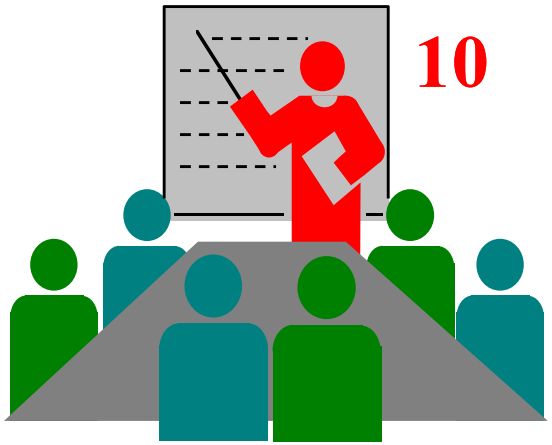
第四章　内部数据类型与表控I/O



4.0 概述

- FORTRAN 90预定义五种内部数据类型:
  - 整型(INTEGER)
  - 实型(REAL)
  - 复型(COMPLEX)
  - 字符型(CHARACTER)
  - 逻辑型(LOGICAL)

数据类型都具有参数化特性(KIND参数，种别),通过KIND参数对其进行细化。可直接使用内部数据类型说明数据对象(常量、变量、数组等)

- 内部数据类型分为两类:
  - 数值型(整型、实型和复型)
  - 非数值型(字符型和逻辑型)
- 在程序中需要选择和使用符合精度和范围要求的数据类型，根据需要可指定KIND值参数，种别。通过KIND值参数确定数据最佳的存储开销、精度和范围。表4-1给出内部数据类型的不同KIND值参数及存储开销。

表4-1 内部数据类型KIND值参数及存储开销

类型	子类型	KIND 值	字节数	说明
复型	COMPLEX	4 或 8	8 或 16	与缺省有关,COMPLEX 有 8 或 16 个字节. 初始缺省为 8,缺省与实型缺省值有关(2 倍)
	COMPLEX(4)	4	8	
	DOUBLE COMPLEX	8	16	与 COMPLEX(8)等同
	COMPLEX(8)	8	16	
字符型	CHARACTER	1	1	CHARACTER 与 CHARACTER(1)等同. 1 是 KIND 值,不是字符串长度
	CHARACTER*len	1	len	len 是字符串长度. 对 Intel CPU 有 1≤len≤ 65535,对 Alpha CPU 有 1≤len≤$2^{31}-1$
逻辑型	LOGICAL	2、4 或 8	2、4 或 8	与缺省有关,LOGICAL 有 2、4 或 8 个字节. 初始缺省为 4
	LOGICAL(1)	1	1	
	LOGICAL(2)	2	2	
	LOGICAL(4)	4	4	
	LOGICAL(8)	8	8	仅 Alpha 系统有效

4.1 常量

●常量是直接写在程序中且在程序运行期间保持不变的数据，在程序中直接生成并直接用于计算和处理,如125、12.5、(12.5,25.5)、“China”、.TRUE.等。

●五种内部数据类型常量:

- ◆整型常量：整数。
- ◆实型常量：实数。
- ◆复型常量：复数。
- ◆字符型常量：字符串。
- ◆逻辑型常量：逻辑值，布尔值。

4.1 常量

整数语法

●整数表示

FORTRAN 90允许在程序中表示2至36进位制整数。如十进制整数3994575可在程序中按下面形式描述(输出结果相同):

```
PRINT*,2#111100111100111001111    ！2进制整数
PRINT*,7#45644664                  ！7进制整数
PRINT*,+8#17171717                 ！8进制整数
PRINT*,#3CF3CF                    ！16进制整数
PRINT*,+17#2DE110                  ！17进制整数
PRINT*,3994575                    ！10进制整数
PRINT*,36#2DM8F                    ！36进制整数
```

●整数范围

●整数示例

4.1 常量

- 实数语法
- 实数表示

FORTRAN 90只允许十进制实数，有小数形式和指数形式。

```
PRINT*,-12.5                    ！实数-12.5
PRINT*,0.125, .125              ！实数0.125
PRINT*,125.0, 125.              ！实数125.0
PRINT*,1.25E1, 0.125E2, .125E2  ！实数12.5
```

- 实数范围
- 实数示例
- 实数性质
- 标准形式

4.1 常量

- 复数语法

  <复数>→(<实部>,<虚部>)

