Introdução ao Fortran 90 - 2

Alexandre Diehl

Departamento de Física – UFPel

Identificador na forma de Matriz

Definição 1

Um identificador na forma de uma matriz consiste de um conjunto retangular de elementos, todos do mesmo tipo e espécie do tipo, também chamados de variáveis compostas homogêneas.

Definição 2

Uma matriz é um grupo de posições na memória do computador, as quais são acessadas por intermédio de um único nome, fazendo-se uso dos subscritos da matriz.

- Uma matriz de dimensão 1 é chamada de vetor.
- Uma matriz pode ter até 7 subscritos, cada um relacionado com uma dimensão da matriz.

Identificador na forma de Matriz

Definição 1

Um identificador na forma de uma matriz consiste de um conjunto retangular de elementos, todos do mesmo tipo e espécie do tipo, também chamados de arranjos.

Definição 2

Uma matriz é um grupo de posições na memória do computador, as quais são acessadas por intermédio de um único nome, fazendo-se uso dos subscritos da matriz.

- Os índices de cada subscrito da matriz são constantes inteiras, que começam em 1 ou a partir de um intervalo fornecido.

Declaração de Matrizes - Alocação Fixa

Vetor (Dimensão 1)

```
INTEGER, PARAMETER :: N = 100
REAL, DIMENSION (N) :: B
REAL, DIMENSION(10) :: A
INTEGER, DIMENSION(0:9) :: C
END
```

- Posto (*rank*): número de dimensões da matriz. Escalar (posto 0), vetor (posto 1), matriz (posto ≥ 2)
- Extensão (*extent*): número de componentes da cada dimensão da matriz
- Forma (*shape*): vetor cujos componentes são a extensão de cada dimensão da matriz.
- Tamanho (*size*): número total de elementos da matriz.

Declaração de Matrizes - Alocação Fixa

Matriz (Dimensão 2)

```
INTEGER, PARAMETER :: N = 2, M = 5
REAL, DIMENSION (N,M) :: C
REAL, DIMENSION (0:N,0:M) :: D
REAL, DIMENSION(-3:4,7) :: A
INTEGER, DIMENSION(8,0:8) :: B
END
```

Matriz A

- Posto (*rank*): 2
- Extensão (*extent*): 8 e 7
- Forma (*shape*): (/8,7/)
- Tamanho (*size*): 56

Matriz B

- Posto (*rank*): 2
- Extensão (*extent*): 8 e 9
- Forma (*shape*): (/8,9/)
- Tamanho (*size*): 72

Declaração de Matrizes - Alocação Fixa

```
CHARACTER(LEN=25), DIMENSION(5) :: nome
nome(1) = 'Joao Victor'
nome(2) = 'Maria Joana'
nome(3) = 'Alexandre Diehl'
nome(4) = 'Bruno Duarte'
nome(5) = 'Vinicius Silva'
END
```

- Vetor nome, com posto 1 e tamanho 5:

  nome(1) nome(2) nome(3) nome(4) nome(5)

- Cada elemento do vetor nome é do tipo CHARACTER com até 25 caracteres.

Construtores de Matrizes - Vetores

- Usados para inicializar os elementos de um vetor
- Forma geral do construtor: (/ < lista de valores> /)

```
implicit none
real, dimension(4) :: A = (/1.0,3.0,-1.5,4.5/)
integer,dimension(3), parameter :: B = (/1,4,6/)
real(8), dimension(4) :: C
C = (/1.0d0, 3.0d0, -1.5d0, 4.5d0/)
end
```

- Vetor A (variável), com posto 1 e tamanho 4:

  A(1) = 1.0 A(2) = 3.0 A(3) = -1.5 A(4) = 4.5

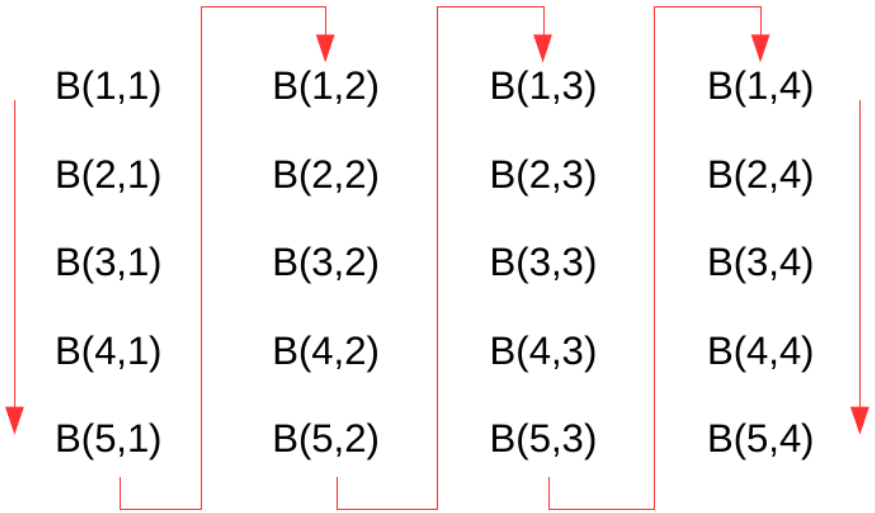
- Vetor B (constante), com posto 1 e tamanho 3.

