Subprogramas en Fortran 90

Seminario de Computación 2009

1. Algoritmos con nombre en Fortran 90

- Un algoritmo con nombre es la generalización de un operador

- En Fortran 90 los algoritmos con nombre pueden ser:
  - Intrínsecos (propios del compilador o del módulo)
  - Externos (propios del usuario o externos al módulo)

- Tipos de algoritmos con nombre o subprogramas:
  - Subroutines
  - Functions

1. Algoritmos con nombre en Fortran 90

- Esquema

Algoritmo <identificador> [(<param.> {, <param.>})]
{<declaración de parámetros>}
{<declaración de constantes>}
{<declaración de variables internas>}

Inicio
<acción>
Fin

2. SUBROUTINE (subrutinas o procedimientos)

- Equivalente a los algoritmos con nombre vistos en teoría.
- Modifican el valor de alguno de los parámetros reales.
- No devuelve valores (parámetros no declarados).

SINTAXIS:

```
SUBROUTINE nombre_subrutina ([parámetros])
IMPLICIT NONE
[Declaración de parámetros]
[Declaración de variables locales y constantes]
[Sentencias]
RETURN

[CONTAINS
subprogramas internos usados por la subrutina]

END [SUBROUTINE [nombre_subrutina]]
```

Subprogramas en Fortran 90

2. SUBROUTINE (características)

- Categoría de parámetros formales (se declara mediante el atributo INTENT de los parámetros):
  - Parámetro dato = INTENT (IN),
  - Parámetro resultado = INTENT (OUT),
  - Parámetro dato-resultado = INTENT (IN OUT)
  - INTENT(IN OUT) es la opción por defecto (=F77)

- Llamada desde el programa principal (u otro subprograma):
  CALL nombre_subrutina (parámetros_reales)

- Los argumentos pueden ser:
  - Datos escalares o expresiones,
  - Nombres de vectores o matrices,
  - Nombres de subprogramas intrínsecos,
  - Nombres de subprogramas externos (definidos por el usuario en ese u otro módulo)

- Debe contener la sentencia RETURN

2. SUBROUTINE

Ejemplo 1:

```
! Nombre del subprograma y parámetros

PROGRAM INTERCAMBIO
INTEGER A/3/,B/4/,C/0/

  CALL MODULO1(A,B,C)
  WRITE(*,*)'EL PRODUCTO ES:',C

END PROGRAM INTERCAMBIO


SUBROUTINE MODULO1(A,B,T)
  INTEGER A,B,T
  T=A*B
END SUBROUTINE MODULO1
```

2. SUBROUTINE

Ejemplo 2:

```
! Nombre del subprograma y parámetros

SUBROUTINE INTERCAMBIA(X,Y)
IMPLICIT NONE
        REAL,INTENT(IN OUT)::X, Y ! TIPO DE PARÁMETROS
        REAL:: TMP

        TMP = X
        X = Y
        Y = TMP

RETURN

END SUBROUTINE INTERCAMBIA
```

2. SUBROUTINE

Ejemplo 3:

```
SUBROUTINE MULMATRIZ(D1MAX,D2MAX,M,L,N,A,B,C)
INTEGER D1MAX,D2MAX,M,N,L
REAL*8 A(D1MAX,D2MAX),B(D1MAX,D2MAX),C(D1MAX,D2MAX)
REAL*8 suma

        DO I=1,M
                DO j=1,L
                suma=0.d0
                        DO k=1,N
                                suma=suma+a(i,k)*b(k,j)
                        END DO
                        c(i,j)=suma
                END DO
        END DO

RETURN
END SUBROUTINE MULMATRIZ
```

2. SUBROUTINE

Ejemplo 4:

```
SUBROUTINE VOLUMEN(VOL_ESFERA,R)
REAL VOL_ESFERA,R
REAL PI/3.1415927/

 IF (R.LT.0.) THEN
          WRITE(*,*)'El radio es negativo!'
          RETURN
 ELSE
         VOL_ESFERA=4/3*PI*R**3
 END IF

END SUBROUTINE VOLUMEN
```

