Notas (breves) de FORTRAN 90.

Arreglos y asignación dinámica de memoria.

Lo que hemos visto sobre arreglos en Fortran 77 sigue siendo válido en Fortran 90. Lo que nos permite hacer el Fortran 90, que no nos permite el Fortran 77, es la asignación dinámica de memoria. Esto quiere decir que podemos dimensionar los arreglos durante la ejecución del programa, sin tener que asignarles un tamaño al momento de declararlos. Para esto, los arreglos deben declararse con el atributo `ALLOCATABLE`, por ejemplo:

```
REAL, ALLOCATABLE :: V(:), A(:,:)
```

donde `V` será un vector y `A` una matriz. Para asignarles un tamaño, usamos la sentencia `ALLOCATE`, por ejemplo:

```
ALLOCATE(V(5))
ALLOCATE(A(N,M))
```

donde `V` tendrá 5 componentes y `A` será un arreglo de dimensión $N \times M$, donde los valores de `N` y `M` son asignados previamente por el usuario, digamos por ejemplo, por teclado o a partir de los datos leidos de un archivo.
Usando la sentencia `DEALLOCATE` liberamos la memoria cuando dejamos de utilizar los arreglos:

```
DEALLOCATE(V)
DEALLOCATE(A)
```

Esto además, nos permite a un mismo arreglo, asignarle luego de liberar la memoria una dimensión diferente a la que le asignamos previamente.
Por otro lado, podemos asignarle valores a los arreglos como veníamos haciéndolo o utilizando:

```
V = [1.0, 2.0, 3.0, 4.0, 5.0]
A = RESHAPE([2.0, 3.0, 7.0, 5.0, 6.0, &
             7.0, 4.0, 2.0, 3.0, 9.0, &
             8.0, 3.0, 5.0, 7.0, 8.0, &
             1.0, 3.0, 5.0, 0.0, 4.0],[5,4])
```

La función intrínseca `RESHAPE` contruye un arreglo bidimensional a partir de uno unidimensional cuyas componentes son los elementos de la matriz ordenados por columna.
Esta forma de asignarle valores a un arreglo, nos permite en los casos en que sea posible, asignar valores usando ciclos `DO` implícitos, por ejemplo:

```
V = [(REAL(i**2), i=1,5)]
A = RESHAPE([((REAL(i-j),i=1,5),j=1,4)],[5,4])
```

También es posible si fuera necesario, asignar los valores al momento de declarar los arreglos, por ejemplo:

```
REAL :: V(5) = [1.0, -2.0, -3.0, 4.0, -5.0]
```

Si queremos asignarle a todas las componentes de un arreglo un mismo valor, por ejemplo cero, podemos hacerlo de la siguiente manera:

```
V = 0.0
A = 0.0
```

Los arreglos pueden ser utilizados como un todo en operaciones aritméticas si los arreglos involucrados tienen la misma forma (igual número de dimensiones y de elementos en cada una de ellas). Por ejemplo, si tenemos tres arreglos `X`, `Y`, `Z`, podremos hacer la siguiente operación, en donde al triple de `Y` le restamos `Z` para obtener `X`:

```
X = 3.0 * Y - Z
```

Un par de funciones útiles son `DOT_PRODUCT(X,Y)` que nos devuelve el producto escalar entre los vectores `X` e `Y` y `MATMUL(A,B)` que nos devuelve el producto de las matrices `A` y `B`.

```
P = DOT_PRODUCT(X,Y)
C = MATMUL(A,B)
```

También nos es posible usar sólo parte de un arreglo como mostramos en los siguientes ejemplos usando los arreglos definidos más arriba:

- `V(1:2)` ó `V(:2)` es el subarreglo de `V` de elementos `V(1)` y `V(2)`.
- `V(1:5:2)` ó `V(::2)` es el subarreglo de `V` de elementos con índice impar: `V(1)`, `V(3)` y `V(5)`.
- `A(1:5,3)` ó `A(:,3)` es la tercera columna de la matriz representada por el arreglo `A`.
- `A(2,1:4)` ó `A(2,:)` es la segunda fila de la matriz representada por el arreglo `A`.