  B(1) = 1 B(2) = 4 B(3) = 6

Matriz multidimensional: ordem dos elementos

O ordenamento é feito variando primeiro o índice da primeira dimensão, depois da segunda dimensão e assim por diante.

$$B = \begin{pmatrix} 1 & 4 & 0 & 10 \\ -2 & 4 & 3 & 0 \\ 5 & 3 & 0 & 11 \\ 7 & 1 & 2 & 9 \\ 11 & 1 & 4 & 4 \end{pmatrix}$$

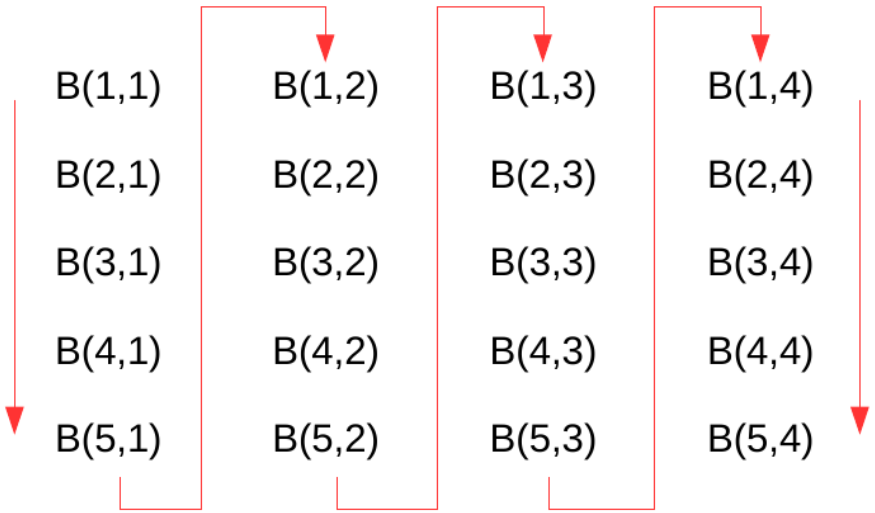


- Os arranjos (matrizes) multidimensionais são organizados por colunas
- O acesso aos elementos de uma matriz multidimensional é mais rápido pelas suas colunas

Matriz multidimensional: ordem dos elementos

```
INTEGER, DIMENSION(5,4) :: B
B(1,1) = 1 ; B(1,2) = 4; B(1,3) = 0; B(1,4) = 10
B(2,1) = -2; B(2,2) = 4; B(2,3) = 3; B(2,4) = 0
B(3,1) = 5 ; B(3,2) = 3; B(3,3) = 0; B(3,4) = 11
B(4,1) = 7 ; B(4,2) = 1; B(4,3) = 2; B(4,4) = 9
B(5,1) = 11; B(5,2) = 1; B(5,3) = 4; B(5,4) = 4
print*,B
END
```

$$B = \begin{pmatrix} 1 & 4 & 0 & 10 \\ -2 & 4 & 3 & 0 \\ 5 & 3 & 0 & 11 \\ 7 & 1 & 2 & 9 \\ 11 & 1 & 4 & 4 \end{pmatrix}$$



Construtores de Matrizes Multidimensionais

```
INTEGER, DIMENSION(2,2) :: A
REAL, DIMENSION(2,3) :: B
A = reshape((/1.0,0.0,0.0,1.0/),shape=(/2,2/))
B = reshape(source=(/1,-2,4,4,0,3/), shape=(/2,3/))
print*,A
print*,B
END
```

- Função Intrínseca SHAPE(SOURCE):
  retorna um vetor inteiro padrão, contendo a forma da matriz ou escalar SOURCE
- Função Intrínseca RESHAPE(SOURCE,SHAPE):
  retorna uma matriz com forma dada pelo vetor inteiro SHAPE e tipo e espécie iguais aos da matriz SOURCE

