

# Programação Estruturada

## Estruturas e Funções

**Prof. Edson Alves**

Faculdade UnB Gama

2020

# Sumário

1. Estruturas de controle
2. Tipos de dados compostos
3. Procedimentos e Funções
4. Módulos
5. Programação Procedural

## Estruturas de seleção

- ▶ Em Fortran, a principal estrutura de seleção é o construto IF-THEN-ELSE, cuja sintaxe é

```
if (condicao) then
    blocoA
else
    blocoB
end if
```

- ▶ A condicao é uma variável ou expressão lógica
- ▶ Se a condicao for **verdadeira**, o blocoA é executado, e ao fim deste a execução segue para a código que segue o **end if**
- ▶ Caso contrário, o blocoB é executado
- ▶ A cláusula **else** é opcional
- ▶ Se um **if** segue imediatamente um **else**, é criada uma cascata de blocos mutuamente excludentes, sendo executado o primeiro cuja condição associada for verdadeira (ou o último, caso exista uma cláusula **else** final)

## Exemplo de uso do construto IF-ELSE

```
1 ! Calcula o imposto de renda mensal
2 program IRRF
3
4     implicit none
5
6     real :: salario, aliquota, deducacao, imposto
7
8     write(*,*) 'Insira o salário mensal: '
9     read(*,*) salario
10
11     ! Determinar a aliquota e a dedução a partir do salário
12     if (salario <= 1903.98) then
13         aliquota = 0
14     else if (salario <= 2826.65) then
15         aliquota = 0.075
16         deducacao = 142.80
17     else if (salario <= 3751.05) then
18         aliquota = 0.15
19         deducacao = 354.80
20     else if (salario <= 4664.68) then
21         aliquota = 0.225
22         deducacao = 636.13
```

## Exemplo de uso do construto IF-ELSE

```
23     else
24         aliquota = 0.275
25         deducacao = 869.36
26     end if
27
28     ! Imprime o imposto a ser pago
29     if (aliquota == 0) then
30         write(*,*) 'Isento'
31     else
32         imposto = salario * aliquota - deducacao
33         write(*,1) imposto
34     end if
35
36 1   format ('Imposto devido: ', F9.2)
37
38 end program IRRF
```

## SELECT-CASE

- ▶ Outra estrutura de seleção disponível em Fortran é o construto SELECT-CASE, cuja sintaxe é

```
select case (seletor)
  case (lista de rótulos 1)
    bloco1
  case (lista de rótulos 2)
    bloco2
  ...
  case (lista de rótulos N)
    blocoN
  case default
    bloco_padrao
end select
```

- ▶ O seletor é uma variável ou expressão cujo tipo é **integer**, **character** ou **logical**
- ▶ As listas de rótulos descrevem os rótulos que compõem cada caso, separados por vírgulas

## SELECT-CASE

- ▶ Os rótulos podem ser especificados de quatro maneiras:

```
x  
a : b  
L :  
: R
```

- ▶ Na primeira forma, um único valor `x` é especificado
- ▶ Na segunda forma, são especificados todos os valores no intervalo `[a, b]` (aqui, `a` deve ser necessariamente menor do que `b`)
- ▶ Na terceira forma, são especificados todos os valores maiores ou iguais a `L`; na quarta, todos os valores menores ou iguais a `R`
- ▶ Uma vez determinado o valor do seletor, será executado o primeiro bloco cujo valor está relacionado na lista de rótulos, e em seguida a execução segue para a linha que sucede o **end select**
- ▶ O **case default** é opcional, e seu bloco será executado apenas se o valor do seletor não estiver listado em nenhum **case**

