Funções

Definição

- Trechos de código executáveis invocáveis a partir de um nome

Sintaxe

```
tipo FUNCTION  nomeFuncao (arg1,..., argn)
   IMPLICIT  NONE
   [Especificação]
   [execução]
   [subprogramas]
END FUNCTION  nomeFuncao
```

Funções

- Uma função em fortran é um trecho de código que recebe alguns parâmetros, executa alguma computação e retorna algum valor
- Logo, em algum ponto do código deve existir uma expressão do tipo

  nomeFuncao = expressão

- Esta instrução faz com que o resultado da expressão fique armazenado em uma variável automática onde oresultado da função deve ser armazenado

Funções

- Quando a execução do programa atingir a diretiva end function o valor armazenado na variável automatica nomeFuncao é retornado como resposta da chamada da função
- Qualquer instrução ou comando usado em um programa pode ser usado em uma função

Exemplo

```
INTEGER FUNCTION  Soma(a, b, c)
IMPLICIT  NONE
INTEGER, INTENT(IN) :: a, b, c
Soma = a + b + c
END FUNCTION  Soma
```

Funções

- INTEGER, INTENT(IN) :: a, b, c ????
- Esta linha especifica os dados formais dos parâmetros
- a,b,c são inteiros, e não podem ter seus valores alterados

Usando Funções

```
REAL FUNCTION  Average(x, y, z)
  IMPLICIT  NONE
   REAL, INTENT(IN) :: x, y, z
   Average = (x + y + z) / 3.0
END FUNCTION Average
```

Exemplo

```
PROGRAM  Avg
  IMPLICIT  NONE
  REAL :: a, b, c, m
  READ(*,*)  a, b, c
  m = Average(a, b, c)
  WRITE(*,*) a, b, c, m
CONTAINS
  REAL FUNCTION  Average(a, b, c)
    IMPLICIT  NONE
    REAL, INTENT(IN) :: a, b, c
    Average = (a + b + c) / 3.0
  END FUNCTION  Average
END PROGRAM  Avg
```

Usando Funções

- A função do slide anterior usa três números reais como argumento e retorna a média destes elementos

- Para fazer o uso (invocar) da função basta usar o nome da função passsando os argumentos pedidos. Ex Average(1.0,2.0,3.0)

Usando Funções

- O número de parâmetros (argumentos) especificado na função e usados na invocação deve ser o mesmo

- Os tipos dos parâmetros usados na invocação também devem ser os mesmos tipos especificados na função

Usando Funções

- Os parâmetros podem ser constantes, variáveis e expressões

- Caso algum parâmetro não seja declarado como intent(in) o valor deste parâmetro pode ser modificado. O evento de mudança do valor de um parâmetro em uma função, chamado de “efeito colateral”, não é uma boa prática de programação, devendo ser evitado sempre que possível

Detalhes importantes

- Programas podem conter funções
- Funções de programa podem conter funções internas
- Funções internas não podem conter funções internas

Exercício

- Faça uma função que receba um número como parâmetro e retorne um valor lógico indicando se este número é positivo ou não

Exercício

- Faça uma função que calcule o fatorial de um número

Exercício

- Faça uma função para ler um número positivo que o usuário irá digitar. A função deve exigir que o usuário digite um número até que um número positivo seja digitado

Erro

```
REAL FUNCTION  Teste(a, b)
   IMPLICIT  NONE
   INTEGER, INTENT(IN) :: a, b
   Teste = a*a + b*b
   Teste = SQRT(a*a - b*b)
END FUNCTION  Teste
```

Erro

- O valor retornado pela função é o último valor armazenado na variável de mesmo nome da função

Arrays e matrizes como parâmetro

```
Tipo Function nome(x, L, U)
    IMPLICIT  NONE
    INTEGER, INTENT(IN)                          :: L
    INTEGER, INTENT(IN)                          :: U
    INTEGER, DIMENSION(L:U), INTENT(IN) :: x
                .........
END Function nome
```

Arrays de forma assumida

```
PROGRAM  Test
IMPLICIT  NONE
INTEGER, PARAMETER :: MAX_S = 1000
REAL, DIMENSION(1:MAX_S) :: D
INTEGER :: tamReal,i
READ(*,*) tamReal
READ(*,*) (D(i), i=1, tamReal)
WRITE(*,*) "Soma = ", soma(D, 5, tamReal)
CONTAINS
 REAL FUNCTION  soma(x,i,f)
   IMPLICIT  NONE
   INTEGER, INTENT(IN) :: i,f
   REAL, DIMENSION(:), INTENT(IN) :: x
   INTEGER  ::c
   soma = 0.0
    DO c = i, f
              soma = soma + x(c)
    END DO
    END FUNCTION  soma
END PROGRAM  Test
```

Array de forma assumida

- Dois modos:
  - 1) início:
  - 2) :

Escopo

- Regras de escopo de um membro de programa (variável ou função) ditam a visibilidade e acessibilidade deste membro

- Usadas para responder perguntas tais como: uma função pode usar variáveis do programa principal? O programa principal pode usar variáveis das funções?

