Vectores, matrices y variables N dimensionales.

Seminario de Computación - 2009

Ejemplo: Definición y asignación de un vector.

```
PROGRAM  vector
IMPLICIT NONE

!Declaramos un vector con diez elementos.
REAL, DIMENSION(10) :: MI_VECTOR

!Asignamos los elementos del vector.
MI_VECTOR=(/1 , 2, 3, 4, 5, 6, 7, 8, 9, 10 /)


!Modificamos o asignamos un valor solo a un elemento del vector.


MI_VECTOR(1)=20


!Modificamos o asignamos un valor a una parte del vector.

MI_VECTOR(2:3)=(/3 , 4 /)


WRITE(*,*)MI_VECTOR


STOP
END PROGRAM vector
```

El resultado de la ejecución de este programa es que el contenido de la variable vector es:

20 3 4 4 5 6 7 8 9 10

Otro ejemplo de declaración y asignación de vectores.

```
PROGRAM  vector
IMPLICIT NONE

!Declaramos un vector con diez elementos.
REAL, DIMENSION(10)  :: MI_VECTOR
INTEGER              :: I
INTEGER, DIMENSION(3) :: SUBS=(/1, 7, 10/)

!Asignamos (en este caso como colocamos un solo valor todos los elementos
!del vector van a ser modificados a este valor).
!Hay que tener mucho cuidado con esto, porque en el caso de que nos
!olvidemos de especificar la componente a la cual estamos haciendo una asigancion,
!el programa podria asignarle el valor destinado a una componente a todo el vector,
!lo cual no resultaria en un error pero si en un resultado erroneo.

MI_VECTOR=0.


!También podemos usar un DO implicito para asignar un vector.

MI_VECTOR=(/(I,I=1,10)/)


!También podemos usar un vector INTEGER para designar los elementos de un vector.

MI_VECTOR(SUBS)=2.


WRITE(*,*)MI_VECTOR


STOP
END PROGRAM vector
```

En este caso el contenido de mi_vector es 2 2 3 4 5 6 2 8 9 2

Otro ejemplo usando la sentencia READ para asignar los valores de un array.

```
PROGRAM  vector
IMPLICIT NONE

!Declaramos un vector con diez elementos.
REAL, DIMENSION(5)    :: MI_VECTOR
INTEGER               :: I


!Asignamos los valores utilizando la sentencia READ.
!Al ingresar cada componente es igual a si estuvieramos leyendo varias variables.

WRITE(*,*)"Ingrese las 5 componentes del vector MI_VECTOR separadas por comas o espacios"
READ(*,*)MI_VECTOR(1),MI_VECTOR(2),MI_VECTOR(3),MI_VECTOR(4),MI_VECTOR(5)

WRITE(*,*)MI_VECTOR


!El READ también se puede hacer usando un DO implicito como vimos en un ejemplo anterior.

WRITE(*,*)"Ingrese las 5 componentes del vector MI_VECTOR separadas por comas o espacios"
READ(*,*)(MI_VECTOR(I),I=1,5)

WRITE(*,*)MI_VECTOR


STOP
END PROGRAM vector
```

En este ejemplo el contenido de MI_VECTOR es determinado por los números que el usuario del programa ingresa por pantalla.

Ejemplo: Definición de matrices.

```
PROGRAM  matriz
IMPLICIT NONE

!Declaramos una matriz de 2x2
REAL, DIMENSION(2,2)  :: MI_MATRIZ
INTEGER               :: I,J

!Otras formas equivalentes son
!REAL, DIMENSION(1:2,1:2)  :: MI_MATRIZ
!REAL                      :: MI_MATRIZ(2,2)


!Puedo asignar los valores de la matriz con un DO

DO J=1,2
  DO I=1,2
    MI_MATRIZ(I,J)=(I-1)*2+J
  ENDDO
ENDDO
!Nota: siempre que se barran los elementos de una matriz utilizando dos do
!anidados, es más eficiente hacerlo barriendo los elementos de la matriz por
!columnas y no por filas, esto es porque la información se guarda por columnas
!en la memoria de la maquina.

WRITE(*,*)"Asi esta ordenada la matriz en la memoria de la maquina"
WRITE(*,*)MI_MATRIZ

WRITE(*,*)"Asi es como representariamos nosotros la matriz"
WRITE(*,*)MI_MATRIZ(1,:)
WRITE(*,*)MI_MATRIZ(2,:)

STOP
END PROGRAM matriz
```

