TEMA A.4 Funciones y Subprogramas

A.4.1 Introducción

- En programas con cierta complejidad pueden darse los siguientes problemas:
  - Algoritmos complicados
  - Implementación dificultosa
  - Depuración difícil
  - Más documentación para hacer el programa comprensible
  - Se precisan tareas similares en varias partes del programa

- Tipos:
  - Funciones intrínsecas
  - Funciones de proposición aritmética (funciones sentencia)
  - Subprogramas función (**FUNCTION**)
  - Subrutinas (**SUBROUTINE**)

A.4.2 Funciones de proposición aritmética (funciones sentencia)

- *Utilización*: Se utilizan para llevar a cabo cálculos relativamente simples y repetitivos

- *Sintaxis*:

  Nombre (lista de variables)= definición de la función usando las variables contenidas en la lista

- Normas de utilización:
  - La definición de la función debe aparecer antes de cualquier proposición ejecutable (a continuación de la declaración de variables y matrices), y sólo está definida en la unidad de programa donde está contenida (programa principal o subrutinas).

  - Las funciones que impliquen resultados en doble precisión, complejos o lógicos deben declararse mediante una proposición que indique el tipo de número utilizado.

    p.ej. **REAL** IND
    IND(X,J,Z)=(-X)**(J**Z)

  - La función de proposición aritmética no debe incluir más de una sentencia.

- La función puede incluir funciones intrínsecas o de biblioteca.

➥ *Ejemplo propuesto*:

Calcular la integral $I = \int_a^b \left(e^x \cdot \sqrt{x} \cdot \cos\left(x^2+1\right) + \frac{1}{e^x \cdot \sqrt{x}} \right) dx$ mediante el método de Simpson, donde I=h/3· $(f_0+4f_1+2f_2+4f_3+2f_4+\ldots+f_{2n})$, siendo h=(b-a)/2n y fi=f(a+ih) donde i=0,1,2,...2n

```
C ************************************************
C Programa para el cálculo de integral por método de Simpson
C ************************************************
      F(x)=EXP(X)*SQRT(X)*ACOS(X**2+1.)+EXP(-X)/SQRT(X)
      READ(*,10) A, B, N
10    FORMAT (1x, 2F7.3, I3)
      H=(B-A)/(2.*N)
      S=F(A)+F(B)
      J=1
      DO 15 I=1,2*N-1
            S=S+F(A+I*H)*(3+J)
            J=-J
15    CONTINUE
      S=S*H/3.
      WRITE(*,20) S
20    FORMAT(1x,'Integral= ', F7.3)
      PAUSE
      STOP
      END
```

A.4.3 Subprogramas función (**FUNCTION**)

➥ *Utilización*: Se utilizan cuando para la definición de la función se requiere de más de una sentencia. Se usan del mismo modo que se usaría una función intrínseca.

➥ *Sintaxis*:

```
FUNCTION Nombre (arg1, arg2, ..., argn)
        Bloque de instrucciones
Nombre= ....
RETURN
END
```

- *Ejemplo propuesto:*
  - Cálculo del binomio de Newton $\binom{m}{n} = \frac{m!}{(m-n)!n!}$

- *Reglas de utilización:*
  - El nombre del subprograma **FUNCTION** suele ser el nombre de alguna variable del subprograma.
  - La ejecución termina al llegar a **RETURN**
  - El nombre de la función no puede utilizarse en el programa principal o en otro subprograma como variable.
  - Dependiendo del tipo de número que sea el resultado de la función, son de aplicación todas las normas expuestas para la declaración de variables. P.ej.

    FUNCTION FACT(K)
    INTEGER FACT
  - El tipo de argumentos (que son mudos) lo determinan los argumentos de la definición

    FUNCTION FUN(A,B,K)
    REAL K
    INTEGER B
  - Las variables utilizadas son independientes de las que haya en el programa principal. Lo mismo se aplica a la numeración de etiquetas.
  - No pueden llamarse a sí mismas, directa o indirectamente.
  - Debe haber el mismo número de argumentos en la llamada que en la definición.
  - Los nombres de los subprogramas **FUNCTION** pueden ser argumentos de otros subprogramas. En este caso se debe incluir su nombre en una sentencia **EXTERNAL** tras la declaración de variables en el programa principal.

Ejemplo propuesto:

Calcular el valor de ln(x) cuando 0.5 ≤ x ≤ 1.5 si se cumple que: $\ln(x) = \frac{x-1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \ldots + \frac{(x-1)^{16}}{16}$ utilizando Δx=0.1 unidades.

```
      PROGRAM Logaritmo
      DO 1 X=0.5,1.5,0.1
        WRITE(*,10) ALN(X), ALOG(X), (ALN(X)-ALOG(X))
10      FORMAT(1x,3(E16.10,3x))
1     CONTINUE
      PAUSE
      STOP
      END
```

```
C *****************************************************
C Subprograma FUNCTION ALN(x)
C *****************************************************
      FUNCTION ALN(X)
      ALN=0.
      DO 1 N=1,16
        ALN=ALN+((X-1)**N)/N*((-1)**(N-1))
1     CONTINUE
      RETURN
      END
```

