Programação estruturada no Fortran 90 - 3

Alexandre Diehl

Departamento de Física – UFPel

Subprograma do tipo SUBROTINA

Em Fortran 90 existem dois tipos de subprogramas: Funções e Subrotinas

- Uma Função, quando chamada, retorna um único valor calculado dentro da unidade de programa, associado com o nome da Função
- Uma Subrotina, quando chamada, pode retornar diversos valores calculados dentro da unidade de programa, associados com os argumentos usados na chamada da Subrotina

Subprograma do tipo SUBROTINA

Um subprograma do tipo SUBROTINA tem a seguinte sintaxe

```
<atributos> SUBROUTINE <nome> (arg1, arg2, ..., argn)
      IMPLICIT NONE
      [campo de declaração de variáveis]
      [campo de execução]
END SUBROUTINE <nome>
```

- A opção <atributos> é opcional, possibilitando o uso de atributos como RECURSIVE, com a mesma funcionalidade apresentada no caso de Funções.
- A Subrotina é nomeada pelo identificador <nome>. É ele que devemos usar para invocar o uso da Subrotina.

  → O nome da Subrotina não pode ser usado dentro de expressões aritméticas e/ou lógicas, nem em comando do tipo PRINT.

  → Para invocar a Subrotina, usamos o comando CALL <nome>.

Subprograma do tipo SUBROTINA

Um subprograma do tipo SUBROTINA tem a seguinte sintaxe

```
<atributos> SUBROUTINE <nome> (arg1, arg2, ..., argn)
      IMPLICIT NONE
      [campo de declaração de variáveis]
      [campo de execução]
      RETURN
END SUBROUTINE <nome>
```

- Se um dos comando dentro da Subrotina é uma instrução do tipo RETURN, a execução da Subrotina é encerrada e o programa segue a partir do ponto em que a Subrotina foi invocada.

  → Podem existir mais de uma instrução RETURN dentro da Subrotina, isoladamente ou em combinação com instruções do tipo IF.

  → A instrução RETURN é opcional. Sem ela, a execução da Subrotina é encerrada no comando END.

Subprograma do tipo SUBROTINA

Um subprograma do tipo SUBROTINA tem a seguinte sintaxe

```
<atributos> SUBROUTINE <nome> (arg1, arg2, ..., argn)
      IMPLICIT NONE
      [campo de declaração de variáveis]
      [campo de execução]
END SUBROUTINE <nome>
```

- Os argumentos da Subrotina são identificados por uma lista de identificadores mudos arg1, arg2, ..., argn.

  → Eles são usados na unidade de programa para produzir o resultado a ser extraído da Subrotina.

  → Os argumentos podem ser de entrada, INTENT(IN).

  → Os argumentos podem ser de saída INTENT(OUT).

Declaração de argumentos e identificadores em Subrotina

- Os argumentos de uma Subrotina **podem** ser declarados com o opção de intenção INTENT:

  → usado para se especificar como é passado um argumento para um subprograma e como ele é retornado.

- Opção de intenção de entrada, INTENT(IN):

  → neste caso o valor do argumento não pode ser alterado pela Subrotina

  → A sintaxe a ser usada é a seguinte

  INTENT(IN) :: <lista de argumentos mudos da Subrotina>

  → Sem o INTENT(IN) os valores dos argumentos mudos podem mudar pelos cálculos realizados dentro da Subrotina.

Declaração de argumentos e identificadores em Subrotina

- Os argumentos de uma Subrotina **podem** ser declarados com o opção de intenção INTENT:

  → usado para se especificar como é passado um argumento para um subprograma e como ele é retornado.

- Opção de intenção de saída, INTENT(OUT):

  → neste caso o argumento só pode ser usado em expressões e comandos depois de um valor ter sido atribuído a ele dentro da Subrotina.

  → A sintaxe a ser usada é a seguinte

  INTENT(OUT) :: <lista de argumentos mudos da Subrotina>

  → A opção INTENT(OUT) em geral é usada para extrair os resultados calculados dentro da Subrotina através dos argumentos mudos desta.

Subprograma do tipo SUBROTINA

As Subrotinas podem ser internas ou externas

- Subrotinas internas (com CONTAINS):

  → São definidas dentro de uma unidade de programa, a partir da instrução CONTAINS.

```
program teste
  implicit none
  [campo de declaracao de identificadores]
  [campo de execucao]
  CONTAINS
    [Subroutine 1]
    [Subroutine 2]
end program teste
```

- Todas as Subrotinas são **listadas a partir** do CONTAINS.
- O Programa Principal é **finalizado após** a declaração de todas as Subrotinas.