## Exemplo de uso do construto SELECT-CASE

```
1 ! Determina a prioridade de atendimento do paciente, de acordo com a idade
2 program priority
3
4     implicit none
5
6     integer :: idade
7     character (len = 6) :: prioridade
8
9     write(*,*) 'Insira a idade do paciente: '
10    read(*,*) idade
11
12    select case (idade)
13        case (0 : 6)
14            prioridade = 'media'
15        case (65 : )
16            prioridade = 'maxima'
17        case (7 : 64)
18            prioridade = 'minima'
19        case default
20            write(*,*) 'Idade inválida!'
21            return
22    end select
```



## Exemplo de uso do construto SELECT-CASE

```
23
24     ! Imprime a prioridade do paciente
25     write(*,1) idade, prioridade
26
27 1   format (I3, ' anos, prioridade: ', A6)
28
29 end program priority
```

# Estruturas de laço

- ▶ Fortran disponibiliza duas estruturas de repetição
- ▶ A primeira delas é a estrutura DO, cuja sintaxe é

```
do variavel = a, b [, delta]
    bloco
end do
```
- ▶ A variavel de controle deve ser do tipo **integer**
- ▶ A variavel terá a como valor inicial e b como valor final
- ▶ Após cada execução do bloco, o valor da variável é acrescido do delta
- ▶ Se o delta (passo) for omitido, ele assume o valor 1 (um)
- ▶ O comando **exit**, se executado, encerra o laço imediatamente
- ▶ Já o comando **cycle** finaliza a execução do bloco, seguindo imediatamente para a atualização da variavel

## Exemplo de uso do construto DO

```
1 ! Computa o fatorial de n
2 program priority
3
4     implicit none
5
6     integer :: n, i, factorial = 1
7
8     read(*,*) n
9
10    do i = 1, n
11        factorial = factorial * i
12    end do
13
14    write(*,1) n, factorial
15
16 1    format ('Fatorial de ', I2, I10)
17
18 end program priority
```

## DO-WHILE

- ▶ Fortran possui uma segunda estrutura de repetição: o construto DO-WHILE, cuja sintaxe é

```
do while (condicao)
    bloco
end do
```

- ▶ A condicao é uma variável ou uma expressão do tipo **logical**
- ▶ Se a condicao for verdadeira, o bloco associado será executado
- ▶ Após a execução do bloco, a condição é reavaliada e, se permanecer verdadeira, o bloco é executado novamente
- ▶ Se o bloco não modifica as variáveis que compõem a condição de modo que ela possa eventualmente se tornar falsa, o laço será infinito
- ▶ Os comandos **exit** e **cycle** também podem ser usados neste construto, com o mesmo significado do construto DO

## Exemplo de uso do construto DO-WHILE

```
1 ! Computa a^n com complexidade O(log n)
2 program fast_exp
3
4     implicit none
5
6     integer(16) :: a, n, res = 1, base      ! Inteiros de 128-bits
7     read(*,*) a, n
8
9     base = a
10
11     do while (n > 0)
12         if (iand(n, 1) > 0) then            ! iand(x, y) = x & y
13             res = res * base
14         end if
15
16         base = base * base
17         n = ishft(n, -1)                    ! n = n >> 1
18     end do
19
20     write(*,*) res
21
22 end program fast_exp
```

# Vetores

- ▶ Fortran tem suporte nativo para **vetores** (*arrays*) de elementos de um mesmo tipo
- ▶ A sintaxe para a declaração de um vetor é

```
tipo_de_dado :: nome(dim1, dim2, ..., dimN)
```
- ▶ O parêntesis que segue o nome da variável, e as dimensões listadas, determinam a **forma** (*shape*) do vetor
- ▶ A notação de parêntesis também pode ser utilizada para acessar os elementos individuais do vetor
- ▶ Fortran utilizar a indexação matemática, de modo que o primeiro elemento do vetor tem índice 1
- ▶ A palavra-chave **allocatable** pode ser utilizada para declarar vetores dinâmicos
- ▶ A função **allocate**() reserva espaço em memória para tais vetores, e esta memória deve ser liberada após o uso por meio da função **deallocate**()