El resultado del programa anterior es:

```
 Asi esta ordenada la matriz en la memoria de la maquina
   1.000000       3.000000       2.000000       4.000000
 Asi es como representariamos nosotros la matriz
   1.000000       2.000000
   3.000000       4.000000
Press any key to continue
```

Cuando imprimimos por pantalla toda la matriz, el programa muestra los resultados de la misma forma en que están ordenados en la memoria de la máquina (es decir primero la primera columna y luego la segunda columna).

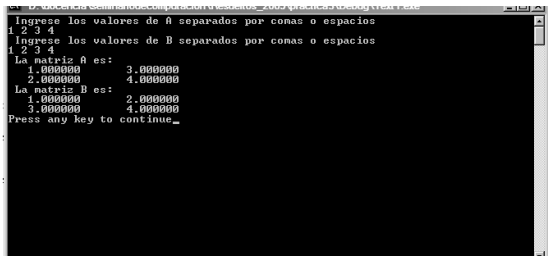
En el segundo WRITE modificamos la forma en la que se imprime la matriz para que se parezca a la forma como nosotros la escribiríamos.

También podemos asignar el valor a una matriz utilizando un DO implícito como en el siguiente ejemplo.

```
PROGRAM  matriz
IMPLICIT NONE

! Declaramos una matriz de 2x2
REAL, DIMENSION(2,2)   :: A , B
INTEGER                :: I,J

WRITE(*,*)"Ingrese los valores de A separados por comas o espacios"

READ(*,*)((A(I,J),I=1,2),J=1,2)

WRITE(*,*)"Ingrese los valores de B separados por comas o espacios"

READ(*,*)((B(I,J),J=1,2),I=1,2)

! Si los valores ingresados por pantalla son exactamente los mismos
! supongamos 1 2 3 4 como son A y B???

WRITE(*,*)"La matriz A es:"
WRITE(*,*)A(1,:)
WRITE(*,*)A(2,:)

WRITE(*,*)"La matriz B es:"
WRITE(*,*)B(1,:)
WRITE(*,*)B(2,:)

STOP
END PROGRAM matriz
```

En este caso utilizamos DO implícitos anidados para asignar las componentes de A y B.

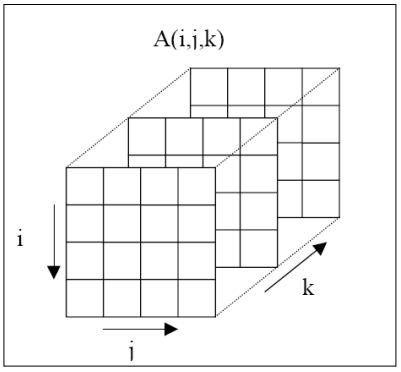
Si el INPUT es 1 2 3 4 en ambos casos, como es A y como es B?



A se completa por columnas y B se completa por filas, debido a que invertimos el orden de los DO implícitos.

La lectura de matrices usando DO implícitos es una herramienta muy importante y la vamos a ver en detalle más adelante cuando la utilicemos para leer datos de un archivo.

También podemos definir variables con más de 2 dimensiones.



Por ejemplo un arreglo de 3 dimensiones como el de la figura.

Que tiene 4 x 4 x 3 elementos.