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 4 & 0 \\ -2 & 4 & 3 \end{pmatrix}$$

Construtores de Matrizes Multidimensionais

```
REAL, DIMENSION(2,3) :: B
B = reshape(source=(/1,-2,4,4,0,3/),shape=(/2,3/),order=(/2,1/))
print*,B
END
```

- Função Intrínseca RESHAPE(SOURCE,SHAPE,ORDER):

  retorna uma matriz com forma dada pelo vetor inteiro SHAPE e tipo e espécie iguais aos da matriz SOURCE, armazenados de acordo com a ordem das dimensões listadas no arrranjo ORDER (opcional).

$$B = \begin{pmatrix} 1 & -2 & 4 \\ 4 & 0 & 3 \end{pmatrix}$$

Alocação Dinâmica de Matrizes

Definição 1

Usada quando não queremos usar alocação fixa para matrizes durante a etapa de compilação do código-fonte.

Definição 2

Permite que a extensão, forma e tamanho, ou seja, o espaço de memória utilizado, da matriz seja alocado durante a execução do código-fonte.

Definição 3

Permite que o espaço de memória alocado à matriz seja liberado durante a execução do código-fonte.

Exige o uso de dois comandos

- No campo de declaração usamos o atributo ALLOCATABLE para declarar o posto da matriz
- O posto (número de dimensões) é indicado pelo número de símbolos : separados por vírgula
- No campo de execução, uma vez conhecido o tamanho da matriz, alocamos o espaço de memória usando a comando ALLOCATE

```
REAL, DIMENSION(:), ALLOCATABLE :: A
INTEGER, DIMENSION(:,:), ALLOCATABLE :: B
INTEGER :: N, M
READ*,N, M
ALLOCATE (A(N))
ALLOCATE (B(N,M))
END
```

→ Matriz A: posto 1, tamanho N → Matriz B: posto 2, forma (/N,M/), tamanho N*M

Liberando o espaço de memória utilizado

- Não sendo mais necessárias, os espaços de memória utilizados pelas matrizes podem ser liberados usando o comando DEALLOCATE
- Sem o comando DEALLOCATE os espaços de memória serão liberados na finalização da execução do código (comando END)

```
REAL, DIMENSION(:), ALLOCATABLE :: A
INTEGER, DIMENSION(:,:), ALLOCATABLE :: B
INTEGER :: N, M
READ*,N, M
ALLOCATE (A(N))
ALLOCATE (B(N,M))
! bloco de comandos
DEALLOCATE (A)
DEALLOCATE (B)
END
```

Operações com Matrizes

- No Fortran 90 as matrizes são tratadas como um objeto único
- Para que as operações envolvendo matrizes sejam possíveis, as matrizes consideradas devem ser conformáveis (mesma forma)
- Não confunda operações com matrizes em Fortran 90 com operações entre matrizes matemáticas

→ o produto $C = A \times B$ de duas matrizes, $A_{mn}B_{nl}$, produzirá uma matriz C_{ml} numa multiplicação entre matrizes matemáticas

Operações com Matrizes

- Atribuir valor constante para uma Matriz

```
REAL, DIMENSION(20) :: A = 1.0, B = -1.0
INTEGER, DIMENSION(100) :: C
REAL, DIMENSION(:,:), ALLOCATABLE :: D
INTEGER :: N, M
READ*,N, M
ALLOCATE (D(N,M))
C = 5
D = 0.0
END
```

→ Todos os elementos da matriz são igualados à constante

→ Zere sempre uma matriz alocada

Operações com Matrizes

- Soma de Matrizes

$$A = \begin{pmatrix} 3 & 4 & 8 \\ 5 & 6 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 5 & 2 & 1 \\ 3 & 3 & 1 \end{pmatrix} \qquad C = A + B = \begin{pmatrix} 8 & 6 & 9 \\ 8 & 9 & 7 \end{pmatrix}$$

```
INTEGER, DIMENSION(2,3) :: A
INTEGER, DIMENSION(2,3) :: B
INTEGER, DIMENSION(2,3) :: C
A = reshape((/3,5,4,6,8,6/),shape=(/2,3/))
B = reshape((/5,3,2,3,1,1/),shape(B))
C = A + B
END
```

- A matriz resultante é produzida a partir da soma das matrizes, somando elemento a elemento