Programa para el cálculo del binomio de NEWTON

```
      PROGRAM Newton
      INTEGER FACT
      READ (*,10) M, N
10    FORMAT (2I4)
      COMB=FACT(M) / (FACT(M-N) * FACT(N))
      WRITE(*,20) M, N, COMB
20    FORMAT(1x, 'Factorial de ',I4,' sobre ',I4,' = ',I8)
      PAUSE
      STOP
      END
C *****************************************
C Función para el cálculo de números factoriales
C *****************************************
      FUNCTION FACT(K)
      INTEGER FACT
      FACT=1
      DO 1 I=1,K
        FACT=FACT*I
1     CONTINUE
      RETURN
      END
```

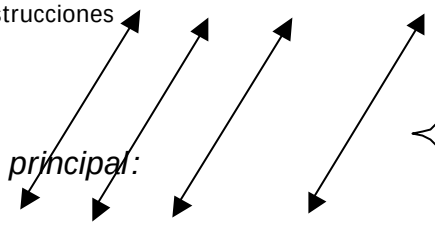
A.4.4 Subrutinas (**SUBROUTINE**)

- Su funcionamiento y normas son similares a los de los subprogramas **FUNCTION**, aunque con algunas diferencias:
  - No tienen valores asociados al nombre. Todas las salidas de información se definen como argumentos.

- La ejecución de una subrutina no comienza invocando sólo su nombre, sino que es necesario preceder este nombre de la instrucción **CALL**.
- Como resultado de la operación en una subrutina puede obtenerse más de una variable (argumentos de salida).
- Una subrutina puede llamar a otras.

➥ *Sintaxis en la propia Subrutina*:

SUBROUTINE *Nombre* (arg_1, arg_2, arg_3, …, arg_n)
Bloque de instrucciones
RETURN
END



Argumentos "mudos". Lo importante no es como se llaman sino qué posición ocupan en el conjunto.

➥ *Sintaxis en el programa principal:*

CALL *Nombre* (arg_1, arg_2, arg_3, …, arg_n)

- Los argumentos 1 á n son argumentos tanto de entrada como de salida.
- No se transmiten nombres de variables sino argumentos solo asociados a su posición en la llamada a la subrutina
- Si algún argumento de la subrutina es una función Fortran, es necesario declarar al comienzo del programa principal la función como **EXTERNAL**.

➥ *Ejemplo propuesto:*

Programa para el cálculo de raíces de un polinomio de segundo grado.

```
!  RAIZ IMAGINARIA.f90
!*************************************************************
!
!  PROGRAM: RAIZ IMAGINARIA
!
!
!*************************************************************
      program RAICES
      LOGICAL IMAG
      PRINT*, 'Escribe el valor de A,B y C'
      READ(*,*) A,B,C
      CALL RAIZ(A,B,C,X1,X2,IMAG)
      IF (IMAG.EQ..TRUE.) THEN
            PRINT*,'Raiz imaginaria, no existe solución'
      ELSE
            WRITE(*,5) X1,X2
5           FORMAT (1X,'La solución 1 es',2x,F8.4,/,'La solución
     #      es',2x,F8.4)
```

```
END IF
STOP
end program RAICES

SUBROUTINE RAIZ(AA,BB,CC,X1,X2,IMAG)
LOGICAL IMAG
IF((BB**2-4.*AA*CC).LT.0.) THEN
      IMAG=.TRUE.
ELSE
      X1=(-BB+SQRT(BB**2-4.*AA*CC))/2./AA
      X2=(-BB-SQRT(BB**2-4.*AA*CC))/2./AA
END IF
RETURN
END
```

Instrucción **COMMON**

- Es posible eliminar los argumentos de entrada y/o de salida de las instrucciones de invocación a subrutinas mediante la utilización de instrucciones COMMON. El sentido de estas instrucciones es el de hacer que una determinada dirección de memoria sea compartida por distintas unidades del programa (p.ej. programa principal y subrutina). De este modo, no será necesario incluir al conjunto de variables etiquetadas en paquetas COMMON con argumentos de entrada y/o salida, y su valor podrá modificarse tanto en el programa principal como en todas aquellas subrutinas en las que aparezcan.

- *Sintaxis*:

  COMMON /Nombre/ *lista de variables*

  Esta instrucción se coloca tras la declaración de variables, antes de la declaración de funciones sentencia.

  El nombre del conjunto **COMMON** y la lista de variables asociada a este nombre debe mantenerse fija en todas las unidades de programa en que se desee mantener el agrupamiento.

  p.ej.

```
PROGRAM Principal
Declaración de variables1
COMMON /Paquete1/ A0, k, Ea
      Resto del programa principal
STOP
END
```

```
**********************************************************
      SUBROUTINE Cinética (DHr)
      Declaración de variables2
      COMMON /Paquete1/ A0, k, Ea
            Resto de subrutina Cinetica
      RETURN
      END
**********************************************************
      SUBROUTINE Equilibrio (DG0,DH0)
      Declaración de variables3
      COMMON /Paquete1/ A0, k, Ea
            Resto de subrutina Equilibrio
      RETURN
      END
**********************************************************
```