  <实部>→<整数>|<实数>

  <虚部>→<整数>|<实数>

- 复数表示

  代数表示: 12.5+23.4i

  F90表示: (12.5,23.4),(12,23.4),(12.5,23),(12,23)

- 复数范围

  复数分单精度复数和双精度复数,复数范围由其实部和虚部的范围决定。单精度复数存储开销为8字节,实部和虚部各为4字节,其范围为2个单精度实数范围。双精度复数存储开销为16字节,实部和虚部各为8字节,其范围为2个双精度实数范围。精度由实部和虚部最高实数精度决定。

- 复数示例

4.1 常量

●字符串语法

＜一般字符串＞→(‘|“){＜计算机系统允许使用字符＞}(’|”)说明:

①引号为英文引号，且必须配对。

②字符串内含引号解决办法：交替使用法、重复使用法。

③字符串长度为引号之间字符个数。

④字符可为非打印字符，表示形式如表4-9所示。如换行符“\n”。

●字符串表示

‘A’

"I am a student.“

"I'm a student.“

'He said:"We are going to Suzhou." ‘

'I"m a student. ‘

"He said:""We are going to Suzhou."" "

4.1 常量

●逻辑值语法

<逻辑值>→(.true.|.false.)[_ <逻辑型KIND值>]

<逻辑型KIND值>→1|2|4|8

说明:

①缺省KIND值为4。

②对于逻辑值.true.，存储单元内每位为1，可视为整数-1。

③对于逻辑值.false.，存储单元内每位为0，可视为整数0。

④逻辑值可与整数一起参与运算，如: 5+.true.值为4。

●逻辑值表示

.true. .false. .TRUE. .FALSE. .True. .FalsE.

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

<整数>→[<符号>][[<基数>]#]<数字>{<数字>}[_ <整型KIND值>]
<符号>→+|-
<基数>→2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|
14|15|16|17|18|19|20|21|22|23|24|25|
26|27|28|29|30|31|32|33|34|35|36
<数字>→1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|G|
H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|
X|Y|Z
<整型KIND值>→1|2|4|

说明：KIND值只对十进制整数有效。

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

FORTRAN 90通过KIND值确定整数的存储开销、最大位数和取值范围，如表4-2所示。缺省KIND值则取默认值4。

KIND值	字节数	二进位数	取值范围
1	1	8	-128~127,0~255
2	2	16	-32768~32767,0~65535
4	4	32	-2147483648~2147483647

说明：超出取值范围的整数,不产生语法和运行错误,而产生错误的整数，如整数32771_2的实际结果是-32765。在程序中要特别注意整数取值范围。

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

[例4.1]判定整数,指出合法整数和非法整数？说明原因。

合法整数:-128、+3276、-52467、+ 125、8#57、#4DE、16#458A。

非法整数如表4-3所示。

非法整数	说明
358_5	5不是有效整型KIND值
5,234	不允许出现逗号“,”
130_1	超出1字节127范围
8#79	8进制数字不允许出现9
8#5,12	不允许出现逗号“,”
8# 245	不允许出现空格
8#4532_2	非10进制不允许出现整型KIND值
#3F7G	16进制不允许出现G

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

<实数>→<小数形式实数>|<指数形式实数>
<小数形式实数>→[<符号>](<整数部分>.[<小数部分>]|
[<整数部分>].<小数部分>)[_<实型KIND值>]
<整数部分>→<十进制数字>{<十进制数字>}
<小数部分>→<十进制数字>{<十进制数字>}
<十进制数字>→0|1|2|3|4|5|6|7|8|9
<实型KIND值>→4|8
<指数形式实数>→<数值部分><指数部分> [_ <实型KIND值>]
<数值部分>→<十进制整数>|<小数形式实数>
<十进制整数>→[<符号>]<整数部分>
<指数部分>→(E|e|D|d) <十进制整数>
说明：若KIND值为4，则为单精度，KIND值为8，则为双精度。

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

FORTRAN 90通过KIND值确定实数存储开销(即字节数)、精度和取值范围，如表4-6所示。缺省KIND值则取默认值4。

KIND值	字节	二进位	精度	取值范围(绝对值)
4	4	32	7位	0，[10^{-38}，10^{+38}]
8	8	64	15位	0，[10^{-308}，10^{+308}]
说明：大于最大值，产生上溢错，小于最小值，按0处理。				

```
PRINT*,1.25E1, 0.125E2, .125E2    ！实数12.5
```