Operações com Matrizes

- Soma de um escalar por uma matriz

→ Um escalar (posto 0) é conformável com qualquer matriz

$$A = \begin{pmatrix} 3 & 4 & 8 \\ 5 & 6 & 6 \end{pmatrix} \quad 5 = \begin{pmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{pmatrix} \quad B = A + 5 = \begin{pmatrix} 8 & 9 & 13 \\ 10 & 11 & 11 \end{pmatrix}$$

```
INTEGER, DIMENSION(2,3) :: A
INTEGER, DIMENSION(2,3) :: B
A = reshape(source=(/3,5,4,6,8,6/),shape(/2,3/))
B = A + 5
print*,B
END
```

→ O escalar é distribuído em uma matriz, conformável com a matriz com que está sendo somado

Operações com Matrizes

- Multiplicação de Matrizes

$$A = \begin{pmatrix} 3 & 4 & 8 \\ 5 & 6 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 5 & 2 & 1 \\ 3 & 3 & 1 \end{pmatrix} \qquad C = A * B = \begin{pmatrix} 15 & 8 & 8 \\ 15 & 18 & 6 \end{pmatrix}$$

```
INTEGER, DIMENSION(2,3) :: A
INTEGER, DIMENSION(2,3) :: B
INTEGER, DIMENSION(2,3) :: C
A = reshape((/3,5,4,6,8,6/),shape=(/2,3/))
B = reshape((/5,3,2,3,1,1/),shape(B))
C = A * B
print*,C
END
```

- A matriz resultante é produzida a partir da multiplicação das matrizes, multiplicando elemento a elemento

Operações com Matrizes

- Divisão de Matrizes

$$A = \begin{pmatrix} 3 & 4 & 8 \\ 5 & 6 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 5 & 2 & 1 \\ 3 & 3 & 1 \end{pmatrix} \qquad C = A / B = \begin{pmatrix} 3/5 & 2 & 8 \\ 5/3 & 2 & 6 \end{pmatrix}$$

```
INTEGER, DIMENSION(2,3) :: A
INTEGER, DIMENSION(2,3) :: B
INTEGER, DIMENSION(2,3) :: C
A = reshape((/3,5,4,6,8,6/),shape=(/2,3/))
B = reshape((/5,3,2,3,1,1/),shape(B))
C = A / B
print*,C
END
```

- A matriz resultante é produzida a partir da divisão das matrizes, dividindo elemento a elemento

Funções Intrínsecas para Matrizes

- Maior dos elementos de uma matriz A: MAXVAL(A)
- Menor dos elementos de uma matriz A: MINVAL(A)

```
program max_min_valores
  implicit none
  integer, parameter :: n = 4
  integer, dimension(n) :: dados

  print*,"digite 4 inteiros quaisquer:"
  read*,dados(1),dados(2),dados(3),dados(4)
  print*,"dados --->",dados
  print*,"maior valor =",MAXVAL(dados)
  print*,"menor valor =",MINVAL(dados)

end program max_min_valores
```

Funções Intrínsecas para Matrizes

- Soma dos elementos de uma matriz A: SUM(A)
- Produto dos elementos de uma matriz A: PRODUCT(A)

```
program soma_produto
  implicit none
  integer, parameter :: n = 4
  integer, dimension(n) :: dados

  print*,"digite 4 inteiros quaisquer:"
  read*,dados(1),dados(2),dados(3),dados(4)
  print*,"dados -->",dados
  print*," soma dos elementos =",SUM(dados)
  print*," produto dos elementos =",PRODUCT(dados)

end program soma_produto
```

Funções Intrínsecas para Matrizes

- Transposta de uma matriz A de posto 2: TRANSPOSE(A)

```
program transposta_matriz
  implicit none
  real, dimension(3,2) :: A

  print*,"Digite os elementos reais de uma matriz A(3,2):"
  read*,A(1,1), A(2,1), A(3,1), A(1,2), A(2,2), A(3,2)
  print*,'Matriz A ->',A
  print*,'Transposta da matriz A ->', TRANSPOSE(A)

end program transposta_matriz
```

→ Se a matriz é da forma DIMENSION(M,N) a transposta tem forma DIMENSION(N,M)

Entrada e Saída de dados

O Fortran 90 oferece algumas formas de entrada (leitura) e saída (escrita) de dados:

Entrada de Dados

- Os dados podem ser inseridos através do teclado
- Os dados podem ser inseridos através de arquivos

Saída de Dados

- Os dados podem ser disponibilizados através do monitor
- Os dados podem ser disponibilizados através de arquivos

Nos dois casos, os dados podem estar não-formatados ou formatados

Entrada de Dados

- Usando read*, ou read(*,*) através de um arquivo de dados

```
real :: x0, y0, v0,theta, t
read*,x0
read*,y0
read*,v0
read(*,*)theta
read(*,*)t
END
```

