Introdução ao Fortran 90 - 3

Alexandre Diehl

Departamento de Física – UFPel

Controles de Fluxo de Execução

Definição

Estruturas (ou blocos) de programação que permitem controlar o fluxo de execução de um programa, além da construção sequencial, incluindo decisão, seleção e repetição.

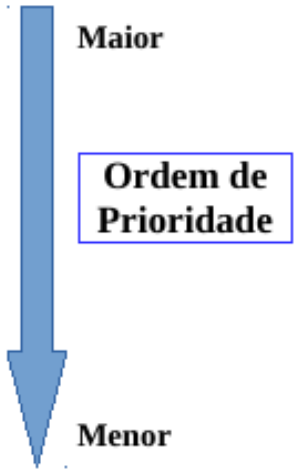
Tipos de Controles de Fluxo

- Estrutura de Decisão do tipo IF
- Estrutura de Seleção do tipo CASE
- Estrutura de Repetição do tipo DO

Definição

A estrutura de decisão IF fornece um mecanismo para controle de desvio de fluxo, dependendo de uma condição relacional e/ou lógica.

Tipo	Operador	Associatividade
Aritmético	**	Direita para Esquerda
	* /	Esquerda para Direita
	+ -	Esquerda para Direita
Relacional	< <= > >= == /=	Nenhuma
Lógico	.NOT.	Direita para Esquerda
	.AND.	Esquerda para Direita
	.OR.	Esquerda para Direita
	.EQV. .NEQV.	Esquerda para Direita



Caso 1: Comando IF

IF (<expressão relacional e/ou lógica>) <comando executável>

→ Se o valor da <expressão relacional e/ou lógica> for verdadeira, um único comando executável é realizado

```
LOGICAL :: flag = .true., noflag = .false.
REAL :: x = 1.0, Y = -0.5

IF (flag) print*,' flag =',flag
IF (X - Y > 0.0) X = 0.0
IF ((X - Y > 0.0) .AND. (.NOT. noflag)) print*,' oi'

END
```

→ Se a condição relacional e/ou lógica não for verdadeira, o comando na sequência da estrutura IF é executado.

Caso 2: Construto IF ... END IF

IF (<expressão relacional e/ou lógica>) THEN
    <bloco de comandos>
END IF

→ Se o valor da <expressão relacional e/ou lógica> for verdadeira, um bloco de comandos (mais de um comando) ou apenas um comando é executado.

```
REAL :: x = 1.0, Y = -0.5

IF (X > Y) THEN
   print*,' X maior do que Y'
   STOP
END IF
END
```

→ Se a condição relacional e/ou lógica não for verdadeira, o comando na sequência da estrutura IF ... END IF é executado.

Caso 3: Construto IF ... ELSE ... END IF

IF (<expressão relacional e/ou lógica>) THEN
    <bloco de comandos 1>
ELSE
    <bloco de comandos 2>
END IF

→ Se o valor da <expressão relacional e/ou lógica> for verdadeira, o bloco de comandos 1 é executado; se for falso, o bloco de comandos 2 é executado (sem qualquer teste).

```
REAL :: x = 1.0, Y = -0.5
IF (X > Y) THEN
   print*,' X maior do que Y'
   STOP
ELSE
   print*,' X menor do que Y'
   STOP
END IF
END
```

Caso 4: Construto IF ... ELSE IF ... END IF

IF (<expressão relacional e/ou lógica 1>) THEN
    <bloco de comandos 1>
ELSE IF (<expressão relacional e/ou lógica 2>) THEN
    <bloco de comandos 2>
END IF

→ Se o valor da <expressão relacional e/ou lógica 1> for verdadeira, o bloco de comandos 1 é executado; se for falso, a <expressão relacional e/ou lógica 2> é testada para que o bloco de comandos 2 seja executado em caso verdadeiro.

```
real :: x = 1.0
if (X < 0.0) then
  print*,'␣X␣menor␣do␣que␣zero'; stop
else if (X >= 0.0) then
  print*,'␣X␣igual␣ou␣maior␣que␣zero'; stop
end if
end
```

Caso 5: Construto IF ... ELSE IF ... ELSE ... END IF

IF (<expressão relacional e/ou lógica 1>) THEN
    <bloco de comandos 1>
ELSE IF (<expressão relacional e/ou lógica 2>) THEN
    <bloco de comandos 2>
ELSE
    <bloco de comandos 3>
END IF

→ Se o valor da <expressão relacional e/ou lógica 1> for verdadeira, o bloco de comandos 1 é executado; se for falso, a <expressão relacional e/ou lógica 2> é testada para que o bloco de comandos 2 seja executado em caso verdadeiro; se for falsa o bloco de comandos 3 é executado (sem teste).

