PRACTICO Nº 4

Introducción

El objetivo de este práctico es practicar en la creación e invocación de scripts y funciones en Octave, mostrando las principales diferencias entre ambos.

Ejercicio 1

Considere los siguientes programas:

parte1.m:

```
% script
a=a+2;
b=b+4;
c=a+b;
```

parte2.m:
parte2.m:

```
% funcion
function [a,b]=parte2(x,y,z)
x=(y+z)/2;
c=x+y;
a=c;
b=y+z;
```

principal.m:
principal.m:

```
% principal
a=4;
b=6;
c=2;
parte1;
a=c;
[c,b]=parte2(a,b,c);
```

Si ejecutamos desde la línea de comandos del Octave el script llamado *principal*, ¿cuáles son los valores finales de las variables *a*, *b* y *c*?

Ejercicio 2

Considere los siguientes programas:

medio.m:

```
%calcula pto medio
%delintervalo dado
% en vector x
a=x(1);
b=x(2);
pto = (a+b)/2;
```

extremos.m:
extremos.m:

```
% calcula nuevos
% extremos
a=x(2);
b=x(1);
x(1)=(a+b)/2;
x(2)=4*(a+b)/2;
```

extremosF.m:
extremosF.m:

```
% calcula nuevos
% extremos
function x=extremosF(x)
   a=x(2);
   b=x(1);
   x(1)=(a+b)/2;
   x(2)=4*(a+b)/2;
```

Luego de copiar los programas ejecute desde la línea de comandos las siguientes instrucciones:

```
x=[3,15]
extremos;
medio;
pto
```

y ahora las siguientes:

```
x=[3,15]
extremosF(x);
medio;
pto
```

¿Cómo explica la diferencia encontrada al calcular el punto medio del intervalo?

Ejercicio 3

Parte a

Supongamos que queremos implementar en Octave las funciones hiperbólicas *senh* y *cosh* cuyas fórmulas son: $\text{senh}(x) = \frac{e^x - e^{-x}}{2}$ y $\cosh(x) = \frac{e^x + e^{-x}}{2}$

i. Escribir dos scripts llamados *senh1s* y *cosh1s* que implementen dichas funciones.

ii. Escribir dos funciones llamadas *senh1f* y *cosh1f* que implementen las funciones hiperbólicas.

El valor senh y cosh guárdelo en una variable llamada *y*.

Parte b

Sabemos que la derivada n-ésima de *senh(x)* es *senh(x)* para *n* pares y *cosh(x)* para *n* impares. Utilizando las funciones hiperbólicas predefinidas en Octave:

i. Escriba un script que evalúe la derivada n-ésima (valor dado en la variable *n*) de *senh* para un valor dado en la variable *x*.

ii. Ídem que anterior pero ahora escriba una función.

Parte c

Ahora vamos a implementar las mismas fórmulas de *senh* y *cosh* que en la parte a pero con la siguiente variante. Defina en su programa una variable auxiliar llamada *auxi* que contenga el valor de e^x y luego utilícela para calcular el resultado (de *senh* o *cosh* según corresponda). Recuerde que $e^{-x} = 1/e^x$.

Este cambio realícelo en los scripts y cambie su nombre a *senh2s* y *cosh2s*, y también en las funciones y renómbrelas a *senh2f* y *cosh2f*. El uso de esta variable intermedia podría justificarse si suponemos que el cálculo de la exponencial es mas costoso que el realizar una división (ésto no siempre es cierto).

Parte d

Considere el siguiente script:

```
x=2;
senh1s;
auxi=y;
x=4;
cosh1s;
resultado=y+auxi
```

Ejecútelo en Octave y recuerde su resultado. Ahora sustituya los nombres de los scripts *senh1s* y *cosh1s* por *senh2s* y *cosh2s* respectivamente. Ejecútelo nuevamente y compare con el resultado anterior. Podría explicar, ¿porque difieren los resultados?

Parte e

Consideremos ahora el siguiente script:

```
x=2;
y=senh1f(x);
auxi=y;
x=4;
y=cosh1f(x);
resultado=y+auxi
```

Ejecútelo en Octave y recuerde su resultado. Ahora sustituya los nombres de las funciones *senh1f* y *cosh1f* por *senh2f* y *cosh2f* respectivamente. Ejecútelo nuevamente y compare con el resultado anterior. ¿Podría explicar qué sucedió en este caso?

Ejercicio 4

El programador del siguiente algoritmo cometió un error, identifíquelo y corríjalo. Este algoritmo ordena un vector de números de menor a mayor.

```
% ordena un vector

function x=ordenar(x)
n=length(x); %obtengo largo del vector
for i=1:n-1
   for j=i+1:n
      if x(i)>x(j) %intercambiar valores
         x(j)=x(i);
         x(i)=x(j);
      end
   end
end
```

Ejercicio 5

El siguiente algoritmo no se comporta como está especificado, encuentre el problema y corríjalo.

```
% Función mayor(x) devuelve el mayor de los elementos
% del vector x el cual puede estar desordenado
function maximo = mayor(x)
n=length(x);
maximo=x(1);
for i=2:n
    if x(i) > x(i-1)
        maximo = x(i);
    end
end
```

Ejercicio 6

a) Escriba una función *agregarf* que agregue F nuevas filas a una matriz, rellenas con el número X.
b) Escriba una función *agregarc* que agregue C nuevas columnas a una matriz, rellenas con el número X.
c) Escriba una función *agregarfc* que agregue F nuevas filas y C nuevas columnas a una matriz, rellenas con el número X.