- Crie um arquivo de dados, com qualquer nome (input.dat por exemplo), com a mesma sequência e tipo de dados a serem inseridos pelos comandos read como descrito no código

```
1.0          ! posicao x inicial
0.0          ! posicao y inicial
100.0        ! velocidade inicial
45.0
14.0
```

Entrada de Dados

- Usando read*, ou read(*,*) através de um arquivo de dados

```
real :: x0, y0, v0,theta, t
read*,x0
read*,y0
read*,v0
read(*,*)theta
read(*,*)t
END
```

- Crie um arquivo de dados, com qualquer nome (input.dat por exemplo), com a mesma sequência e tipo de dados a serem inseridos pelos comandos read como descrito no código
- Execute o código (por exemplo o executável **a.out**) usando o comando

./a.out < input.dat

o símbolo **<** implica que os dados listados no arquivo input.dat serão lidos pelo código na sequência dos comandos read do programa

Saída de Dados

- Usando print*, ou write(*,*) através de um arquivo de dados

```
real :: x0, y0, v0,theta, t
read*,x0
read*,y0
read*,v0
read(*,*)theta
read(*,*)t
print*,'x = ',x0
print*,'y = ',y0
print*,v0
write(*,*)'angulo = ',theta
print*,t
END
```

- Execute o código (por exemplo o executável **a.out**) usando o comando

./a.out < input.dat > dados.dat

o símbolo **>** implica que a saída dos comandos print* e write(*,*) será direcionada para o arquivo dados.dat

Saída de Dados

- Usando print*, ou write(*,*) através de um arquivo de dados

```
real :: x0, y0, v0,theta, t
read*,x0
read*,y0
read*,v0
read(*,*)theta
read(*,*)t
print*,'x =',x0
print*,'y =',y0
print*,v0
write(*,*)'angulo =',theta
print*,t
END
```

- A saída do código (não-formatada) será salva no arquivo dados.dat como

```
 x =    1.00000000
 y =    0.00000000
   100.000000
 angulo =    45.0000000
   14.0000000
```

Saída de Dados

- Usando print*, ou write(*,*) através de um arquivo de dados

```
real :: x0, y0, v0,theta, t
read*,x0
read*,y0
read*,v0
read(*,*)theta
read(*,*)t
print '(A,F8.3)','x = ',x0
print '("y = ",F5.1)',y0
print '("v = ",F6.2)',v0
write(*,fmt='("angulo=",F5.1)')theta
write(*,'(A,F7.3)')" t = ",t
END
```

- A saída do código (formatada) será salva no arquivo dados.dat como

```
x =   100.000
y =   0.0
v = 100.00
angulo= 45.0
 t =   14.000
```

Formatação de dados de entrada e saída

- A formatação é feita através dos descritores de edição
- Cada tipo intrínseco (REAL, INTEGER, CHARACTER, LOGICAL, COMPLEX) tem um formato de descritor de dados

Tipo de dado	Descritores de dado
Inteiro	`I<w>[.<m>], B<w>[.<m>], O<w>[.<m>], Z<w>[.<m>], G<w>.<d>[E<e>]`
Real e Complexo	`F<w>.<d>, E<w>.<d>[E<e>], EN<w>.<d>[E<e>], ES<w>.<d>[E<e>], D<w>.<d>, G<w>.<d>[E<e>]`
Lógico	`L<w>, G<w>.<d>[E<e>]`
Caractere	`A[<w>], G<w>.<d>[E<e>]`

Formatação de dados de entrada e saída

- Tipo INTEGER: **<r>I<w>**

  <r> : contador de repetição

  <w> : número total de dígitos no campo

1. Usado na representação de dados inteiros
2. O sinal negativo é contado como uma posição no campo <w>
3. O sinal positivo não é contado como uma posição no campo <w>

$$\underbrace{\text{I1 : } 5}_{\text{1 caracteres}} \qquad \underbrace{\text{I4 : } 1000}_{\text{4 caracteres}} \qquad \underbrace{\text{I8 : } -1000000}_{\text{8 caracteres}}$$

Formatação de dados de entrada e saída

- Tipo REAL: **<r>F<w>.<d>**

<r> : contador de repetição
<w> : número total de dígitos no campo
<d> : número de dígitos a direita do ponto decimal

$$w \geq d+2$$

1. Usado na representação em REAL sem expoente
2. O ponto decimal conta como uma posição no campo <w>
3. O sinal negativo é contado como uma posição no campo <w>
4. O sinal positivo não é contado como uma posição no campo <w>
5. Se parte decimal tem mais dígitos que o campo <d> é feito o arredondamento

```
          5 dígitos                 7 dígitos                 9 dígitos
F7.5:    0.14286        F9.7:    -.1414214      F12.9:   -3.141592654
         7 caracteres              9 caracteres             12 caracteres
```