# Exemplo de uso de vetores

```
1 ! Computa a média e o desvio padrão dos elementos do vetor xs
2 program statistics
3
4     implicit none
5
6     integer, allocatable :: xs(:)    ! Vetor dinâmico
7     integer :: n, i                  ! n = dimensão de xs
8     real :: stats(2)                 ! Vetor com duas posições
9
10    write(*,*) 'Insira o número de entradas: '
11    read(*,*) n
12
13    if (n < 1) then
14        return
15    end if
16
17    allocate(xs(n))
18
19    do i = 1, n
20        write(*,*) 'Insira a entrada ', i, ': '
21        read(*,*) xs(i)
22    end do
```

# Exemplo de uso de vetores

```
24 stats(1) = 0.0 ! Média
25
26 do i = 1, n
27     stats(1) = stats(1) + xs(i)
28 end do
29
30 stats(1) = stats(1) / n
31
32 stats(2) = 0.0 ! Desvio-padrão
33
34 do i = 1, n
35     stats(2) = stats(2) + (xs(i) - stats(1)) ** 2
36 end do
37
38 write(*,*) 'Média = ', stats(1)
39
40 stats(2) = sqrt(stats(2)/n)
41
42 write(*,*) 'Desvio = ', stats(2)
43
44 deallocate(xs)
45
46 end program statistics
```



# Manipulação de vetores

- ▶ Fortran disponibiliza uma série de características úteis para a manipulação de vetores
- ▶ Por exemplo, se  $a$ ,  $b$  e  $c$  são vetores de mesma dimensão ( $N$ ), a expressão " $c = a + b$ " equivale a

```
do i = 1, N
    c(i) = a(i) + b(i)
end do
```

- ▶ A atribuição " $xs = k$ " atribui o valor  $k$  a todos os elementos do vetor  $xs$
- ▶ A atribuição também pode ser utilizada para copiar vetores de mesma dimensão
- ▶ Além disso, há várias funções intrínsecas que manipulam vetores diretamente, como `dot_product`, `matmul`, `maxval`, `minval`, `product` e `sum`

## Exemplo de manipulação de vetores

```
1 ! Calcula o ângulo entre dois vetores
2 program angle
3
4     real, parameter :: pi = acos(-1.0)
5     real :: theta, xlen, ylen
6     real :: xs(2) = (/ 1, 0 /), ys(2) = 1    ! ys = (1, 1)
7     real :: A(2, 2) = reshape((/ 0, 1, -1, 0 /), (/ 2, 2 /))
8
9     xlen = sqrt(dot_product(xs, xs))
10    ylen = sqrt(dot_product(ys, ys))
11    theta = acos(dot_product(xs, ys) / (xlen * ylen))
12    theta = theta * 180 / pi
13
14    write(*,*) 'Ângulo, em graus: ', theta
15
16    ys = matmul(A, ys)
17    theta = acos(dot_product(xs, ys) / (xlen * ylen))
18    theta = theta * 180 / pi
19
20    write(*,*) 'Ângulo após rotação, em graus: ', theta
21
22 end program angle
```

# Referências

1. **CHEUNG**, Shun Yan. [Loops \(DO, DO WHILE, EXIT, CYCLE\)](#), acesso em 04/02/2020.
2. GNU Fortran. [IAND – Bitwise logical and](#), acesso em 04/02/2020.
3. GNU Fortran. [ISHFT – Shift bits](#), acesso em 04/02/2020.
4. **PADMAN**, Rachael. [Computer Physics: Self-study guide 2 – Programming in Fortran 95](#), University of Cambridge, Department of Physics, 2007.
5. **SHALOM**, Elad. *A Review of Programming Paradigms Throughout the History – With a Suggestion Toward a Future Approach*, Amazon, 2019.
6. **SHENE**, C. K. [SELECT CASE Statement](#), acesso em 04/02/2020.