Caso 5: Construto IF ... ELSE IF ... ELSE ... END IF

IF (<expressão relacional e/ou lógica 1>) THEN
    <bloco de comandos 1>
ELSE IF (<expressão relacional e/ou lógica 2>) THEN
    <bloco de comandos 2>
ELSE
    <bloco de comandos 3>
END IF

```
real :: x = 1.0
if (X < 0.0) then
  print*,'␣X␣menor␣do␣que␣Y';  stop
else if (X == 0.0) then
  print*,'␣X␣igual␣a␣zero';  stop
else
  print*,'␣X␣maior␣que␣zero';  stop
end if
end
```

Estrutura de Seleção CASE

Definição

A estrutura de seleção CASE fornece um mecanismo para controle de desvio de fluxo para diversas opções, a partir do teste de uma única expressão, do tipo inteiro, lógico ou de carateres (não pode ser do tipo real).

```
SELECT CASE (<expressão>)
   CASE (<seletor 1>)
      <bloco de comandos 1>
   CASE (<seletor 2>)
      <bloco de comandos 2>
   ...
   CASE (<seletor n>)
      <bloco de comandos n>
   CASE DEFAULT
      <bloco de comandos>
END SELECT
```

Formas de uso

- <expressão>: seletor da estrutura CASE. Deve ser escalar, do tipo inteiro, lógico ou caractere. Não pode ser real
- seletor: opções de seleção para a expressão, com o mesmo tipo da expressão
- CASE DEFAULT: caso selecionado se a expressão não pertencer a nenhum dos seletores.

Estrutura de Seleção CASE

Definição

A estrutura de seleção CASE fornece um mecanismo para controle de desvio de fluxo para diversas opções, a partir do teste de uma única expressão, do tipo inteiro, lógico ou de carateres (não pode ser do tipo real).

```
SELECT CASE (<expressão>)
   CASE (<seletor 1>)
      <bloco de comandos 1>
   CASE (<seletor 2>)
      <bloco de comandos 2>
   ...
   CASE (<seletor n>)
      <bloco de comandos n>
   CASE DEFAULT
      <bloco de comandos>
END SELECT
```

Formas de uso

- CASE (<inf> : <sup>): para <expressões> inteiras ou de caracteres, um intervalo pode ser especificado, separando os limites inferior <inf> e superior <sup> por dois pontos :
- CASE (: <sup>): o limite inferior pode estar ausente (mas não ambos)
- CASE (<inf> :): o limite superior pode estar ausente (mas não ambos)

Estrutura de Seleção CASE

Definição

A estrutura de seleção CASE fornece um mecanismo para controle de desvio de fluxo para diversas opções, a partir do teste de uma única expressão, do tipo inteiro, lógico ou de carateres (não pode ser do tipo real).

```
SELECT CASE (<expressão>)
   CASE (<seletor 1>)
      <bloco de comandos 1>
   CASE (<seletor 2>)
      <bloco de comandos 2>
   ...
   CASE (<seletor n>)
      <bloco de comandos n>
   CASE DEFAULT
      <bloco de comandos>
END SELECT
```

Formas de uso

- A forma geral do <seletor> é uma lista de valores e de intervalos não sobrepostos, todos do mesmo tipo que <expressão>, entre parênteses, tal como

  CASE (1, 2, 7, 10:17, 23)

Exemplo de uso

```
program testa_case
  implicit none
  character(len=1) :: nome

  print*,"Digite um caracer (1 digito):"
  read*,nome
  select case (nome)
    case ('a' : 'z')
       print*,"caracter minusculo :", nome
    case ("A" : "Z")
       print*,"caracter maiusculo :"
    case ("0" : "9")
       print*,"caracter numeral :", nome
    case default
       print*,"caracter especial :", nome
  end select
end program testa_case
```

Exemplo: conversor de temperaturas

Faça um programa em FORTRAN 90 que converta uma temperatura fornecida pelo usuário, entre as escalas Celsius (C), Fahrenheit (F) e Kelvin (K). O programa deve ter as seguintes características:

- A entrada da temperatura deve ser clara para o usuário: a identificação da escala, usando um caracter de comprimento 1, e do valor a ser convertido.
- A saída deve apresentar a temperatura fornecida nas 3 escalas.
- O uso do construto CASE.
- O uso do construto IF para evitar a conversão de valores de temperaturas não permitados (por exemplo, temperaturas negativas na escala Kelvin).