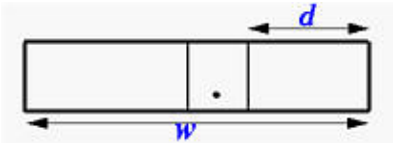
Formatação de dados de entrada e saída

- Tipo REAL: **<r>F<w>.<d>**

<r> : contador de repetição
<w> : número total de dígitos no campo
<d> : número de dígitos a direita do ponto decimal

$w \geq d+2$

Fw.d



REAL :: a = 123.345, b = -123.345

		1	2	3	4	5	6	7	8	9	10
1	WRITE(*,"(F10.0)") a							1	2	3	.
2	WRITE(*,"(F10.1)") a						1	2	3	.	3
3	WRITE(*,"(F10.2)") a					1	2	3	.	3	5
4	WRITE(*,"(F10.3)") a				1	2	3	.	3	4	5
5	WRITE(*,"(F10.4)") a			1	2	3	.	3	4	5	0
6	WRITE(*,"(F10.5)") a		1	2	3	.	3	4	5	0	0
7	WRITE(*,"(F10.6)") a	1	2	3	.	3	4	5	0	0	0
8	WRITE(*,"(F10.7)") a	*	*	*	*	*	*	*	*	*	*
9	WRITE(*,"(F10.4)") b		-	1	2	3	.	3	4	5	0
10	WRITE(*,"(F10.5)") b	-	1	2	3	.	3	4	5	0	0
11	WRITE(*,"(F10.6)") b	*	*	*	*	*	*	*	*	*	*

Formatação de dados de entrada e saída

- Tipo REAL: **<r>E<w>.<d>**

<r> : contador de repetição
<w> : número total de dígitos no campo
<d> : número de dígitos a direita do ponto decimal

$$w \geq d+7$$

1. Usado na representação em REAL com exponente
2. O ponto decimal conta como uma posição no campo <w>
3. O sinal negativo é contado como uma posição no campo <w>
4. O sinal positivo não é considerado como uma posição no campo <w>
5. A letra E do expoente é contada como uma posição no campo <w>
6. O sinal do expoente (+ ou -) é contado como uma posição no campo <w>
7. O maior expoente é 99. Acima deste valor a letra E é omitida na saída

E11.5 : $\underbrace{0.\ \overbrace{31416}^{\text{5 dígitos}}\ \text{E}+01}_{\text{11 caracteres}}$

E14.7 : $\underbrace{-0.\ \overbrace{4430949}^{\text{7 dígitos}}\ \text{E}-11}_{\text{14 caracteres}}$

Formatação de dados de entrada e saída

- Tipo REAL: **<r>E<w>.<d>E<e>**

<r> : contador de repetição
<w> : número total de dígitos no campo
<d> : número de dígitos a direita do ponto decimal
<e> : número de dígitos do expoente (excluído o sinal + ou -)

$$w \geq d+e+5$$

1. Usado na representação em REAL com exponente
2. O ponto decimal conta como uma posição no campo <w>
3. O sinal negativo é contado como uma posição no campo <w>
4. O sinal positivo não é considerado como uma posição no campo <w>
5. A letra E do expoente é contada como uma posição no campo <w>
6. O sinal do expoente (+ ou -) é contado como uma posição no campo <w>
7. Forma obrigatória para expoentes maiores que 999.

E10.5E1 : $\underbrace{0.\overbrace{31416}^{\text{5 dígitos}}\text{E}+1}_{\text{10 caracteres}}$ **E11.5E1** : $\underbrace{-0.\overbrace{31416}^{\text{5 dígitos}}\text{E}-2}_{\text{11 caracteres}}$ **E10.5E1** : $\underbrace{-.\overbrace{31416}^{\text{5 dígitos}}\text{E}-2}_{\text{10 caracteres}}$

Formatação de dados de entrada e saída

- Tipo REAL: **<r>E<w>.<d>[E<e>]**

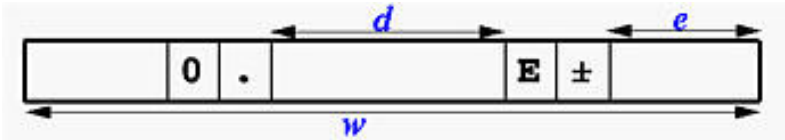
<r> : contador de repetição
<w> : número total de dígitos no campo
<d> : número de dígitos a direita do ponto decimal