Escopo regra 1

- O escopo de um membro é o programa/função em que o membro é declarado

- Assim, variáveis declaradas dentro de funções não são visíveis fora destas funções

```
PROGRAM  Escopo
   IMPLICIT  NONE
    INTEGER          :: m, n
CONTAINS
   INTEGER FUNCTION  Funct1(k)
      IMPLICIT  NONE
      INTEGER, INTENT(IN) :: k
      REAL                :: f, g
   END FUNCTION  Funct1
END PROGRAM  Escopo
```

Escopo Regra 2

- Um membro global é visível dentro de membros internos
- Assim, variáveis de programa podem ser acessadas por funções internas
- Recomenda-se evitar este tipo de operação

```
PROGRAM  Escopo
  IMPLICIT  NONE
  INTEGER :: a = 1, b = 2, c = 3
  WRITE(*,*)  Add(a)
  c = 4
  WRITE(*,*)  Add(a)
  CONTAINS
  INTEGER FUNCTION  Add(q)
    IMPLICIT  NONE
    INTEGER, INTENT(IN) :: q
    Add = q + c
  END FUNCTION  Add
END PROGRAM Escopo
```

Escopo Regra 3

- Membros de escopos diferentes são membros diferentes, mesmo tendo o mesmo nome

- Assim é possível ter uma variável “i” no programa e uma variável “i” dentro de funções sem que alterações em uma delas reflitam em alterações na outra

Exemplo

```
PROGRAM  Escopo
  IMPLICIT  NONE
  INTEGER :: i, Max = 5
  DO i = 1, Max
    Write(*,*)  Sum(i)
  END DO
CONTAINS
  INTEGER FUNCTION  Sum(n)
    IMPLICIT  NONE
    INTEGER, INTENT(IN) :: n
    INTEGER             :: i, s
    s = 0
    DO i = 1, n
      s = s + i
    END DO
    Sum = s
  END FUNCTION  Sum
END PROGRAM  Escopo
```

Exercícios

- Sabe-se que 1 cm corresponde à 0.3937 polegadas e 1 polegada é igual à 2.54 cm. Escreva um programa que escreva na tela os valores correspondentes a 0.0, 0.5,1.0, ..., 10.0 cm em polegadas.

Exercício

- Escreva um programa que contenha uma função que seja capaz de dizer se três números representam os lados de um triângulo. O programa deve ler diversos conjuntos de números e só parar quando o usuário digitar um número <=0

Exercício

- Faça um programa que calcule o fatorial de números lidos. O programa deve encerrar quando um número negativo for digitado

Exercício

- Faça um programa que dado o tamanho de um conjunto e o tamanho de um subconjunto, seja capaz de calcular quantas combinações de subconjuntos são possíveis

Exercício

- Faça um programa que contenha uma função capaz de encontrar o máximo divisor comum entre dois números

Módulos

- Em muitos dos exercícios e exemplos anteriores há várias funções no final do arquivo
- Muitas funções podem ser comuns a vários programas
- Para agrupar funções de uso comum fortran permite a criação de Módulos

Módulos

- Conjunto de funções e constantes que não pode ser executado como program

- Necessita que outro programa o chame para que possa fazer suas tarefas

Exemplo

```
MODULE DegreeRadianConversion
   REAL, PARAMETER :: PI = 3.1415926
   REAL, PARAMETER :: Degree180 = 180.0
CONTAINS
   REAL FUNCTION DegreeToRadian(Degree)
      IMPLICIT  NONE
      REAL, INTENT(IN) :: Degree
      DegreeToRadian = Degree*PI/Degree180
   END FUNCTION  DegreeToRadian
   REAL FUNCTION RadianToDegree(radian)
      IMPLICIT  NONE
      REAL, INTENT(IN) :: Radian
      RadianToDegree = Radian*Degree180/PI
   END FUNCTION  RadianToDegree
END MODULE DegreeRadianConversion
```

Usando módulos

```
PROGRAM  MainProgram
   USE   DegreeRadianConversion
   IMPLICIT  NONE
   ..........
END PROGRAM  MainProgram
```

Usando partes específicas

```
MODULE  SomeConstants
   IMPLICIT  NONE
   REAL, PARAMETER :: PI = 3.1415926
   REAL, PARAMETER :: g = 980
   INTEGER         :: Counter
END MODULE  SomeConstants
```

Usando Partes específicas

```
PROGRAM  MainProgram
   USE   SomeConstants, ONLY: PI, Counter
   IMPLICIT  NONE
   ..........
END PROGRAM  MainProgram
```

Renomeando

```
PROGRAM  MainProgram
   USE   SomeConstants, Gravidade => g
   IMPLICIT  NONE
   INTEGER :: e, f, g
   ..........
END PROGRAM  MainProgram
```

Compilando Programas e Módulos

- gfortran someConstants.f90 ProgPrincipal.f90

Exercício

- Faça um módulo que contenha funções para lidar com análise combinatória (permutações, arranjos, combinações) e faça um programa que utilize este módulo