Ejemplo: Declaración y asignación de un array de 3 dimensiones.

```
PROGRAM  matriz
IMPLICIT NONE

!Declaramos una matriz de 2x2x2
REAL, DIMENSION(2,2,2)  :: A
INTEGER                 :: I,J,K

WRITE(*,*)"Ingrese los valores de A separados por comas o espacios"

READ(*,*)(((A(I,J,K),I=1,2),J=1,2),K=1,2)


!Si los valores ingresados por pantalla son exactamente los mismos
!supongamos 1 2 3 4 como son A y B???


!Imprimimos A
WRITE(*,*)"A(:,:,1)"
WRITE(*,*)A(1,:,1)
WRITE(*,*)A(2,:,1)

WRITE(*,*)"A(:,:,2)"
WRITE(*,*)A(1,:,2)
WRITE(*,*)A(2,:,2)


STOP
END PROGRAM matriz
```

Si el Input es 1 3 2 4 5 7 6 8 , ¿Qué es lo que escribe este programa por pantalla?

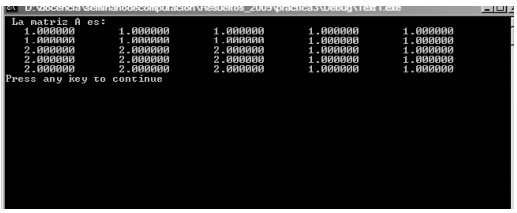
Resultado de la ejecución del programa anterior con el input indicado.

```
 Ingrese los valores de A separados por comas o espacios
1 3 4 2 5 7 6 8
 A(:,:,1)
   1.000000       4.000000
   3.000000       2.000000
 A(:,:,2)
   5.000000       6.000000
   7.000000       8.000000
Press any key to continue
```

Todas las reglas que vimos que se pueden aplicar para seleccionar una parte de un vector se aplican a las matrices o variables de más dimensiones.

```
PROGRAM  matriz
IMPLICIT NONE

!Ejemplo de seleccion de una parte de la matriz A
REAL, DIMENSION(5,5)  :: A
INTEGER               :: I,J

!A partir de la ejecucion de esta sentencia A deberia ser todos 1.
A=1.


!A partir de esta otra un sector de A pasa a valer 2.
A(3:5,1:3)=2.


WRITE(*,*)"La matriz A es:"
DO I=1,5
   WRITE(*,*)(A(I,J),J=1,5)
ENDDO


STOP
END PROGRAM matriz
```

¿Cómo es la matriz A que imprime por pantalla este programa?

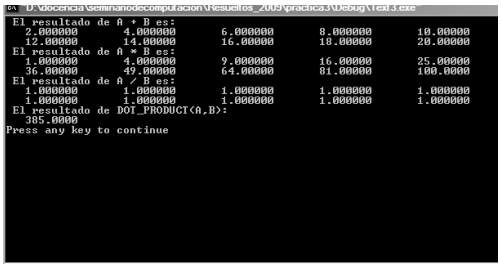


De acuerdo con lo que dice el código, cuando las filas son mayores o iguales que 3 y las columnas están entre 1 y 3 los elementos de A valen 2, en las restantes ubicaciones valen 1.

Ejemplos de operaciones matematicas con arrays numericos

```
PROGRAM array_oper
IMPLICIT NONE

REAL , DIMENSION(10) :: A , B
INTEGER              :: I

!Este programa muestra como se opera matematicamente con arrays.

A=(/( I , I=1,10 )/)
!Puedo asignar todo el contenido de un array a otro array.
B=A

!Podemos operar directamente con todo el array (en este caso las operaciones
!se hacen componente a componente.

WRITE(*,*)"El resultado de A + B es:"
WRITE(*,*)A+B

WRITE(*,*)"El resultado de A * B es:"
WRITE(*,*)A*B

WRITE(*,*)"El resultado de A / B es:"
WRITE(*,*)A/B

!Tambien en el caso de vectores podemos recurrir a la funcion intrinseca
!DOTPROD para calcular el producto interno entre A y B.

WRITE(*,*)"El resultado de DOT_PRODUCT(A,B):"
WRITE(*,*)DOT_PRODUCT(A,B)

STOP
END PROGRAM array_oper
```

Este es el resultado obtenido al ejecutar el programa anterior.



