Programación Básica
Laboratorio 7 / Proyecto: Estructuras de Datos y Funciones en Fortran
Mayo, 2024

Los objetivo de esta sesión son:

1. Solucionar dudas sobre sesiones anteriores
2. Comprender como operan los arreglos , estructuras y su relación con las funciones y procedimientos
3. Comprender el uso de funciones y procedimientos
4. Comprender el alcance del ámbito de las variables.

NOTAS

- **Todos los programas deben ir comentados y con el nombre del autor.**
- **Todos los ejercicios deben ir acompañados de la tabla de variables correspondiente.**
- **Todos los ejercicios deben ir acompañados del diagrama de flujo correspondiente.**

Funciones y procedimientos en Fortran

- Una función o procedimiento es un conjunto de instrucciones (declaraciones, definiciones, ...), que realizan una tarea especifica (unidad programática).
- En Fortran, el procedimiento *program* es la que indica donde comienza la ejecución de nuestro programa, es decir es el programa principal (procedimiento principal)
- Funciones vs. Procedimientos
  - Una función regresa un valor
  - Un procedimiento no regresa un valor, pero puede modificar sus argumentos.

En Fortran existen cuatro tipos de unidades programáticas:

1. *program* que es la unidad programática principal.
2. *subroutine* (subrutina o procedimiento) unidad programática que contempla instrucciones ejecutables.
3. *function* (función) unidad programática que contempla instrucciones ejecutables
4. *module* unidad programática que contempla instrucciones de declaración de variables, inicialización de variables, interfaces entre funciones y subrutinas.

Listado 1: Ejemplo de subprograma o subrutina

```
subroutine <nombre>[(<argumentos (ficticios)>)]
! instrucciones de declaración de los argumentos (ficticios)
:
! instrucciones de declaración de los objetos (variables) locales
:
! instrucciones ejecutables
:
end subroutine <nombre>
```

Una subrutina es subprograma de orden jerárquico de nivel inferior al programa principal. Sintaxis:

Donde los `<argumentos (ficticios)>`, en el caso que sean utilizados, son los objetos, sobre los cuales la subrutina trabajará preferentemente, están separados por comas y pueden ser variables, funciones, subrutinas, arreglos, apuntadores o procedimientos de módulo.

La subrutina es invocada por otra unidad programática por medio de la instrucción *call*.

Listado 2: Ejemplo invocación de la unidad programática

```
    <identificación unidad programática>
    :
    call <nombre>[<argumentos (usados)>]
    :
    end <unidad programática>
```

Listado 3: Ejemplo: procedimiento para obtener el área de un círculo

```
SUBROUTINE Area_Circulo(r,a)
  IMPLICIT NONE
  REAL, INTENT(IN) :: r
  REAL, INTENT(OUT) :: a
  REAL, PARAMETER :: Pi = 3.1415927
    A = Pi * r * r
  RETURN
END SUBROUTINE Area_Circulo
```

En Fortran es conveniente especificar que tipos de datos utiliza un procedimiento

Una función es un subprograma de orden jerárquico de nivel inferior al programa principal. Sintaxis:

Una FUNCTION se invoca de la misma forma que una función intrínseca. Es decir, en aquellas expresiones aritméticas en las que se desee evaluar el valor de la función se escribe:

Listado 4: Ejemplo: invocación del procedimiento para obtener el área de un círculo

```
PROGRAM Circulo
  IMPLICIT NONE
  REAL :: radio, area
  INTERFACE
    SUBROUTINE Area_Circulo(r,a)
    IMPLICIT NONE
    REAL, INTENT(IN) :: r
    REAL, INTENT(OUT) :: a
    END SUBROUTINE Area_Circulo
  END INTERFACE
  !
  write(*,*) "Radio del circulo (m) "
  read(*,*) radio
  CALL Area_Circulo(radio,area)
  ! Escribir resultado
  write(*,*) " El area del un circulo con radio", radio,"(m) es", area, "m2"
  !
  STOP
END PROGRAM Circulo
```

Listado 5: Ejemplo: Especificar que tipos de datos utiliza un procedimiento

```
interface
  subroutine lee_datos(z)
    real :: z
  end subroutine lee_datos
end interface
```

Listado 6: Sintaxis de una función

```
FUNCTION Nombre_funcion (lista de argumentos) RESULT (variable_resultado)
IMPLICIT NONE
! instrucciones de declaración de tipo de los argumentos
:
:
! instrucciones de declaración de tipo de variable_resultado
:
:
! instrucciones de declaración de tipo de las variables locales
:
:
! instrucciones ejecutables
:
:
RETURN
END FUNCTION Nombre_funcion
```

```
Nombre_funcion ( arg_1 , arg_2 , ... , arg_n)
```

Ejemplo: Obtener el mayor de tres números

Listado 7: Ejemplo: Función que obtiene el mayor de tres números

```
FUNCTION Mayor(A,B,C) RESULT (D)

  IMPLICIT NONE
  REAL :: D ! Variable resultado
  REAL, INTENT(IN) :: A, B, C ! Argumentos de la función
  REAL :: Dummy ! Variable local

    IF (A <= B) THEN
        D = B
    ELSE IF (A <= C) THEN
        D = C
    ELSE
        D = A
    END IF

END FUNCTION Mayor
```

Ejercicios

1. El programa en los listados 8 y 9 describen un programa en Fortran que calcula las potencias de un conjunto de números dados por el usuario. El programa utiliza una función y un procedimiento.

