Operación “dos puntos” en el lenguaje MATLAB

Objetivos. Conocer la operación *dos puntos* (: , *colon* en inglés) del lenguaje MATLAB y sus aplicaciones principales.

Requisitos. Arreglos en el lenguaje MATLAB, scripts en el lenguaje MATLAB.

La operación “dos puntos” (:) es una de las más útiles y poderosas en el lenguaje MATLAB y en en sus análogos libres GNU Octave, Scilab y FreeMat. En los ejemplos se usa la sintaxis de GNU Octave.

El propósito de la operación : consiste en **crear arreglos de números equidistantes** (en otras palabras, arreglos cuyos elementos forman progresiones aritméticas), también se aplica para organizar ciclos for y para extraer subarreglos o modificar subarreglos.

Crear arreglos de números equidistantes (sucesiones aritméticas)

1. Ejemplos. Se recomienda ejecutar los siguientes comandos para comprender la sintaxis de la operación “dos puntos”.

```
3 : 7
length(3 : 7)
(3 : 7)'
-2 : 15
length(-2 : 15)
3 : 2 : 7
2 : 5 : 30
2 : 5 : 31
7 : 3
length(7 : 3)
7 : -1 : 3
23 : -3 : 4
3.2 : 8
```

2. Ejercicios. Se propone crear los siguientes arreglos usando la operación : :

- `[1, 2, 3, 4, 5, 6, 7]`
- `[-5, -2, 1, 4, 7, 10, 13, 16]`
- `[8, 7, 6, 5, 4, 3, 2, 1]`
- `[14, 12, 10, 8, 6, 4]`

3. Ejemplos: crear arreglos de números reales equidistantes con la función linspace.

```
a = linspace(2, 3, 11)
length(a)
diff(a)
b = linspace(1.3, 3.9, 7)
length(b)
diff(b)
```

4. Ejercicios: crear arreglos de números reales equidistantes con la función linspace. Se propone crear los siguientes arreglos:

- Un arreglo de 7 números reales equidistantes que empiece desde 0 y termine en 1.
- Un arreglo de 15 números reales equidistantes que empiece con 3.2 y termine con 6.8.

Organizar ciclos for

5. Ejemplo: organización de ciclos con la operación : . Se recomienda guardar los siguientes comandos en un archivo `examplefor1.m` y luego ejecutar el comando `examplefor1` en el intérprete. Otra opción es escribir los siguientes comandos directamente en el intérprete.

```
for j = 1 : 5
  disp(j);
endfor
for j = 3 : 5 : 20
  disp(j);
endfor
for k = 8 : -1 : 2
  disp(k);
endfor
```

6. Ejemplo: organizar ciclos for con la función linspace. Cuando la variable del ciclo debe tomar valores reales y se sabe el primer valor, el último y el número de pasos, puede ser más cómoda la función linspace.

```
for x = linspace(3, 4.4, 13),
  y = x * x;
  disp([x, y]);
endfor
```

7. Ejercicios.

- Organizar un ciclo for cuya variable de ciclo tome los valores $1, 2, \ldots, 8$.
- Organizar un ciclo for cuya variable de ciclo tome los valores $14, 11, 8, 5, 2$.
- Organizar un ciclo for cuya variable de ciclo tome los valores $0, 0.2, 0.4, \ldots, 1.4$.

Extrar subarreglos o modificar subarreglos

La operación : es útil para sacar de un arreglo ciertas entradas cuyos índices forman una progresión aritmética.

8. Extracción de subarreglos con el comando : .

```
a =  round(1000 * rand(1, 9) - 500)
a([3, 4, 9])
a([3, 5, 7])
a(3 : 2 : 7)
a(5 : end)
a(end: -1 : 1)
a(:)
```

En el caso de matrices, se pueden sacar submatrices, incluso renglones, subrenglones, columnas y subcolumnas.

9. Extracción de submatrices con el comando : .

```
a = rand(4, 7)
a(1 : 3, 1 : 3)
a(2 : 4, 2 : 3 : 7)
a(4, :)
a(4, 2 : 6)
a(:, 2)
a(2 : end, 5)
```

10. Ejercicios. Componga y resuelva ejercicios que utilicen la sintaxis mostrada en los ejemplos anteriores.