Tener en cuenta que siempre que realizamos operaciones con arrays o partes de los mismos, los elementos intervinientes en las operaciones deben tener las mismas dimensiones. Es decir si A y B son dos vectores y A tiene 10 elementos y B 20 elementos, la sentencia A+B va a resultar en un error (generalmente al momento de la compilación)

Utilización de parámetros para definir el tamaño de los arrays.

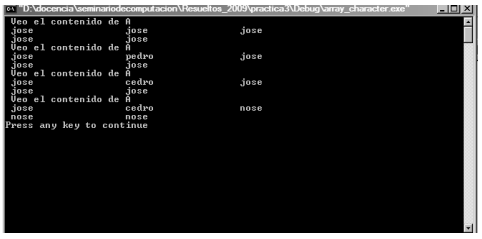
```
PROGRAM parameterdim
IMPLICIT NONE

INTEGER, PARAMETER     :: N=10 , M=20 , L=30
REAL, DIMENSION(N,M,L) :: A
```

Esto puede resultar muy útil si el programa que estamos haciendo requiere declarar varias variables con la misma dimensión y queremos que el código sea fácil de modificar.

Los arrays también se pueden aplicar a otros tipos de variables. Ejemplo con una variable de tipo CHARACTER.

```
PROGRAM  array_character
IMPLICIT NONE

! Definimos un array character donde cada elemento tiene 20 caracteres
CHARACTER(LEN=20), DIMENSION(5)  :: A

A="jose"

WRITE(*,*)"Veo el contenido de A"
WRITE(*,*)A

A(2)="pedro"

WRITE(*,*)"Veo el contenido de A"
WRITE(*,*)A

! Si queremos mostrar o asignar una cadena de caracteres a un segmento de uno de
! los elementos del array, primero tenemos que indicar el subindice correspondiente
! al elemento, y luego el / los subindices correspondientes al segmento dentro del elemento deseado
A(2)(1:2)="ce"

WRITE(*,*)"Veo el contenido de A"
WRITE(*,*)A

! Tambien podemos indicar varios elementos a la vez como en el caso de los arrays numeréricos.
A(3:5)(1:1)="n"

WRITE(*,*)"Veo el contenido de A"
WRITE(*,*)A

STOP
END PROGRAM array_character
```

Resultado obtenido al ejecutar el programa.



```
PROGRAM array_logical
IMPLICIT NONE
LOGICAL, DIMENSION(10)  :: LOGIC1 , LOGIC2 , LOGIC3
REAL    , DIMENSION(10)  :: A , B

LOGIC1=.TRUE.

LOGIC2=.FALSE.


LOGIC2(2)=.TRUE.

WRITE(*,*)"LOGIC2 es:"
WRITE(*,*)LOGIC2

!Los operadores lógicos en este caso operan componente a componente.

WRITE(*,*)"El resultado de LOGIC2 .AND. LOGIC1 es:"
WRITE(*,*)LOGIC2 .AND. LOGIC1

!Un array LOGICAL tambien se puede asignar mediante el resultado de una operación logics
!con variables numericas (o incluso CHARACTER).

A=(/1,2,3,4,5,6,7,8,9,10/)
B=(/1,2,3,4,5,4,7,8,9,10/)

LOGIC3= ( A == B )

WRITE(*,*)"LOGIC3 es:"
WRITE(*,*)LOGIC3

STOP
END PROGRAM array_logical
```

Los arrays lógicos son muy útiles en Fortran 90, ya que nos permitirán aprovechar mucho mejor funciones intrínsecas que se aplican a arrays que veremos en las próximas clases.

Resultado obtenido al ejecutar el programa.