   - Edite y compile el programa
   - Verifique su funcionamiento
   - Comente el programa
   - A partir del código fuente genere el diagrama de flujo correspondiente.

2. Modifique la función potencia descrita anteriormente, como se muestra a continuación (listado 10) y mencione cuales son las diferencias al ejecutar el programa. ¿Cuál es su explicación para que exista tal diferencia?

3. Escriba un programa en Fortran que lea un arreglo de enteros de 10 elementos y encuentre el mínimo y máximo, utilizando funciones y procedimientos

4. Escriba un programa en Fortran que lea un arreglo de enteros de 10 elementos y los ordene. Utilice funciones y procedimientos

5. Escriba un programa en Fortran para resolver un sistema de partículas cargadas. El programa debe utilizar funciones y sub-rutinas. El programa también puede utilizar estructuras.

   Considere un sistema de $N = 100$ partículas puntuales, confinadas en el plano xy dentro de una caja cuadrada de longitud $L = 10$. Dichas partículas interaccionan vía el potencial de Coulomb, es decir

$$u(|\vec{r}_{ij}|) = K\frac{q_i q_j}{|\vec{r}_{ij}|}$$

Listado 8: Programa para calcular las potencias de un conjunto de números

```
program Cpotencia
! Declarar variables.
 implicit none
 real , dimension(10) :: datos
 integer , dimension(10) :: npotencias
 real , external ::potencia
 integer :: Max = 50;
 integer k

 do while ((Max .lt. 0) .or. (Max .gt. 10))
   print *,"¿Cuantos numeros deseas calcular su potencia (1-10)? "
   read *,Max
 enddo

 call lee_datos(datos,npotencias,Max)
 print *, "  X   |   n   |   Y"
 print *, "----------------------"

 do k=1,Max
    write (*,100) datos(k),npotencias(k),potencia(datos(k),npotencias(k))
100 format(1x,f4.1,1x,"|",1x,i3,2x,1x,"|",1x,f7.1)
 enddo
stop 'Programa potencias termino de forma normal'

end program Cpotencia
```

Listado 9: procedimientos para el programa que calcula las potencias de un conjunto de números

```
subroutine lee_datos(dat, nPot, n)
    integer :: j,n
    real, dimension(10) :: dat
    integer, dimension(10) :: nPot

    do j=1,n
    write (*,100) j
100 format("Teclea el valor (x) y su potencia (n) ",i3,":")
    read *,dat(j), nPot(j)
    enddo
  return
end subroutine lee_datos



real function potencia(a,m)
 integer :: m,j
 real   :: a
 real :: pot

 pot = 1.0
 do j=1,m
   pot = pot * a
 enddo
 potencia = pot
 return
end function potencia
```

Listado 10: Nueva función potencia

```
real function potencia(a,m)
 integer :: m,j
 real   :: a
 real :: pot = 1.0

 do j=1,m
   pot = pot * a
 enddo
 potencia = pot
 return
end function potencia
```

donde q_i y q_j son la carga de i-ésima y j-ésima partícula respectivamente, K es la amplitud del potencial de Coulomb y $|\vec{r}_{ij}|$ es la distancia que hay entre las dos partículas.

$$|\vec{r}_{ij}| = |\vec{r}_i - \vec{r}_j|$$

En ausencia de un campo externo, la energía de exceso total se define como:

$$U_{exc} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} u(|\vec{r}_{ij}|)$$

Notas:

(a) Utilice un generador de números aleatorios para colocar las N partículas de manera aleatoria dentro de la caja cuadrada.

(b) Calcule la energía de exceso total considerando que 500 partículas tienen carga $q_+ = 1$ y el resto tienen carga $q_- = -1$ para un valor de $K = 1.8$.

(c) Todos los valores son adimensionales

Características de la actividad

- Elaboración: En equipo
- Formato: Ver formato de reporte de laboratorio en moodle
- Colocar los NOMBRES COMPLETOS DE LOS AUTORES EN EL ENTREGABLE
- Colocar titulo del laboratorio en el entregable
- Colocar tabla de co-evaluación en el entregable
- Agregar en un archivo comprimido (zip) el código fuente de todos los programas
- Agregar en el mismo reporte, como anexo, la rúbrica de trabajo en equipo y rúbrica de reporte.
- Tipo de archivo final en MOODLE:
  - PDF para el reporte y anexos.
  - ZIP conteniendo los archivos del código fuente.
- Nombre del archivo:CA_E#_L08_CG

  Ejemplo: Reporte_E2_L08_PB

  *CG: Clave de Grupo | La clave del grupo es PB

  *CA: Clave de Actividad | Reporte, Código

El reporte del laboratorio debe contener los siguientes archivos:

Descripción	Nombre del archivo
Reporte del laboratorio	CA_E#_L08_CG ⇒ Reporte_E2_L08_PB
Código fuente de todos los programas	CA_E#_L08_CG ⇒ Codigo_E2_L08_PB