6.1.- ARRAYS

A lo largo del Capítulo 2 se mostraron los diferentes tipos de variables y constantes que puede ser empleados en FORTRAN. Los ejemplos y ejercicios que se han desarrollado han utilizado sus variantes mas importantes, sin embargo en todo momento hemos empleado variables que únicamente podían almacenar un único valor en un instante dado, el cual podía ir cambiando durante la ejecución del programa. Este tratamiento limita en gran medida el hecho de abordar almacenamientos grandes de datos del mismo tipo y relacionados entre si, ya que deberíamos realizar una sección de declaración de variables de longitud intratable con el fin de declarar todas las variables necesarias.

Si por ejemplo deseamos almacenar los números de teléfono de todos los vecinos de un edificio de 10 plantas y cuatro propietarios por planta. Una solución seria empezar a declarar tantas variables diferentes como vecinos con el fin de almacenar los números de teléfono respectivos; sin embargo este proceder podría ser eterno en caso de vivir en un rascacielos.

La solución en FORTRAN consiste en la utilización de ARRAYS los cuales como veremos permiten almacenar y operar con un volumen de datos elevado de manera muy simple.

Para el ejemplo anterior, la manera mas sencilla de almacenar los números de teléfono mediante un ARRAY seria mediante la siguiente variable:

DECLARACION	DESCRIPCION
INTEGER TELEF(40)	**INTEGER** TIPO DE VARIABLE **TELEF** NOMBRE DEL ARRAY **(40)** LONGITUD DEL ARRAY

En donde hemos definido una variable tipo ARRAY denominada TELEF. El numero que aparece entre paréntesis indica la longitud máxima del ARRAY, en este caso se ha seleccionado 40 para así poder almacenar como máximo hasta 40 números de teléfono diferentes. Por otro lado la declaración INTEGER ya es conocida e indica que cada uno de los cuarenta números de teléfono posibles serán enteros.

Con la declaración anterior durante la ejecución del programa podremos establecer la siguiente secuencia de asignación:

TELEF(1) = 215678 TELEF(2) = 345678 TELEF(3) = 456712 .... TELEF(39) = 231232 TELEF(40) = 212456	*Como puede observarse es el índice que aparece entre paréntesis el que controla el orden de almacenamiento y por ser un único índice, la variable de tipo ARRAY que estamos utilizando se denomina UNIDIMENSIONAL*

En este caso se ha optado por emplear un ARRAY entero ya que los números de teléfono pueden catalogarse como enteros, sin embargo los ARRAYS pueden ser también reales o de tipo carácter, a continuación se muestran algunos ejemplos:

```
REAL    VALOR(10), PHI(3)
CHARACTER*10  NOMBRE(14)
```

Todo lo anterior nos introduce en la técnica de como almacenar con una variable diferentes valores, mediante un ordenamiento controlado por el índice entero que aparece entre paréntesis. Ese índice y el nombre de la variable permitirá a lo largo del programa extraer cualquier valor del ARRAY de manera inmediata y operar como si de una

variable normal se tratara. A continuación se presentan algunos ejemplos de operaciones con variables tipo ARRAY declaradas previamente.

Ejemplo 1	***Acción***
REAL A, ASE(10), DER(10)	Se declaran tres variables reales la primera una variable real simple y las dos restantes de tipo ARRAY de longitud máxima 10
A = ASE(1)*DER(4)	Una variable simple **A** toma el valor del producto del valor numérico almacenado en la variable tipo ARRAY **ASE** en su posición 1 por el valor numérico almacenado en la variable tipo ARRAY **DER** en su posición (4)
Ejemplo 2	***Acción***
REAL PHI(5), VAL(5)	Se declaran dos variables reales de tipo ARRAY de longitud máxima 5
PHI(2) = VAL(3)/9.0	La posición 2 de la variable tipo ARRAY **PHI** toma el valor numérico de la variable tipo ARRAY **VAL** en su posición 3 divido por 9.0
Ejemplo 3	***Acción***
CHARACTER*4 NOM(5)	Se declara una variable carácter*4 de tipo ARRAY de longitud máxima 5
NOM(1) = 'HOLA'	La posición 1 de la variable tipo ARRAY **NOM** toma el valor HOLA

Un aspecto importante a resaltar es que cuando vayamos a emplear este tipo de variable durante el programa deberá ser escrita de manera completa, es decir el nombre de la variable deberá ir acompañada del paréntesis y el índice de la posición de almacenamiento para la cual realizamos la operación.

- Para el ***ejemplo 2*** anterior no seria correcto escribir la siguiente asignación:

PHI = VAL(3)/9.0

Error: PHI es una variable de tipo ARRAY y en la sentencia de asignación no aparecen los paréntesis y el índice de almacenamiento.

