到变量 a 和 b
c=a+b                                    !计算 a、b 的和
print 100, "C=",a,"+",b,"=",c            !输出 c 的值
100  format(a,i2,1x,a,i2,1x,a,i4)
end
```

程序运行结果如图 3.6 所示。

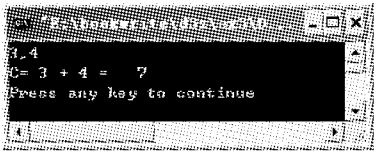

图 3.6　例 3-4 运行结果

当格式不复杂时，可以把输出格式直接写在 print 或 write 中。一般格式为：

```
print '格式说明符',输出项
```

或

```
write (*,'格式说明符') 输出项
```

例如：

```
print '(2i3)',m,n
write(*, '(2i3)')m,n
```

2. 格式化输入

格式化输入的形式与格式化输出的形式完全相同，只是把 print 或 write 换成 read 就可以了。

例如：

```
read '(2i3)',m,n
read(*, '(2i3)')m,n
```

在进行输入数据时，除使用文件外，通常不需要设置输入的格式。故下面对格式输入的介绍要简单些。

3. 格式说明

格式说明由编辑符组成，它的作用是将数据按照编辑符指定的格式输出输入。Fortran 语言中格式化输出输入的编辑符非常丰富，但是常用的并不多，这里只介绍最常使用的几个编辑符。

1）I 编辑符

I 编辑符用于整型数据的输出。

一般格式为：

```
rIw [.m]
```

表示以 w 个字符的宽度来输出整数，至少输出 m 个数字。r 为重复系数，为 1 时常省略。

I 编辑符输出的使用规则为：

当 w 等于输出项实际的数字位数时，直接输出。

当 w 大于输出项实际的数字位数时，在输出字段值左侧添加空格补足w个字符；即“右对齐，左补空格”。

当 w 小于输出项实际的数字位数时，将输出 w 个 * 。

例如：

```
print 100, 150,12,1500,22
100 format (i3,i5,i3,i5.3)
```

输出结果为：

```
150___12***__022
```

I 编辑符输入的使用规则为：在输入记录中按顺序截取 w 个字符存入对应的输入项。

2）F 编辑符

F 编辑符用于小数形式的实型数据的输出。

一般格式为：

```
rFw.d
```

表示以 w 个字符的宽度来输出实数(包含小数点)，小数部分占 d 个字符宽度。r 为重复系数，为 1 时常省略。

F 编辑符输出的使用规则为：

(1) 先处理小数部分：

如果输出项小数部分实际位数小于 d，则输出时小数部分右边以 0 补足到 d 位。

如果输出项小数部分实际位数大于 d，则保留 d 位，从 d+1 位开始四舍五入。

(2) 再整体处理实数：

当 w 等于输出项实际的数字位数时，直接输出。

当 w 大于输出项实际的数字位数时，在输出字段值左侧添加空格补足 w 个字符。即“右对齐，左补空格”。

当 w 小于输出项实际的数字位数时，将输出 w 个 * 。

例如：

```
    print 10,456.78,55.6855,123450.6789
10  format(3f9.3)
```

输出结果为：

```
__456.780___55.686*********
```

F 编辑符输入的使用规则为：在输入记录中按顺序截取 w 个字符存入对应的输入项。若截取的 w 个字符中不含小数点，则系统自动按 d 决定小数点的位置，若 w 个字符中含有小数点，则“自带小数点优先”。

3) E 编辑符

用于指数形式的实型数据的输出。

一般格式为：

```
rEw.d
```

表示以 w 个字符的宽度来输出实数，数字部分小数占 d 个字符宽度。r 为重复系数，为 1 时常省略。

E 编辑符输出的使用规则为：先处理小数部分，后整体处理指数型指数。处理方式同 F 编辑符。

需要注意的是：E 编辑符采用规格化的指数形式进行输出实数，即数字部分小数前面为 0，小数点后第一位为非 0 数字，指数部分占 4 列(E、指数符号位及两位指数)。

例如：

```
 print 10,456.78,55.6855,123450.6789
10  format(e9.3,e12.3,e6.3)
```

输出结果为：

```
0.457E+03___0.557E+02******
```

E 编辑符输入的使用规则同 F 编辑符。

4) L 编辑符

L 编辑符用于逻辑型数据的输出。