<e> : número de dígitos do expoente (excluído o sinal + ou -)

Ew.d

w ≥ d+7

Ew.dEe



w ≥ d+e+5

Formatação de dados de entrada e saída

- Tipo REAL: **<r>ES<w>.<d>[>E<e>]**

  <r> : contador de repetição
  <w> : número total de dígitos no campo
  <d> : número de dígitos a direita do ponto decimal

  <e> : número de dígitos do expoente (excluído o sinal + ou -)

1. Usado na representação REAL em notação científica
2. Mesmas regras do descritor E
3. Na saída o valor absoluto da parte inteira é maior ou igual a 1 e menor que 10

ES11.5 : $\underbrace{3.\ \overbrace{14160}^{\text{5 dígitos}}\ \text{E}+00}_{\text{11 caracteres}}$ **ES11.4** : $\underbrace{-3.\ \overbrace{1416}^{\text{4 dígitos}}\ \text{E}-03}_{\text{11 caracteres}}$ **ES10.4E1** : $\underbrace{-3.\ \overbrace{1416}^{\text{4 dígitos}}\ \text{E}-3}_{\text{10 caracteres}}$

Formatação de dados de entrada e saída

- Tipo COMPLEX:

1. As regras de formatação são as mesmas do tipo REAL
2. As partes real e imaginária do dado complexo são formatadas de forma independente
3. Os descritores das partes real e imaginária não precisam ser iguais

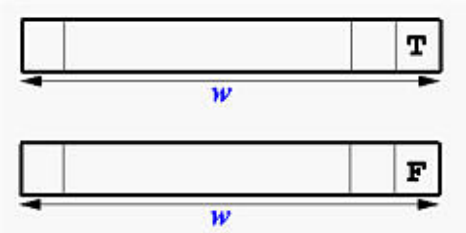
Formatação de dados de entrada e saída

- Tipo LOGICAL: **<r>L<w>**

  <r> : contador de repetição
  <w> : número total de dígitos no campo

1. Usado na representação de dados lógicos
2. Na saída um dos caracteres T ou F será apresentado
3. O campo <w> pode ser omitido, usando apenas L
4. Quando usado L7 as strings .true. e .false. podem ser lidas na entrada

Lw




```
LOGICAL :: a = .TRUE., b = .FALSE.
1 WRITE(*,"(L1,L2)") a, b        T  F
2 WRITE(*,"(L3,L4)") a, b           T     F
                      1  2  3  4  5  6  7
```

Formatação de dados de entrada e saída

- Tipo CHARACTER: **<r>A<w>**

  <r> : contador de repetição
  <w> : número total de dígitos no campo

1. Usado na representação de caracteres
2. O campo <w> pode ser omitido, usando apenas A. Neste caso, a largura do campo é determinada pelo tamanho real do dado de entrada ou saída
3. Usando o campo <w> podemos determinar o tamanho desejado da string lida ou escrita

Descritor	Registro escrito
A	TEMPORARIO
A11	bTEMPORARIO
A8	TEMPORAR

Formatação de dados de entrada e saída

- **Descritores de controle de edição**

→ Um descritor de controle de edição age de duas formas: determinando como o texto é organizado (especialmente na saída) ou afetando as conversões realizadas por descritores de edição de dados subsequentes

Tipo de controle	Descritores de controle
Posicional	`T<n>`, `TL<n>`, `TR<n>`, `<n>X`
Sinal	`S`, `SP`, `SS`
Interpretação de brancos	`BN`, `BZ`
Fator de escala	`<k>P`
Miscelâneo	`:`, `/`

Formatação de dados de entrada e saída

- Descritor geral **<r>G<w>.<d>[E<e>]**

1. Usado para qualquer tipo intrínseco de dados.
2. Útil para a saída de valores cujas magnitudes não são bem conhecidas e quando o uso do descritor F é preferível ao descritor E
3. Quando a saída é do tipo REAL ou COMPLEX a ação do descritor geral é idêntica à do descritor E<w>.d[E<e>], exceto que na saída de dados, quando a magnitude (N) do valor está no intervalo

$$0.1 - 0.5 \times 10^{-<d>-1} \leq N < 10^{<d>} - 0.5$$

   ou zero quando <d>=0, são convertidos como com o descritor F, seguidos pelo mesmo número de espaços em branco que o descritor E teria à parte exponencial.
4. Quando usado para tipos INTEGER, LOGICAL e CHARACTER, as regras são as mesmas dos respectivos descritores de edição