Estrutura de Seleção CASE

```
PROGRAM temperatura
   IMPLICIT NONE
   REAL :: celsius, fahrenheit, kelvin, T
   CHARACTER :: escala
   PRINT*,"Insira a temperatura:"
   READ*,T
   PRINT*,"Insira a escala desejada (C, F ou K):"
   READ*,escala

   SELECT CASE (escala)
   CASE ("c", "C")
        [...]
   CASE ("k","K")
        [...]
   CASE ("f","F")
        [...]
   CASE DEFAULT
        [...]
   END SELECT
END PROGRAM temperatura
```

Estrutura de Seleção CASE

```
PROGRAM temperatura
   [...]
   SELECT CASE (escala)
   CASE ("c", "C")
        celsius = T
        fahrenheit = (9*celsius/5.0) + 32
        kelvin = celsius + 273.15
        IF (kelvin < 0) THEN
            PRINT*,"AVISO: temperatura < 0 K"
            STOP
        END IF
        PRINT*,"A temperatura em celsius: ", celsius,"C"
        PRINT*,"A temperatura em farenheit: ", fahrenheit,"F"
        PRINT*,"A temperatura em kelvin: ", kelvin,"K"
   CASE ("k","K")
        [...]
   CASE ("f","F")
        [...]
   CASE DEFAULT
        [...]
   END SELECT
END PROGRAM temperatura
```

Estrutura de Seleção CASE

```
PROGRAM temperatura
   [...]
   SELECT CASE (escala)
   CASE ("c", "C")
        [...]
   CASE ("k","K")
        kelvin = T
        celsius = kelvin - 273.15
        fahrenheit = (9*celsius/5.0) + 32
        IF (kelvin < 0) THEN
        PRINT*,"AVISO: temperatura < 0 K"
        STOP
        END IF
        PRINT*,"A temperatura em Celsius: ", celsius,"C"
        PRINT*,"A temperatura em Farenheit: ", fahrenheit,"F"
        PRINT*,"A temperatura em Kelvin: ", kelvin,"K"
   CASE ("f","F")
        [...]
   CASE DEFAULT
        [...]
   END SELECT
END PROGRAM temperatura
```

Estrutura de Seleção CASE

```
PROGRAM temperatura
   [...]
   SELECT CASE (escala)
   CASE ("c", "C")
        [...]
   CASE ("k","K")
        [...]
   CASE ("f","F")
        fahrenheit = T
        celsius = (fahrenheit - 32)*5/9.0
        kelvin = celsius + 273.15
        IF (kelvin < 0) THEN
            PRINT*,"AVISO: temperatura < 0 K"
            STOP
        END IF
        PRINT*,"A temperatura em Celsius: ", celsius,"C"
        PRINT*,"A temperatura em Farenheit: ", fahrenheit,"F"
        PRINT*,"A temperatura em Kelvin: ", kelvin,"K"
   CASE DEFAULT
        [...]
   END SELECT
END PROGRAM temperatura
```

Estrutura de Seleção CASE

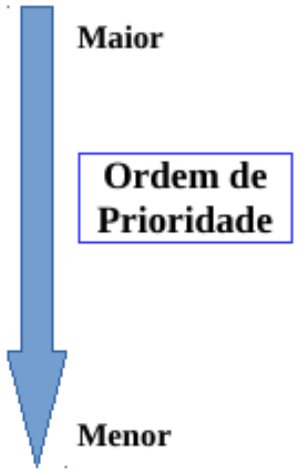
```
PROGRAM temperatura
    [...]
   SELECT CASE (escala)
   CASE ("c", "C")
        [...]
   CASE ("k","K")
        [...]
   CASE ("f","F")
        [...]
   CASE DEFAULT
        print*,"␣escala␣nao␣existente:␣digite␣C,␣F␣ou␣K"
   END SELECT
END PROGRAM temperatura
```

Estrutura de Repetição DO

Definição

A estrutura de repetição DO fornece um mecanismo para controle de desvio de fluxo, que permite que um bloco de comandos seja executado de forma repetida, através de um número finito ou infinito de repetições.

Tipo	Operador						Associatividade
Aritmético	**						Direita para Esquerda
	*		/				Esquerda para Direita
	+		-				Esquerda para Direita
Relacional	<	<=	>	>=	==	/=	Nenhuma
Lógico	.NOT.						Direita para Esquerda
	.AND.						Esquerda para Direita
	.OR.						Esquerda para Direita
	.EQV.		.NEQV.				Esquerda para Direita



Caso 1: Comando DO ... END DO finito

→ Permite que sejam feitas um número finito de iterações, onde a cada iteração os comandos listados dentro do bloco serão executados.

```
DO <controle> = <início>, <fim>, [, <passo>]
      <comando 1>
      ...
      <comando n>
END DO
```

1. A variável <controle>, do tipo INTEGER, é inicializada com o valor da expressão <início>:
   → se <controle> for um valor contido entre <inicio> e <fim>, os comandos listados dentro da estrutura serão executados.
   → caso contrário, o próximo comando imediatamente posterior ao END DO será executado.
2. Após a execução dos comandos, o <passo> é adicionado à variável <controle> e o fluxo de execução é desviado para o item 1 acima, com os testes listados feitos novamente.