Por otro lado habrá que poner especial cuidado en declarar las variables tipo ARRAY con la longitud adecuada para poder almacenar todos los valores que vayan a ser necesarios, ya que de lo contrario podrían producirse errores como en que se muestra a continuación:

- Para el ***ejemplo 1*** anterior no seria correcto escribir la siguiente asignación:

A = ASE(13)*DER(4)

Error: ASE es una variable tipo ARRAY de longitud 10 y sin embargo la sentencia anterior opera con ASE de índice 13.

Otro aspecto a tener en cuenta es el hecho que la declaración de este tipo de variables permite almacenar un gran numero de datos y por lo tanto hay que tener especial cuidado a la hora de definir las longitudes de los ARRAYS declarados puesto que pueden requerir un espacio en memoria demasiado elevado. Tengamos en cuenta que un ARRAY real de longitud máxima 100 equivale a declarar 100 variable reales.

En relación con lo anterior FORTRAN permite una alternativa a la hora de definir la longitud de los ARRAY que bajo ciertas condiciones puede resolvernos algunos problemas. Supongamos que manejamos un ARRAY de longitud 100 pero durante la ejecución del programa únicamente son cargadas y utilizadas las posiciones de almacenamiento del ARRAY comprendidos entre el índice 50 y 70 es decir únicamente se están empleando 20

posiciones del ARRAY y las 80 restantes no se usan y sin embargo están ocupando memoria puesto que inicialmente el ARRAY se declaro para 100. En este caso especifico FORTRAN permite hacer la siguiente declaración ajustada al numero de posiciones que se desean emplear lográndose una reducción del espacio requerido en memoria.

DECLARACION ORIGINAL	DECLARACION AJUSTADA
REAL VALOR(100)	**REAL VALOR(50:70)**

La declaración ajustada anterior permite trabajar con índices altos pero mantener una ocupación de memoria reducida y ajustada a las necesidades de almacenamiento de la variable. Esto es muy útil para cuando deseamos mantener una correspondencia entre índices y entre distintas variables, puesto de que de lo contrario la declaración ajustada anterior podría simplificarse y ser sustituida por:

REAL VALOR(20)

Retomando el ejemplo inicial de los números de teléfono de los vecinos, se nos puede plantear la mejora de almacenamiento siguiente: ¿En vez de almacenar todos los teléfonos de manera consecutiva no seria mejor almacenarlos por planta?. Este nuevo ordenamiento nos adentra en la variable de tipo ARRAY BIDIMENSIONAL. Estas son una extensión del ARRAY UNIDIMENSIONAL y la única diferencia radica que ahora la variable consta de dos índices que controlan las posiciones de almacenamiento.

El modo mas simple de almacenar los números de teléfono por planta con un ARRAY BIDIMENSIONAL seria mediante la siguiente variable:

DECLARACION	DESCRIPCION
INTEGER*6 TELEF(10,4)	**INTEGER*6** TIPO DE VARIABLE **TELEF** NOMBRE DEL ARRAY **(10,4)** LONGITUD DEL ARRAY

Veamos la asignación de los números de teléfono por planta para comprender el modo de almacenamiento de este tipo de ARRAYS

TELEF(1,1) = 215678..... 1 º TELEFONO DEL 1º PISO
TELEF(1,2) = 345678..... 2 º TELEFONO DEL 1º PISO
TELEF(1,3) = 456712..... 3 º TELEFONO DEL 1º PISO
TELEF(1,4) = 656712..... 4 º TELEFONO DEL 1º PISO
TELEF(2,1) = 456716..... 1 º TELEFONO DEL 2º PISO
TELEF(2,2) = 456752..... 2 º TELEFONO DEL 2º PISO
.....
.....
TELEF(10,1) = 456716..... 1 º TELEFONO DEL 10º PISO
TELEF(10,2) = 586752..... 2 º TELEFONO DEL 10º PISO
TELEF(10,3) = 956472..... 3 º TELEFONO DEL 10º PISO
TELEF(10,4) = 856752 .....4 º TELEFONO DEL 10º PISO

Con lo anterior hemos completado el almacenamiento de todos los números de teléfono de los vecinos de manera que ahora podríamos mandar al programa que presentara por pantalla por ejemplo el tercer numero de teléfono del piso 5 del siguiente modo:

PRINT*, TELEF(5,3)

Con lo que nos aparecería el entero correspondiente al numero de teléfono.

Hasta el momento hemos descrito los ARRAYS mas usados que son los UNIDIMENSIONALES y BIDIMENSIONALES, sin embargo FORTRAN permite ARRAYS de dimensiones aun mayores, su construcción y funcionamiento es similar a los ya explicados.