- 实数范围
- 实数示例
- 实数性质
- 标准形式

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

[例4.2]判定实数,指出合法实数和非法实数？说明原因。

合法整数:-12.8、+3.276_4、-.125、125.、125E-1_4、18.5D59

非法实数如表4-4所示。

非法实数	说明
35.8_5	5不是有效实型KIND值
5,234.23	不允许出现逗号“,”
12 8.5_4	不允许出现空格
23.45E1.5	指数部分不允许出现小数
E+5	E前不允许为空
+ 18.9D10_8	D指数不允许指定实型KIND值
-1.25E1 2	指数部分不允许出现空格
12.5E$3	指数部分不允许出现“$”

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

实数有四个性质:

- 实数可写成小数形式实数,也可写成指数形式实数。
- 实数写成指数形式实数可有多种等价写法,如1.2345E1和0.12345E2为同一实数。
- 指数形式实数的指数部分决定小数点的位置,指数部分为正,表示小数点向右移动若干位,指数部分为负,表示小数点向左移动若干位。
- 指数形式实数数值部分决定有效数字位数(即精度),超出有效数字位数,将产生误差。指数部分决定实数大小。

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

标准化形式通常有两种:

●数值部分绝对值小于1(即整数部分为0)且大于等于0.1(即小数点后第一个数字不能为0)。不符合这个标准化条件,可增大或减少指数值(移动小数点位置)来达到标准化条件。如实数125.34的标准化实数为0.12534E3。

●数值部分绝对值小于10且大于等于1(即整数部分为1位数)。不符合这个标准化条件,可增大或减少指数值(移动小数点位置)来达到标准化条件。如实数125.34的标准化实数为1.2534E2。Digial Visual FORTRAN 5.0采用这种标准化形式。

实数在程序中可以不按标准化形式表示,但程序在编译和执行时,自动转化为标准化形式进行存储或输出。实数均按标准化形式存储,不同编译系统采用标准化形式不同。

4.1 常量

- 概述
- 整数
- 实数
- 复数
- 字符串
- 逻辑值

[例4.4]判定下列复数的类型和字节数。
(12,35)、(25,12.5)、(8#745,125_8)、(325.45_8,854)、(35.78,23.345D2)、(387.885,1.435)、(7745.35,5.4553E-12_8)

解:判定结果如表4-8所示。

复数	类型	字节数
(12,35)	单精度复数	8
(25,12.5)	单精度复数	8
(8#745,125_8)	单精度复数	8
(325.45_8,854)	双精度复数	16
(35.78,23.345D2)	双精度复数	16
(387.885,1.435)	单精度复数	8
(7745.35,5.4553E-12_8)	双精度复数	16

4.2 变量

●变量是在程序运行过程中可随时改变的数据。在任何时刻，一个变量只能有且必须有一个确定的值。

●FORTRAN 90为每个变量分配若干连续的存储单元来存放变量的值。KIND值为4的整型变量num的存储表示如图4-2所示。

4.2 变量

- 五种类型:整型、实型、复型、字符型和逻辑型变量。
- 变量KIND值意义与常量KIND值意义相同,如表4-1所示。
- 变量类型需要通过类型声明语句来说明。
- 变量类型声明有两种形式:
  - ◆显式声明(优先)

    INGETER i，j，k，num，max

    REAL a，b，area，score
  - ◆隐式声明(隐含约定)

隐式声明：预先定义且无须显式声明的类型声明,称为I-N规则。

I-N规则规定：凡是变量名以字母I、J、K、L、M、N、i、j、k、l、m、n开头的变量被默认为整型变量，以其它字母开头的变量被默认为实型变量。如：num为整型变量，area为实型变量。

IMPLICIT语句：定义新I-N规则。

```
IMPLICIT NONE
IMPLICIT INTEGER(A,B),REAL(I,K,L:N),CHARACTER(C)
```

4.2 变量

●语法

<整型变量声明语句>→INTEGER[([KIND=]<整型KIND值>)][::]<整
型变量名表>

<整型变量名表>→<变量名称>[=<整数>]{,<变量名称>[=<整数>]}

●形式

INTEGER ----- 一般默认取k=4

INTEGER(k)或INTEGER(KIND=k)或INTEGER*k ---- k:1,2,4,8

●示例

```
INTEGER(1) e,d,e    !声明KIND值为1的3个整型变量
INTEGER  f          !声明KIND值为4的1个整型变量
INTEGER(2)::a=15,b  !a初值为15,b初值为0
INTEGER(1)::c=8#127 !c初值为8进制数127,即10进制数87
BYTE :: c=8#127     !c初值为8进制数127,即10进制数87
```

