Módulo 3: Lógica y depuración

EX 1. Rotar un vector K veces hacia la derecha

```
H = 1 : 1 : 10
```

```
H = 1×10
     1     2     3     4     5     6     7     8     9    10
```

```
N = length(H);
i = 1;
K = 3;
moves = 0;
while (moves < K) % Sólo moverse K veces
    A = H(end); % Guardar el último valor
    for (i = N : -1 : 2)
        H(i) = H(i-1); % Mover todo una posición hacia la derecha
    end
    H(1) = A; % En la primera posición poner lo que antes había en la última
    moves = moves + 1;
end
H % Mostrar resultado
```

```
H = 1×10
     8     9    10     1     2     3     4     5     6     7
```

Ex 2. Mezclar dos vectores en uno intercalando posiciones.

```
clear; clc;
% Definir los dos vectores a mezclar:
H = 1 : 1 : 5
```

```
H = 1×5
     1     2     3     4     5
```

```
M = 31 : 1 : 35
```

```
M = 1×5
    31    32    33    34    35
```

```


fin_size = length(H) + length(M); % El tamaño que tendrá el vector de salida
S = zeros( 1, fin_size ); % Vector de salida inicialmente lleno de ceros
r = 0;
pares_cont = 0;
impares_cont = 0;

for ( i = 1  : length(S) )
    if ( mod(r, 2) == 0 ) % Cada vez par tomo un valor del vector H
        pares_cont = pares_cont + 1; % Indica la posición que debo tomar de H
```

```
        S(i : i ) = H( pares_cont : pares_cont ); %Poner valor en el vector de salida
    else                    % Cada vez impar tomo un valor del vector M
        impares_cont = impares_cont + 1; % Indica la posición que debo tomar de M
        S(i : i ) = M(impares_cont : impares_cont ); %Poner valor en el vector de salid
    end
    r = r + 1;
end

S % Muestre el resultado
```

```
S = 1×10
     1    31     2    32     3    33     4    34     5    35
```

Ex 3. Determine si, dadas dos palabras, una de ellas es anagrama de la otra.

```
clear; clc;
% Palabras a comparar:
palabra1 = 'ballenas';
palabra2 = 'llenabas';
tachadas = zeros(length(palabra2), 1); % Este vector tiene cero si cierta letra no se h
                                       % encontrado en la palabra 2, y
                                       % tiene 1 si ya se encontró (se van
                                       % 'tachando' las letras ya
                                       % encontradas.

if (length(palabra1) == length(palabra2)) % Primero verificar que ambas palabras tengan
                                          % mismo tamaño, sino, no tendría
                                          % sentido verificar si son
                                          % anagramas o no
    for i = 1 : length(palabra1)  % Recorrer cada letra de la palabra 1
        letra_i = palabra1(i);
        for j = 1 : length(palabra2) % Para cada letra de palabra 1, buscar en cada
            letra_j = palabra2(j);   % letra de la palabra 2
            if (letra_i == letra_j) && ( tachadas(j) == 0) % Si se encutra la letra en
                tachadas(j) = 1;                           % la segunda palabra
                                                           % entonces marcarla en el
                                                           % vector 'tachadas'

            end
        end
    end
end
k = 1;

while ((k <= length(tachadas))&&(tachadas(k) == 1))
    k = k +1; % Contar la cantidad de letras 'tachadas'
end
es_anagrama = 0; % La variable es_anagrama tendrá 0 al final si las palabras no son
                 % anagramas o tendrá 1 en caso de que sí sean anagramas
if (k == length(tachadas)+1) % Si todas las letras se tacharon, entonces las palabras
    es_anagrama = 1;         % sí son anagramas
```

```
    disp([palabra1, ' y ', palabra2, ' son anagramas'])
else
    disp([palabra1, ' y ', palabra2, ' no son anagramas'])
end
```

```
ballenas y llenabas son anagramas
```

Ex 4. Extraer los subarreglos que contienen valores positivos del arreglo de entrada.

```
clear; clc;
R = [-10, 1, 7, 8, -5, -1, 3, 2, -2, 0, -8, -4, 1]
```

```
R = 1×13
   -10     1     7     8    -5    -1     3     2    -2     0    -8    -4     1
```

```
S = [];


i = 1;
while ( i <= length(R) )

    while (( i <= length(R) ) && ( R(i) >= 0 )) % S
        S = [S, R(i)]; % Agregar el elemento sólo si es positivo
        i = i + 1;
    end

    if ( i <= length(R) ) % Salgar i cuando se encuentren negativos y no se supere aún
        if (  R(i) < 0  ) % el tamaño de R
            i = i + 1;
        end
    end
end
S
```

```
S = 1×7
     1     7     8     3     2     0     1
```

Ex 5. Busque en una matriz todos los valores que estén entre K-L y K+L, recorriendo la matriz en orden derecha-izquierda, arriba-abajo.

```
clear; clc;
K = 30
```

```
K = 30
```

```
L = 15
```

```
L = 15
```

```
N = 10;
Y = magic(N);

R = []
```

```
R =

     []
```

```
for (i = 1 : N)
    for (j = 1 : N)
        if ( ( Y( i, j ) >= K - L ) && ( Y( i, j ) <= K + L ) )
            R = [ R, Y( i, j ) ];
            % Agregar el elemento sólo si está dentro del rango dado
        end
    end
end

R
```

```
R = 1×31
    15    40    16    41    20    22    19    21    28    25    34    17    24 ···
```

Ex 6. Dibujar un patrón diagonal en cualquier dirección de la matriz.

```
clear; clc;
N = 10;
P = zeros( N );

f_init = 2;
c_init = 5;
r = 1;

% Dirección hacia la que se hará el recorrido:
dir_vert = 1;
dir_horz = 1;

for (repetics = 1 : 5) % Cantidad de caminos a dibujar

    while ( ( f_init >= 1 ) && ( f_init <= N ) && ( c_init >= 1 ) && ( c_init <= N ) )
        % Verificar que no se supere el rango de filas o columnas
        P(f_init, c_init) = r;
        r = r + 1;
        f_init = f_init + dir_vert;
        c_init = c_init + dir_horz;
    end
    % Para cambiar de dirección al chocar con algún límite:
    if ( ( dir_vert == 1 ) && ( dir_horz == 1 ) )
```

```
        c_init = c_init - 2*dir_horz;
        dir_horz = -1;
    elseif ( ( dir_vert == 1 ) && ( dir_horz == -1 ) )
        f_init = f_init - 2*dir_vert;
        dir_vert = -1;
    elseif ( ( dir_vert == -1 ) && ( dir_horz == -1 ) )
        c_init = c_init - 2*dir_horz;
        dir_horz = 1;
    elseif ( ( dir_vert == -1 ) && ( dir_horz == 1 ) )
        f_init = f_init - 2*dir_vert;
        dir_vert = 1;
    end

end

P
```

```
P = 10×10
     0     0     0    18     0     0     0     0     0     0
     0     0    17     0    19     0     0     0     0     0
     0    16     0     0     0    20     0     0     0     0
    15     0     0     0     0     0    21     0     0     0
     0    14     0     0     0     0     0    22     0     0
     0     0    13     0     0     0     0     0    23     0
     0     0     0    12     0     0     0     0     0    24
     0     0     0     0    11     0     0     0     7     0
     0     0     0     0     0    10     0     8     0     0
     0     0     0     0     0     0     9     0     0     0
```