6.2.- COMBINACION DE LAZOS Y ARRAYS

La verdadera capacidad operativa del almacenamiento mediante ARRAYS se consigue mediante la utilización de lazos DO. Como ya vimos un lazo DO permite mediante su *variable de lazo* **i** obtener una secuencia de valores enteros consecutivos que pueden servir de manera optima para controlar los valores de los índices de almacenamiento de los ARRAYS. Esta combinación se muestra en el siguiente programa ejemplo relacionado con el problema de los números de teléfono ya planteado.

```
      PROGRAM TELFO
      IMPLICIT NONE
      INTEGER I,J,TF(4,2)
C     CARGO EL ARRAY
      TF(1,1)=2156777
      TF(1,2)=2557557
      TF(2,1)=2154777
      TF(2,2)=2551187
      TF(3,1)=2853477
      TF(3,2)=2946377
      TF(4,1)=2853477
      TF(4,2)=2357377
C     PRESENTO NUMEROS POR PISOS
      DO I=1,4
      PRINT*,'PISO=',I
      DO J=1,2
      PRINT*,'TFNO',',TF(I,J)
      ENDDO
      ENDDO
      END
```

El programa anterior carga inicialmente los números de teléfono para un edificio de 4 plantas y dos propietarios por planta y los presenta por pantalla agrupados por planta. Se recomienda escribir el programa, compilarlo y ejecutarlo con el fin de entender como funciona.

EJERCICIO 6.1

- Realizar las siguientes modificaciones y mejoras en el programa ejemplo anterior:

  1º- Aumentar el numero de propietarios por planta a 4
  2º- Crear un nuevo ARRAY tipo carácter para asignar las letras A,B,C y D a los propietarios de cada planta 1,2,3 y 4 respectivamente. El nuevo programa deberá presentar el piso, el numero de teléfono y la letra correspondiente.

EJERCICIO 6.2

- Realizar un programa que lea dos vectores R^3 y calcule su producto escalar. Emplear ARRAYS unidimensionales para este propósito.

Continuando con la descripción de las posibilidades que la combinación de lazos DO y ARRAYS permite, a continuación realizaremos un recorrido básico al tratamiento de matrices mediante este tipo de sentencias. Para ello comenzaremos con un ejemplo típico de almacenamiento de una matriz de 3x3.

$$\begin{pmatrix} 2 & 1 & 4 \\ 5 & 6 & 8 \\ 2 & 4 & 9 \end{pmatrix} = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix}$$

La expresion anterior muestra un ejemplo de correlacion entre los valores numericos de la matriz y los indices de los terminos de la misma. Una vez hecho esto podriamos establecer la siguiente declaracion para almacenar la matriz anterior:

REAL MAT(3,3)

La asignacion de coeficientes se describe a continuacion:

MAT(1,1) = T_{11} = 2	MAT(2,1) = T_{21} = 5	MAT(3,1) = T_{31} = 2
MAT(1,2) = T_{12} = 1	MAT(2,2) = T_{22} = 6	MAT(3,2) = T_{32} = 4
MAT(1,3) = T_{13} = 4	MAT(2,3) = T_{23} = 8	MAT(3,3) = T_{33} = 9

Veamos ahora un programa ejemplo dedicado a la suma de matrices de 3X3.

```
C     LEO LAS MATRICES
      DO I=1,3
         DO J=1,3
            READ*,MAT1(I,J)
         END DO
      END DO
      DO I=1,3
         DO J=1,3
            READ*,MAT2(I,J)
         END DO
      END DO
C     SUMO LAS MATRICES Y
C     PRESENTO LOS RESULTADOS
```

```
      DO  I = 1, 3
          DO  J = 1, 3
          S = MAT1(I, J) + MAT2(I, J)
          SPRINT*, I, J, S
      ENDDO
      ENDDO
      END
```

EJERCICIO 6.3

- Realizar las siguientes modificaciones y mejoras en el programa ejemplo anterior:

  1º- Permitir que el programa sume matrices de hasta 6X6 y que sea el usuario el que defina durante la ejecución el tamaño de las matrices a tratar.

EJERCICIO 6.4

- Realizar un programa que lea una matriz cuadrada NXN y obtenga su traspuesta. El valor N deberá ser introducido por el usuario y se deberá advertir al usuario el tamaño máximo de las matrices permitido.

EJERCICIO 6.5 (OPCIONAL)

- Realizar un programa que lea dos matrices cuadradas y obtenga matriz producto.