说明：符号“::”表示在说明中可以赋予初值。取值范围同整数。

4.2 变量

- 语法

<实型变量声明语句>→REAL[([KIND=]<实型KIND值>)][::]<实型变量名表>

<实型变量名表>→<变量名称>[=<实数>]{,<变量名称>[=<实数>]}

- 形式

REAL ----- 一般默认取k=4

REAL(k)或REAL(KIND=k)或REAL*k ---- k:4,8

DOUBLE PRECISION ----- 双精度，等价于REAL(8)

- 示例

```
REAL(8) e,d,e       !声明KIND值为8的3个双精度实型变量
REAL  f             !声明KIND值为4(缺省)的1个实型变量
REAL(4)::a=125.5,b  !a初值为125.5,b初值为0.0
REAL(8)::c=12.5E+3  !c初值为12500.0
DOUBLE PRECISION ::c=12.5D-10  !c初值为12.5×10^-10
```

说明：符号“::”表示在说明中可以赋予初值。取值范围同整数。

4.2 变量

- 语法

  <复型变量声明语句>→COMPLEX[([KIND=]<复型KIND值>)][::]<复型变量名表>

  <复型变量名表>→<变量名称>[=<复数>]{,<变量名称>[=<复数>]}

- 形式

  COMPLEX ----- 一般默认取k=4

  COMPLEX(k) 或 COMPLEX(KIND=k) 或 COMPLEX*2k ---- k:4,8

  DOUBLE COMPLEX ----- 双精度，等价于COMPLEX(8)

- 示例

```
COMPLEX(8)e,d,e      !声明KIND值为8的3个双精度复型变量
COMPLEX  f           !声明KIND值为4(缺省)的1个复型变量
COMPLEX(4)::a=(5.7,8.5),b       !a初值为(5.7,8.5)
COMPLEX ::d=(12.5,25.3)         !d初值为(12.5,25.3)
DOUBLE  COMPLEX  ::c=(12.5D-10,125)!c 初 值 为 (12.5D-10,125D0)
```

4.2 变量

- 概述
- 类型
- 整型
- 实型
- 复型
- 字符型
- 逻辑型

●语法

<字符型变量声明语句>→CHARACTER[<类型参数>][::]<变量名表>

<类型参数>→(<无符号整数>)|([LEN=]<无符号整数>)|*<无符号整数>

<变量名表>→<变量名称>[*<无符号整数>][=<字符串>]{,<变量名称>[*<无符号整数>][=<字符串>]}

<字符串>→<一般字符串>|<H字符串>|<C字符串>

●形式

CHARACTER ----- 一般默认取k=1

CHARACTER(k)或CHARACTER(LEN=k)或CHARACTER*k -- k:正整数

●示例

```
CHARACTER(4)e,d,e  !声明长度为4的3个字符型变量
CHARACTER  f，g*5    !声明长度为1(缺省)和为5的2个字符型变量
CHARACTER(1)::c='$'，s*4='This'
```