Caso 1: Comando DO ... END DO finito

→ Permite que sejam feitas um número finito de iterações, onde a cada iteração os comandos listados dentro do bloco serão executados.

```
DO <controle> = <início>, <fim>
      <comando 1>
      ...
      <comando n>
END DO
```

- Se o <passo> é omitido, será adicionado o valor 1 à variável <controle>:

  → neste caso, <inicio> deve ser menor do que <fim>;

  → <inicio> e <fim> devem ser variáveis ou expressões do tipo INTEGER.

Caso 1: Comando DO ... END DO finito

→ Permite que sejam feitas um número finito de iterações, onde a cada iteração os comandos listados dentro do bloco serão executados.

```
DO <controle> = <início>, <fim>, <passo>
      <comando 1>
      ...
      <comando n>
END DO
```

- Se o <passo> é maior do que zero, será adicionado o valor do <passo> à variável <controle>:

  → neste caso <inicio> deve ser menor do que <fim>

  → <inicio> e <fim> devem ser variáveis ou expressões do tipo INTEGER.

Caso 1: Comando DO ... END DO finito

→ Permite que sejam feitas um número finito de iterações, onde a cada iteração os comandos listados dentro do bloco serão executados.

```
DO <controle> = <início>, <fim>, <passo>
      <comando 1>
      ...
      <comando n>
END DO
```

- Se o <passo> é menor do que zero, será adicionado o valor do <passo> à variável <controle>:

  → neste caso <inicio> deve ser maior do que <fim>

  → <inicio> e <fim> devem ser variáveis ou expressões do tipo INTEGER.

Caso 1: Comando DO ... END DO finito

→ Permite que sejam feitas um número finito de iterações, onde a cada iteração os comandos listados dentro do bloco serão executados.

```
DO <controle> = <início>, <fim>, <passo>
      <comando 1>
      ...
      <comando n>
END DO
```

- O <passo> não pode ser igual a zero.
- Nunca mude o valor das variáveis <controle>, <início>, <fim> e <passo> dentro da estrutura de repetição.
- No Fortran 77 é permitido o uso de uma variável <controle> do tipo REAL. No Fortran 90 não é permitido.

Caso 1: Comando DO ... END DO finito

Exemplo: Cálculo do fatorial de um inteiro N

```
INTEGER :: i, N, fatorial
PRINT*,'Digite o valor de N:'
READ*,N
 fatorial = 1
DO i = 1, N
   fatorial =  fatorial * i
END DO
 print*,'O fatorial de N = ',N,' vale ', fatorial
END
```

Caso 2: Comando DO WHILE ... END DO finito

→ Permite que sejam feitas um número finito de iterações, enquanto uma condição lógica for atendida; a cada iteração os comandos listados dentro do bloco serão executados.

```
DO WHILE ( <condição> )
      <comando 1>
      ...
      <comando n>
END DO
```

1. Se a <condição> for avaliada como .TRUE. os comandos listados dentro da estrutura são executados.
2. Se a <condição> for avaliada como .FALSE. o comando imediatamente posterior ao END DO será executado.

Caso 2: Comando DO WHILE ... END DO finito

Exemplo: Cálculo do fatorial de um inteiro N

```
INTEGER :: i, N, fatorial
PRINT*,'Digite o valor de N:'
READ*,N
fatorial = 1
i = 2
DO WHILE (i <= N)
   fatorial =  fatorial * i
   i = i + 1
END DO
print*,'O fatorial de N = ',N,' vale ', fatorial
END
```

Caso 3: Comando DO ... END DO infinito

→ Permite que sejam feitas um número infinito de iterações; a cada iteração os comandos listados dentro do bloco serão executados.

```
DO
        <comando 1>
        ...
        <comando n>
END DO
```

1. O fluxo de execução será desviado para fora da estrutura usando o comando EXIT:

   → neste caso, nenhum dos comandos entre EXIT e END DO é executado.

Caso 3: Comando DO ... END DO infinito

Exemplo: Cálculo do fatorial de um inteiro N

```
INTEGER :: i, N, fatorial
PRINT*,'Digite o valor de N:'
READ*,N
fatorial = 1
i = 2
DO
   IF (i > N) EXIT
   fatorial =  fatorial * i
   i = i + 1
END DO
print*,'O fatorial de N = ',N,' vale ', fatorial
END
```

Caso 3: Comando DO ... END DO infinito

→ Permite que sejam feitas um número infinito de iterações; a cada iteração os comandos listados dentro do bloco serão executados.

```
DO
      <comando 1>
      ...
      <comando n>
END DO
```

1. O fluxo de execução será desviado dentro da estrutura usando o comando CYCLE:

   → neste caso, nenhum dos comandos entre CYCLE e END DO é executado e uma nova iteração dentro do laço é executada.