6.3.- SENTENCIA DATA

En ocasiones cuando hemos finalizado la escritura de un programa y realizamos un repaso detallado del mismo observamos que aparecen en diferentes líneas del código sentencias de asignación fijas como las siguientes:

Ejemplos:

```
C = 'HOLA'
A = 5.6
PI = 3.14
Y(2) = 4.5
I = 4
```

Este tipo de sentencias, por regla general sirven para dar un valor inicial a una variable y se supone que pueden cambiar su valor a lo largo del programa, su localización a lo largo del código suele responder a las necesidades de su utilización y por ello no suelen estar agrupadas.

Con el fin de agrupar este tipo de sentencias FORTRAN permite a través de su sentencia DATA incluida en la sección de declaración de variables describir las asignaciones deseadas de manera ordenada. Esto permite en el futuro poder cambiar las asignaciones en la sección de declaración de variables sin tener que buscar una a una las sentencias de asignación a lo largo del código.

Para el ejemplo de las cinco variables anteriores la sentencia DATA equivalente se muestra a continuación:

```
REAL        A,PI,Y(5)
```

```
INTEGER          I
CHARACTER*4      C
DATA  A,PI,Y(5),I,C/5.6,3.1415,4.5,4,'HOLA'/
```

Obsérvese que la sentencia DATA tiene la siguiente construcción y características:

- **DATA nombre de variables separadas por comas/valores de las variables respectivas separadas por comas/**

- Todas las variables incluidas en la sentencia DATA deberán estar declaradas previamente.

Supongamos ahora de deseamos hacer la asignación siguiente:

```
A=5.0
B=5.0
C=5.0
D=5.0
```

En este caso tenemos la opción dos opciones de sentencia DATA:

Opción 1:

```
REAL   A,B,C,D
DATA  A,B,C,D/5.0,5.0,5.0,5.0/
```

Opción 2:

```
REAL   A,B,C,D
DATA  A,B,C,D/4*5.0/
```

Para el caso de un array unidimensional podemos simplificar la asignación del siguiente modo:

```
REAL VALOR(3)
DATA   (VALOR(I),I=1,3)/3.2,4.5,6.7/
```

Siendo:

```
VALOR (1) = 3.2
VALOR (2) = 4.5
VALOR (3) = 6.7
```

Si estamos nos planteamos asignar los números de teléfono de los vecino mediante una sentencia DATA podríamos establecer la siguiente declaración:

```
INTEGER   TELEF(2,2)
DATA ((TELEF(I,J),I=1,2),J=1,2)/214356,768954,675432,347890/
```

En resumen este tipo de sentencia permite agrupar asignaciones que de otra manera pueden estar desperdigadas a lo largo del código y que su localización para su revisión o modificación puede ser engorrosa.

6.4.- SENTENCIA PARAMETER

Otra sentencia similar a la sentencia DATA, es la sentencia PARAMETER, su función vuelve a ser definir una sentencia incluida en la sección de declaración de variables que permita asignar un valor inicial a una variable previamente declarada. La diferencia con la sentencia DATA estriba en que las asignaciones que se realicen con la sentencia PARAMETER son inamovibles a lo largo del programa. Si retomamos los ejemplo de asignación anteriores:

```
C = 'HOLA'
```

A = 5.6
PI = 3.14
Y(2) = 4.5
I = 4

Podemos pensar que de todas las variables que se muestran PI puede considerarse como una variable que va a mantener su valor constante y por lo tanto inamovible a lo largo del programa de tal forma que cuando se empleada en alguna expresión aritmética estemos seguros de que PI=3.1415.

En este caso la sentencia PARAMETER es la mas adecuada y su descripción se muestra a continuación:

```
REAL  PI
PARAMETER   (PI = 3.1415)
```

También se permite dentro de la sentencia PARAMETER realizar asignaciones con operaciones incluidas tal y como se muestra a continuación:

```
REAL  PI, DOSPI
PARAMETER   (PI = 3.1415, DOSPI=2*PI)
```

Por ultimo mostrar la utilización de la sentencia PARAMETER para establecer las longitudes de los arrays.

```
INTEGER  LMIN,LMAX
PARAMETER  (LMIN=5,LMAX=100)
REAL   MAT1(LMIN,LMIN), MAT2(LMAX,LMAX)
```

De este modo si se desea modificar la longitud de los ARRAYS MAT1 y MAT2, no será preciso variable por variable cambiando las longitudes, será suficiente con cambiar el valor en la sentencia PARAMETER y de ese modo afectará inmediatamente a todas los ARRAYS.

EJERCICIO 6.6

- Modificar el programa del EJERCICIO 6.1 e incluir las sentencias DATA y PARAMETER adecuadas para mejorar el programa.