Ejercicio 7

Escriba una función traza que calcule la traza de una matriz de tamaño NxN.
Nota: Se define traza de una matriz cuadrada como la suma de los elementos de su diagonal.

Ejercicio 8

Escriba una función *transconj* que devuelva la transpuesta conjugada de una matriz de tamaño MxN, sin utilizar " ' ".
Sugerencia: utilizar la función de Octave *imag* para averiguar si un número es complejo.

Ejercicio 9

Se dispone de una función *en_intervalo* que recibe un número **valor** y dos números **inicio** y **fin**, devolviendo 1 si valor pertenece al intervalo [**inicio**,**fin**] y 0 en caso contrario.
Escriba una función *cantidad_en_intervalo* que reciba un vector **v_in** y dos números **inicio** y **fin**, y devuelva la cantidad de elementos de **v_in**, comprendidos en el intervalo [**inicio**,**fin**].
Sugerencia: Utilice la función *en_intervalo* dentro de la función *cantidad_en_intervalo*.

Ejercicio 10

Se dispone de una función *es_primo* que dado un entero devuelve 1 si el mismo es primo y 0 en caso contrario.

a) Escribir una función *listar_primos* que dados dos valores **a** y **b** devuelva un vector conteniendo los números primos comprendidos en el intervalo [**a**,**b**].

b) Escribir una función *expresar_como_suma* que dado un número entero **x** devuelva dos enteros primos **a** y **b** tal que x = a+b. De no existir a y b que cumplan tal propiedad, la función debe devolver a y b en -1.

Nota: Para la implementación en Octave se sugiere usar la función *isprime* que ya viene incluida en Octave.

Ejercicio 11

Escriba una función *pertenece* que encuentre el número X en una matriz de MxN elementos. En caso de encontrarlo, la función debe retornar dos valores con la posición i, j de la primera aparición del elemento en la matriz. En caso de no encontrarlo, la función debe retornar ambos valores en cero.

Nota: la búsqueda se debe realizar por filas.

Ejercicio 12

Escriba una función *repeticiones* que retorne las posiciones (i, j) de todas las apariciones del número X en una matriz de MxN elementos. La función debe retornar una matriz de 2 columnas y tantas filas como veces aparece X en la matriz. Cada fila debe contener la posición (i, j) de cada aparición. Si X no pertenece a la matriz, la función debe retornar el vector vacío.

Ejercicio 13

Escriba una función *extraer_tridiagonal* que extraiga la tridiagonal de una matriz M, no necesariamente cuadrada, pasada como parámetro. La tridiagonal de M posee únicamente los elementos de su diagonal principal, de la primera diagonal debajo de ésta y de la primera diagonal que se encuentra por encima de la diagonal principal; las demás posiciones tienen valor 0.

Por ejemplo:

```
>> M = [1 4 6 5; 3 4 1 9; 8 2 3 4; 9 1 1 3];
>> D = extraer_tridiagonal(M);
```

Resultado:

```
D = [1 4 0 0; 3 4 1 0; 0 2 3 4; 0 0 1 3];
```

Ejercicio 14

Escriba una función ***suma_por_filas*** que tome como entrada una matriz de tamaño MxN y devuelva un vector de M elementos, donde el elemento en la posición i del vector sea la suma de las celdas de la fila i de la matriz.

Ejemplos:

`v=suma_por_filas([4,1,3; 2,-1,-1; -3,1,-2])` devolvería `v=[8,0,-4]`

`v=suma_por_filas([-1,0,-2; 3,0,0])` devolvería `v=[-3,3]`

Ejercicio 15

Escribir la función ***reverso*** que toma como parámetro un vector v y devuelve un vector con los mismos elementos pero en orden inverso.

Ejemplo:

```
>> w = reverso([2 5 6 1 3 7])
w = [7 3 1 6 5 2]
```

Ejercicio 16

Escribir la función ***rotar*** que toma como parámetro una matriz M, no necesariamente cuadrada, y devuelve una matriz con sus mismos elementos rotados 180º.
Ejemplo:

```
>> r = rotar([9,6,3; 2,1,4; 5,7,8])
r = [8,7,5; 4,1,2; 3,6,9];
```

Ejercicio 17

Hay una sola opción correcta por cada pregunta

1. Indique cuál de las siguientes afirmaciones es verdadera:
   a. en un *script* el alcance de las variables es local, mientras que en las funciones es global
   b. tanto los *scripts* como las funciones tienen alcance global para sus variables
   c. en un *script* el alcance de las variables es global, mientras que en las funciones es local
   d. Ninguna respuesta es verdadera

2. ¿Cuál es el resultado del *script* de *Octave* del recuadro?
   a. x = 1, y = 5
   b. da error, porque está mal usada la sentencia de asignación
   c. x = 5, y = 4
   d. Ninguna respuesta es verdadera

```
y = 1;
x = 4;
y + 1 = x;
```

3. Dada la función del recuadro, indique qué algoritmo implementa (asumir que b es un número natural):
   a. `a*b`
   b. `a*a`
   c. `a/b`
   d. Ninguno de los anteriores

```
function y = incognita(a,b)
y = 0;
for i=1:b
    y = y + a;
end
```