Formatação de dados de entrada e saída

```
INTEGER :: A, B, C

A = 100
B = -200
C = 10

print '(I3)',A
print '(2(I4))',B,C
write (*,'(3(I5))')A,B,C
END
```

A saída será apresentada como

```
100
-200  10
  100 -200   10
```

Formatação de dados de entrada e saída

```
real :: x0, y0, v0,theta, t

x0 = 100.0; y0 = 0.0; v0 = 100.0; theta = 45.0; t = 14.0

print '(A,F8.3)','x = ',x0
print '("x = ",F8.3,10X,"y = ",F5.1)',x0,y0
print '("v = ",F6.2)',v0
write(*,fmt='("angulo=",F5.1,/)')theta
write(*,'(A,F7.3)')" t = ",t
END
```

A saída será apresentada como

```
x =   100.000
x = 100.000              y =    0.0
v = 100.00
angulo= 45.0

 t =   14.000
```

Formatação de dados de entrada e saída

```
real :: x0, y0, v0,theta, t

x0 = 100.0; y0 = 0.0; v0 = 100.0; theta = 45.0; t = 14.0

print '(A,E8.3)','x = ',x0
print '("x = ",E8.3,10X,"y = ",E5.1)',x0,y0
print '("v = ",E6.2)',v0
write(*,fmt='("angulo=",E5.1,/)')theta
write(*,'(A,E7.3)')"t = ",t
END
```

A saída na forma exponencial será apresentada com **estouro de campo (****)**

```
x = .100E+03
x =.100E+03                y = *****
v = ******
angulo=*****

 t = *******
```

Formatação de dados de entrada e saída

```
real :: x0, y0, v0,theta, t

x0 = 100.0; y0 = 0.0; v0 = 100.0; theta = 45.0; t = 14.0

print '(A,E8.3)','x = ',x0
print '("x =",E8.3,10X,"y = ",E9.4)',x0,y0
print '("v = ",E9.4)',v0
write(*,fmt='("angulo=",E7.2,/)')theta
write(*,'(A,E7.2)')" t = ",t
END
```

A saída na forma exponencial será apresentada SEM estouro de campo

```
x = .100E+03
x =.100E+03                y = .0000E+00
v = .1000E+03
angulo=.45E+02

 t = .14E+02
```

Formatação de dados de entrada e saída

```
real :: x0, y0, v0,theta, t

x0 = 100.0; y0 = 0.0; v0 = 100.0; theta = 45.0; t = 14.0

print '(A,E10.3)','x = ',x0
print '("x = ",E10.3,10X,"y = ",E11.4)',x0,y0
print '("v = ",E11.4)',v0
write(*,fmt='("angulo=",E9.2,/)')theta
write(*,'(A,E11.4)')" t = ",t
END
```

A saída na forma exponencial será apresentada com zero à esquerda

```
x =  0.100E+03
x = 0.100E+03          y =  0.0000E+00
v =  0.1000E+03
angulo= 0.45E+02

 t =  0.1400E+02
```

Formatação de dados de entrada e saída

```
real :: x0, y0, v0,theta, t

x0 = 100.0; y0 = 0.0; v0 = 100.0; theta = 45.0; t = 14.0

print '(A,E10.3)','x = ',x0
print '("x = ",E10.3,10X,"y = ",E10.4E1)',x0,y0
print '("v = ",E11.4)',v0
write(*,fmt='("angulo=",E9.2,/)')theta
write(*,'(A,E11.4)')"t = ",t
END
```

A saída na forma exponencial E<w>.<d>E<e> será apresentada como

```
x =   0.100E+03
x = 0.100E+03              y =   0.0000E+0
v =   0.1000E+03
angulo= 0.45E+02

 t =   0.1400E+02
```

Formatação de dados de entrada e saída

```
real :: x0, y0, v0,theta, t

x0 = 100.0; y0 = 0.0; v0 = 100.0; theta = 45.0; t = 14.0

print '(A,G10.3)','x = ',x0
print '("x = ",G10.3,10X,"y = ",G10.4E1)',x0,y0
print '("v = ",G11.4)',v0
write(*,fmt='("angulo=",G9.2,/)')theta
write(*,'(A,G11.4)')" t = ",t
END
```

A saída na forma geral G<w>.<d>[E<e>] será apresentada como

```
x =    100.
x =   100.                    y =    0.000
v =    100.0
angulo=   45.

 t =    14.00
```