2. SUBROUTINE

Ejemplo 5:

```
PROGRAM PRUEBA_SUBRUTINAS
IMPLICIT NONE

        WRITE (*,*) "DENTRO DEL PROGRAMA, FUERA Y ANTES DE LA SUBRUTINA"
        CALL SUBPROGRAMA()
        WRITE (*,*) "DENTRO DEL PROGRAMA, FUERA Y DESPUES DE LA SUBRUTINA"
STOP

CONTAINS

SUBROUTINE SUBPROGRAMA()
IMPLICIT NONE
        WRITE (*,*) “HOLIS, ESTAMOS DENTRO DE LA SUBRUTINA"
        RETURN
END SUBROUTINE SUBPROGRAMA


END PROGRAM PRUEBA_SUBRUTINAS
```

3. FUNCTIONS (Funciones)

- Algoritmos con nombre que devuelven un valor y que, generalmente, no modifican el resto de parámetros.

- En algún punto del cuerpo de la función tiene que haber una asignación:
  nombre_función = valor
  (también puede estar en una instrucción READ())

- Tiene que terminar con RETURN

- Para su uso como función propia o external, hay que declararla junto con el resto de los datos:

- ! Nombre de la función, implícitamente, con EXTERNAL
  REAL, EXTERNAL:: minimo

3. FUNCTIONS (Funciones)

***SINTAXIS*:**

```
[TIPO] FUNCTION NOMBRE_FUNCIÓN ([PARÁMETROS])
        [DECLARACIÓN DE PARÁMETROS]
        [DECLARACIÓN DE VARIABLES LOCALES Y CONSTANTES]

        [SENTENCIAS]
        RETURN
        [CONTAINS - PROCEDIMIENTOS INTERNOS A LA FUNCIÓN]
END [FUNCTION [NOMBRE_FUNCIÓN]]
```

LLAMADA DESDE OTRO SUBPROGRAMA O FUNCIÓN

```
<TIPO>, EXTERNAL:: NOMBRE_FUNCIÓN
VARIABLE = NOMBRE_FUNCIÓN (ARGUMENTOS)

... O ...

WRITE (*,*) “EL RESULTADO ES “, NOMBRE_FUNCIÓN (ARGUMENTOS)
```

4. CONCLUSIONES

1. Los subprogramas:
   - Facilitan la modularidad y estructuración de los algoritmos.
   - Facilitan la lectura e inteligibilidad de los algoritmos.
   - Permiten una economización del esfuerzo del programador al poder escribir código reutilizable en muchas partes de un mismo algoritmo.
   - Facilitan la depuración y mantenimiento de los programas.

2. Los subprogramas pueden ser funciones y subrutinas.

3. Las funciones son subrutinas con 0 ó más argumentos y que devuelven un único valor de retorno.

4. Las funciones pueden formar parte de expresiones o aparecer en la parte derecha de una sentencia de asignación pero nunca pueden constituir una sentencia aislada o aparecer en la parte izquierda de una asignación.

5. Existen dos tipos de funciones: intrínsecas y definidas por el usuario.

4. CONCLUSIONES

6. **Las funciones definidas por el usuario deben describirse dentro del algoritmo principal.**

7. **Los argumentos y variables declaradas dentro del cuerpo de una función (o subrutina) se denominan variables locales, las variables declaradas dentro del programa principal son variables globales. Los subprogramas tienen acceso a las variables globales aunque en el caso de que una variable local se denomine igual que una variable global tiene preferencia la primera.**

8. **Las subrutinas son subprogramas que no devuelven ningún resultado; sin embargo, gracias a la utilización de los efectos laterales es posible su utilización para permitir el "retorno" de varios resultados.**