4.2 变量

- 概述
- 类型
- 整型
- 实型
- 复型
- 字符型
- 逻辑型

●语法

<逻辑型变量声明语句>→LOGICAL[([KIND=]<逻辑型KIND值>)]
[::]<变量名表>

<变量名表>→<变量名称>[=<逻辑值>]{,<变量名称>[=<逻辑值>]}

●形式

LOGICAL ----- 一般默认取k=4

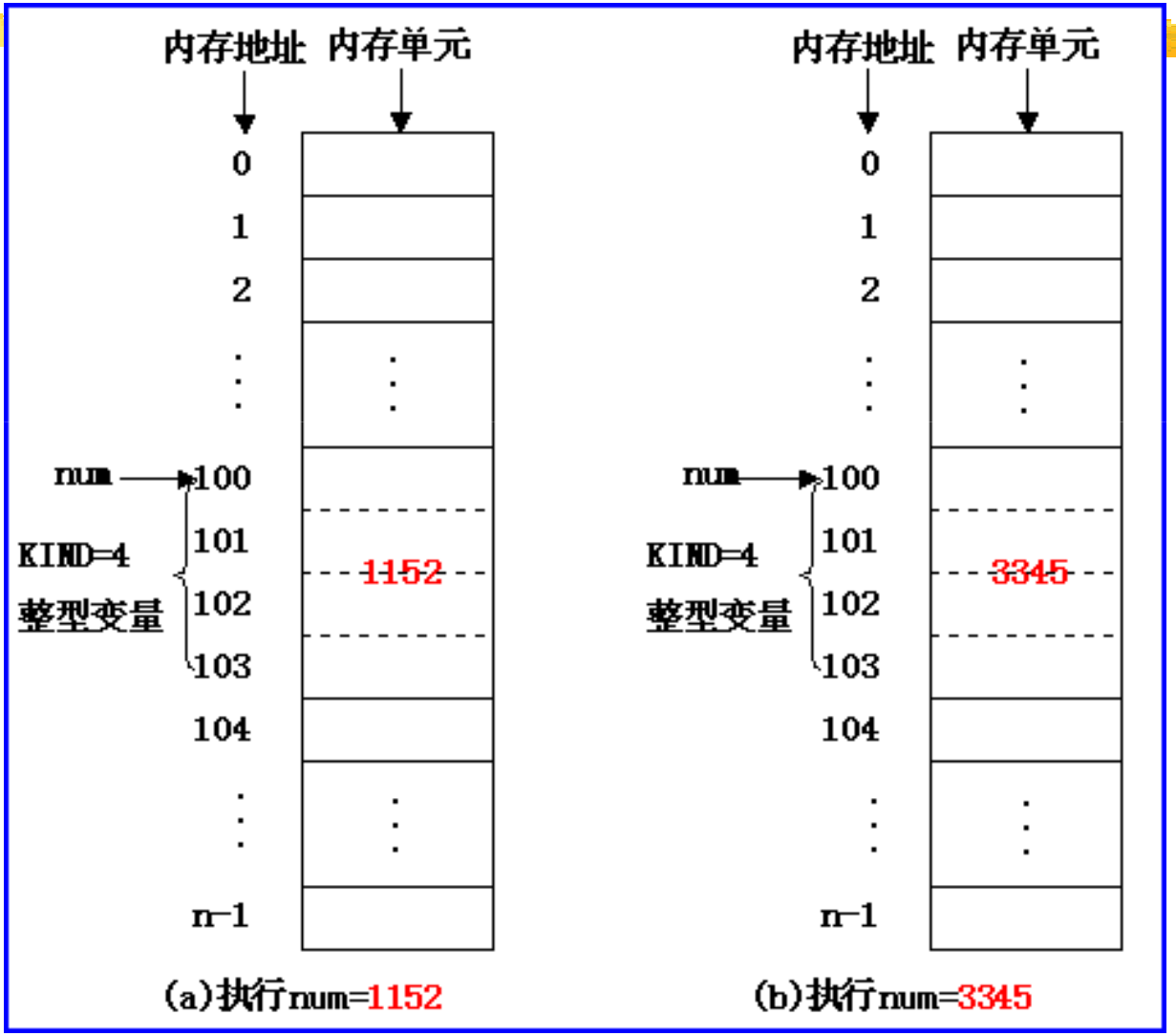
LOGICAL(k) 或 LOGICAL(KIND=k) 或 LOGICAL*k ---- k:1,2,4,8

●示例

```
LOGICAL(1) e,d,e    !声明KIND值为1的3个逻辑型变量
LOGICAL  f          !声明KIND值为4的1个逻辑型变量
LOGICAL(2)::a=.TRUE.,b  !a初值为.TRUE.,b初值为.FALSE.
```

说明：符号“::”表示在说明中可以赋予初值。

变量存储表示


内存地址
内存单元
0
1
2
num
100
101
102
103
104
n-1
KIND=4
整型变量
1152
(a)执行num=1152
内存地址
内存单元
0
1
2
num
100
101
102
103
104
n-1
KIND=4
整型变量
3345
(b)执行num=3345

4.3 表达式

●表达式是由操作数、操作符和圆括号组成的式子。

- ◆操作数：常数、常量、变量或函数
- ◆操作符：算术、字符、关系或逻辑操作符号(运算符)
- ◆圆括号: ( ),用以改变操作数的操作顺序。

●不同类型的操作数和操作符组成的表达式其操作方式、操作规则及操作结果亦不相同。

●四种类型表达式:

- ◆算术表达式: (number+12)*count
- ◆字符表达: str1(i:j)//str2
- ◆关系表达式: (x+y)<1.25E-25
- ◆逻辑表达式: .NOT.(L1 .AND. L2 .OR. L3)

4.3 表达式

- 算术表达式：算术操作数、算术操作符和圆括号组成。
  - 算术操作数：常数、常量、变量或函数(整、实、复)
  - 算术操作符：+、-、*、/、**
  - 圆括号:( ),用以改变操作数的操作顺序。
- 优先级：运算次序按优先级由高到低依次运算。

  ** -> (*,/) -> 单目(+,-) -> 双目(+,-)
- 结合规则:
  - 左结合: 相邻同级运算符从左向右依次运算。如：(*,/)(+,-)
  - 右结合: 相邻同级运算符从右向左依次运算。如：**
- 示例

a+b-c+d*e/f-g**h**i⇔((a+b)-c)+((d*e)/f)+(g**(h**i))

(a+a*b**2-8*c*d)/(-e+f**g**3)
- 操作数类型转换：允许混合运算，按最高级进行转换。

12+13+14.5⇔(12+13)+14.5⇔25+14.5⇔25.0+14.5⇔39.5

4.3 表达式

●字符表达式：字符操作数、字符操作符和圆括号组成。

◆字符操作数：字符、字符串、字符型变量或字符型函数

◆字符操作符：求子串(i:j),连接符//

◆圆括号:（）,用以改变操作数的操作顺序。

●示例

```
CHARACTER*19 :: str,string='This is a computer.'
CHARACTER*1 substr1*4,substr2*2,substr3,substr4*9
n=2;m=4
substr1=string(:4)        !子串为 “This”
substr2=string(6:7)       !子串为 “is”
substr3=string(n+2:m)     !子串为 “a”
substr4=string(11:)       !子串为 “computer.”
str=substr1//‘ ’//substr2//‘ ’//substr3//‘ ’//substr4
                          !str=‘This is a computer.’
```

4.3 表达式

●关系表达式：算术或字符表达式、关系运算符组成,结果为逻辑值。

关系运算符：< 、 <= 、 > 、 >= 、 == 、 /=

.LT.、.LE.、.GT.、.GE.、.EG.、.NE.

说明：

①对于复型关系比较运算，只有等于和不等于两种。

②对于字符型关系比较运算，按字典次序(ASCII码值大小)进行。

●示例

45<25.5 !结果为.FALSE.

(12+2*5) .GE. 10 !结果为.TRUE.

(2.5,5.8) .NE. (2.5,5.87) !结果为.TRUE.

(2.5,5.8) .GE. (2.5,5.87) !该关系表达式非法

'banana'<='apple' !结果为.FALSE.

'apple'<='apple' !结果为.TRUE.

'This is a '//'pen.'<='This is a '//'pencil.' ⇔.TRUE.

4.3 表达式

●逻辑表达式：逻辑操作数、逻辑操作符和圆括号组成。

◆逻辑操作数：逻辑值、逻辑型常量、逻辑型变量或逻辑型函数

◆逻辑操作符：.NOT.、.AND.、.OR.、.XOR.、.EQV.、NEQV.

◆圆括号:（）,用以改变操作数的操作顺序。

逻辑表达式功能如表4-16所示。

●优先级：运算次序按运算符优先级由高到低依次运算。

.NOT. -> .AND. -> .OR. -> .XOR.、.EQV.、.NEQV.

●结合规则:同级运算符均采用左结合。

●示例

```
(12+a)*3>a+3*b .AND. a+b<3*a+2*b
'This'//s1<'price' .OR. 'student'= =s1
90<=score      .and.      score<=100      ⇔
90<score<100
```

4.3 表达式

●整型运算精确,无误差,超出取值范围，将产生错误整数，通过扩大KIND值来解决。如130_1超范围,可用130_2来表示就不会超范围。

●实型运算会产生误差。受有效位数限制,单精度实数有效位数7位,双精度实数有效位数15位。如:11111.1*1111.11=12345654.321,实际结果近似为$1.2345654\times10^{+7}$。

●产生误差情况:

◆加减运算次序不当产生误差

精确计算：0.001+3257845.0-3257840.0=5.001

程序计算：0.001+3257845.0-3257840.0=5.000

改进计算：3257845.0-3257840.0+0.001=5.001

解决办法是交换加减运算次序,避免两个相差很大的数相加或相减。

◆乘除运算次序不当产生误差

精确计算：(1.0/3.0)*3.0=1.0

程序计算：(1.0_4/3.0_4)*3.0_8=1.00000002980232

改进计算：(3.0_8/3.0_4)*1.0_4=1.00000000000000

解决办法是交换乘除运算次序,避免两个非整除的数相除。

表4-16 逻辑运算符功能

A 值	B 值	.NOT.A	A.AND.B	A.OR.B	A.XOR.B	A.EQV.B	A.NEQV.B
.TRUE.	.TRUE.	.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.	.FALSE.
.TRUE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.FALSE.	.TRUE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.

4.4 赋值语句

- 赋值语句：计算表达式的值并将其赋给一个变量、数组元素。
- 赋值语句语法描述:
  - <赋值语句>→<变量名称>=<表达式>
  - <表达式>→<算术表达式>|<字符表达式>|<逻辑表达式>
- 说明:

①赋值号左边变量类型和右边表达式类型必须赋值兼容。

②赋值语句中左边变量和右边表达式不能交换。

如“total=a+b*c”不能写成“a+b*c=total”。“a=b”与“b=a”不同。

左边变量类型	右边表达式类型	赋值语句类型
INTEGER(整型)	INTEGER,REAL,COMPLEX	算术赋值语句
REAL(实型)	INTEGER,REAL,COMPLEX	算术赋值语句
COMPLEX(复型)	INTEGER,REAL,COMPLEX	算术赋值语句
CHARACTER(字符型)	CHARACTER	字符赋值语句
LIGICAL(逻辑型)	LIGICAL	逻辑赋值语句

4.4 赋值语句

- 算术赋值语句：左边变量和右边表达式类型均为整型、实型或复型。
- 算术赋值语句在执行中会发生强制类型转换,转换后值赋于左边变量。
- 在使用算术赋值语句时,应尽可能保持赋值号两侧类型相同,
- 示例：

```
INTEGER(2)  A
INTEGER(4)  B
A=123456
B=12.5*1.5
```

上面算术赋值语句执行后,a的值不是123456,而是-7616,b的值不是18.75,而是18。

算术赋值语句类型转换是算术赋值语句的难点,读者应重点掌握。表4-19给出算术赋值语句类型转换说明。表4-20给出算术赋值语句类型转换示例。

4.4 赋值语句

- 字符赋值语句：左边变量和右边表达式类型均为字符型。
- 字符赋值语句在执行中会发生强制长度转换,转换后值赋予左边变量。
  - 右边长度小于左边,将表达式值赋予变量左侧,不足补空格。
  - 右边长度大于左边,将表达式值左侧部分赋予变量,多余截去。
- 示例：

```
CHARACTER*4 str1*8，str2*17,str3,str4
str1='students'
str2='Are you '//str1//'?'
str3='are'
str4='students'
str2(9:16)='teachers'
```

4.4 赋值语句

- 逻辑赋值语句：左边变量和右边表达式类型均为逻辑型。
- 逻辑赋值语句在执行会发生强制KIND值转换,改变存储,不改变逻辑值。
- 示例：

```
INTEGER :: i=150
LOGICAL(2) log1，log2
LOGICAL(4) log3
log1=.TRUE.
log2=log1 .AND. i>100
log3=log2
```

表4-20 算术赋值语句类型转换示例

变量a类型	表达式	变量a值	说明
INTEGER(1)	A=125	125	未超出A范围
INTEGER(2)	A=32771_4	-32765	超出A范围
INTEGER(1)	A=123.45	123	整数部分未超范围
INTEGER(2)	A=32.77145E+3	-32765	整数部分超范围
INTEGER(1)	A=(123.4,25.5)	123	实部整数部分未超范围
REAL(4)	A=125	125.0	未超出A有效位数
REAL(4)	A=1234567911	1234567900.0	超出A有效位数
REAL(4)	A=1.234567911D9	1234567900.0	超出A有效位数
COMPLEX(4)	A=125	(125.0,0.0)	未超出A实部有效位数
COMPLEX(4)	A=1.234567911D9	(1.2345679E9,0.0)	超出A实部有效位数

4.10 习题四

The end

文件名格式：班级 学号 姓名 简略实验名称

邮件标题同文件名

Any questions please 发送至

xingzhengwu@163.com