Subprograma do tipo SUBROTINA

Subrotina interna com CONTAINS

```
program testa_subrotina
  implicit none
  real :: number
  CALL GetNumber(number)
  print*,' numero digitado:', number
CONTAINS

  SUBROUTINE GetNumber(Input_Value)
    implicit none
    real, INTENT(OUT) :: Input_Value
    do
       write(*,*)' digite um numero positivo:'
       read (*,*)Input_Value
       if (Input_Value > 0.0) exit
       write (*,*)'Erro: tente novamente'
    end do
  end SUBROUTINE GetNumber

end program testa_subrotina
```

Subprograma do tipo SUBROTINA

As Subrotinas podem ser internas ou externas

- Subrotinas internas (**sem** CONTAINS):

  → Todas as Subrotinas devem ser **iniciadas e finalizadas** separadamente, **após a finalização** do Programa Principal.

```
program teste
  implicit none
  [campo de declaracao de identificadores]
  [campo de execucao]
end program teste
  [subroutine 1]
  [subroutine 2]
```

Subprograma do tipo SUBROTINA

Subrotina interna sem CONTAINS

```
program testa_subrotina
   implicit none
   real :: number
   CALL GetNumber(number)
   print*,' numero digitado:', number
end program testa_subrotina

SUBROUTINE GetNumber(Input_Value)
   implicit none
   real, INTENT(OUT) :: Input_Value
   do
      write(*,*)' digite um numero positivo:'
      read (*,*)Input_Value
      if (Input_Value > 0.0) exit
      write (*,*)'Erro: tente novamente'
   end do
end SUBROUTINE GetNumber
```

Subprograma do tipo SUBROTINA

Fatorial de n:

```
program fatorial_sub
  implicit none
  integer :: f, n
  print*,'n?'
  read*,n
  if (n >= 0) then
     CALL fatorial(n,f)
     write(*,'("fatorial(",I12,")=",I12)')n,f
  else
     print*,'valor invalido para n'
  end if

CONTAINS

  subroutine fatorial(n,f)
    [ ... ]
  end subroutine fatorial

end program fatorial_sub
```

Subprograma do tipo SUBROTINA

Fatorial de n:

```
program fatorial_sub

  [ ... ]
CONTAINS

  subroutine fatorial(n,f)
    implicit none
    integer, intent(in) :: n
    integer, intent(out) :: f
    integer :: i
    if (n >= 0) then
       f = 1
       do i = 2,n
          f = f*i
       end do
    else if (n == 0) then
       f = 1
    end if
  end subroutine fatorial
end program fatorial_sub
```

- Execução do código:

```
diehl@lua:~/Documents/ensino/UFPEL/prog_comput
 n?
12
fatorial(          12)=   479001600
diehl@lua:~/Documents/ensino/UFPEL/prog_comput
```

- Tente usar na Subrotina

  integer, intent(out) :: f = 1

  O que acontece na compilação?

Exemplo:

Considere uma partícula de massa m submetida a um movimento harmônico simples (MHS), produzido por uma mola de constante elástica κ. As equações para a posição, velocidade e aceleração da partícula se escrevem, respectivamente, como

$$x(t) = A\,\mathrm{sin}(\omega t + \phi) \qquad v(t) = \omega A\,\mathrm{cos}(\omega t + \phi) \qquad a(t) = -\omega^2 A\,\mathrm{sin}(\omega t + \phi)\ ,$$

onde $\omega = \sqrt{\kappa/m} = 10\,\mathrm{rad/s}$ são a frequêcia de oscilação e $\phi = \pi/2$ a fase inicial. Sabendo que a amplitude inicial do movimento é $A = 10\,\mathrm{cm}$, escreva um programa em Fortran 90 que avalia $x(t)$, $v(t)$ e $a(t)$ da partícula de zero até 2 segundos, usando um intervalo de tempo de 0.01 segundos entre cada ponto avaliado. A saída dos resultados deve ser apresentada num arquivo de saída, em formato exponencial (E) e com 4 casas decimais.

Faça versões sem e com Subprogramas.

Subprograma do tipo SUBROTINA

```
program mhs
  implicit none
  real :: t, x, v, a
  real :: w, amp, pi

  pi = 4.0*atan(1.0)
  w = 10.0
  amp = 0.10
  t = 0.0
  write (*,'(A)')'# MHS de uma particula'
  write (*,'(A)')'#      t            x            v            a'
  write (*,*)
  do
     if (t > 2.0) exit
     x = amp*sin(w*t+pi/2.0)
     v = w*amp*cos(w*t+pi/2.0)
     a = -w**2*amp*sin(w*t+pi/2.0)
     write(*,'(4(E11.4E2,1x))')t,x,v,a
     t = t + 0.01
  end do
end program mhs
```