[[1]](#footnote-1) I. INTRODUCCIÓN.

Tarea 2 – Pilas y colas.

Hernández Varela Ricardo

En esta práctica se desarrollan funciones para implementar soluciones a problemas procesando cadenas en pilas y colas, en ambos casos se realizan operaciones de inserción y extracción de elementos aplicando las propiedades de estas estructuras en objetos.

Para el caso de las colas se realizan operaciones de encolar objetos para insertar elementos al final de la cola y operaciones de desencolar para extraer elementos del frente de la cola, al trabajar con pilas se realizan operaciones de empujar elementos uno sobre otro y extraer el último elemento de la pila.

Con todas las funciones desarrolladas se realizan pruebas con diferentes entradas y después de realizar la llamada a la función se imprimen los resultados para evaluar el desempeño de las funciones.

# II. Ejercicio 1.1

1. Implementar una función en Matlab que, daba una cadena, ponga la primera letra de cada palabra de la cadena en mayúsculas. Utiliza un apuntador para recorrer la cadena. Procesar al menos 5 cadenas.

Implementación en Matlab.

Para cambiar la primera letra de cada palabra en una frase a letras mayúsculas, se crea una copia de la cadena original.

Es necesaria una comprobación del primer carácter, si es un espacio, el siguiente carácter se cambia a mayúsculas con la función upper() de Matlab, sino el primero es el que se cambia.

Se utiliza un ciclo **for** para recorrer la cadena para encontrar la primer letra de cada palabra, en una variable se almacena el carácter actual en cada iteración, cuando este es igual a un espacio, el carácter se cambia a mayúscula.

Código en Matlab.

function [copia] = mayusculas(cadena)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para cambiar la primera letra de cada palabra en una frase a

%letras mayúsculas.

%Creación de una copia de la cadena original.

copia = repmat(cadena,1);

%Comprobación del primer carácter. Si es un espacio el siguiente carácter

%se cambia a mayúsculas con la función upper(), sino el primero es el que se cambia.

if copia(1) == ' '

copia(2) = upper(cadena(2));

else

copia(1) = upper(cadena(1));

end

%Con un ciclo for se recorre la cadena para encontrar la primer letra de

%cada palabra, en una variable se almacena el carácter actual en cada

%iteración, cuando este es igual a un espacio, el carácter se cambia a

%mayúscula.

for i=2:length(copia)

caracter = copia(i);

if caracter == ' '

copia(i+1) = upper(copia(i+1));

else

copia(i) = caracter;

end

end

En el script **ejercicio1** se programa una variable en la cual se almacena una cadena introducida por el usuario y la cual sirve como parámetro de entrada para la función **mayusculas**() con la que la primera letra de cada palabra en la cadena se convierte en letras mayúsculas.

La cadena que ingresa el usuario se almacena en una variable de tipo carácter.

La nueva cadena se almacena en una nueva variable y finalmente se realiza la impresión de la frase original y la frase con la primera letra de cada palabra en mayúsculas.

Código en Matlab.

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Prueba de la función para cambiar la primera letra de cada palabra en una frase a

%letras mayúsculas.

%La cadena que ingresa el usuario se almacena en una variable de tipo carácter.

cadena = input('Escribe una cadena: ', 's');

fprintf('\n');

%Se realiza una llamada a la función mayusculas(), el parámetro de entrada

%es la cadena de la cual se cambiará la primera letra a mayúsculas. La

%nueva cadena se almacena en una nueva variable.

nueva\_cadena = mayusculas(cadena);

%Impresión de la frase original y la frase con la primera letra de cada

%palabra en mayúsculas.

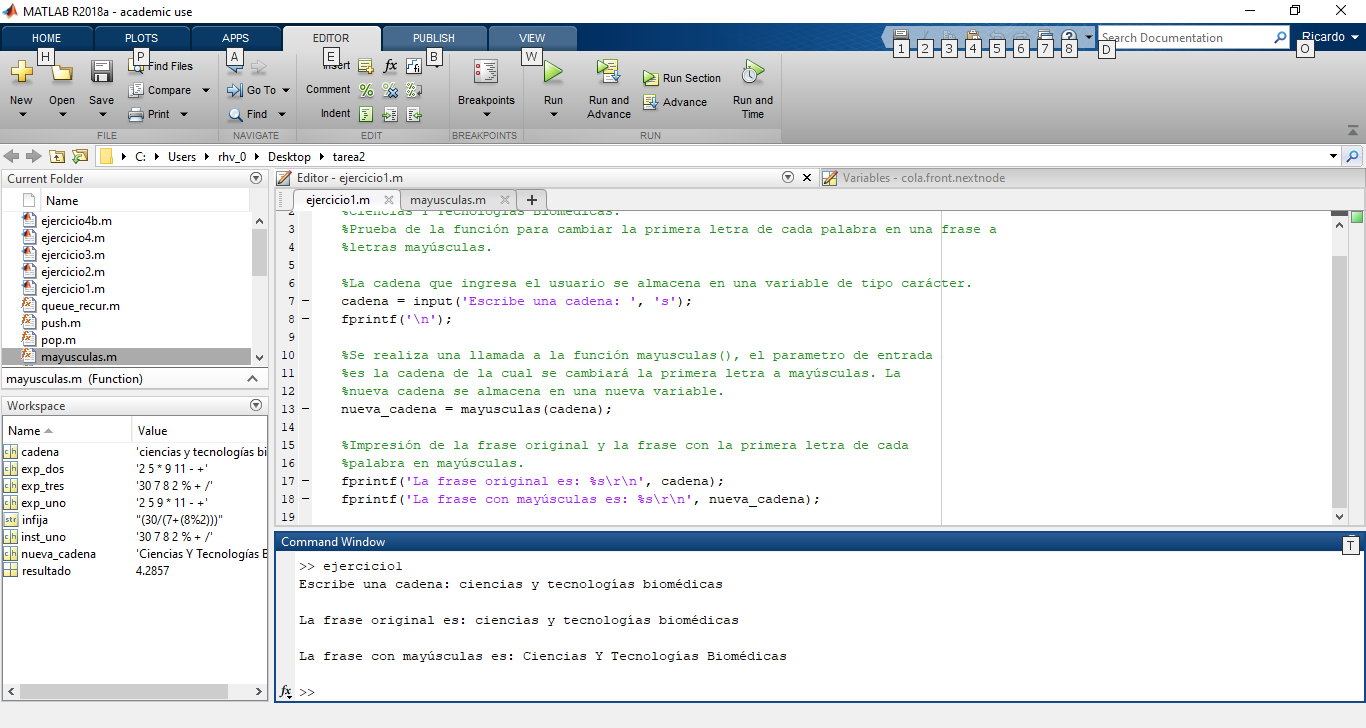
fprintf('La frase original es: %s\r\n', cadena);

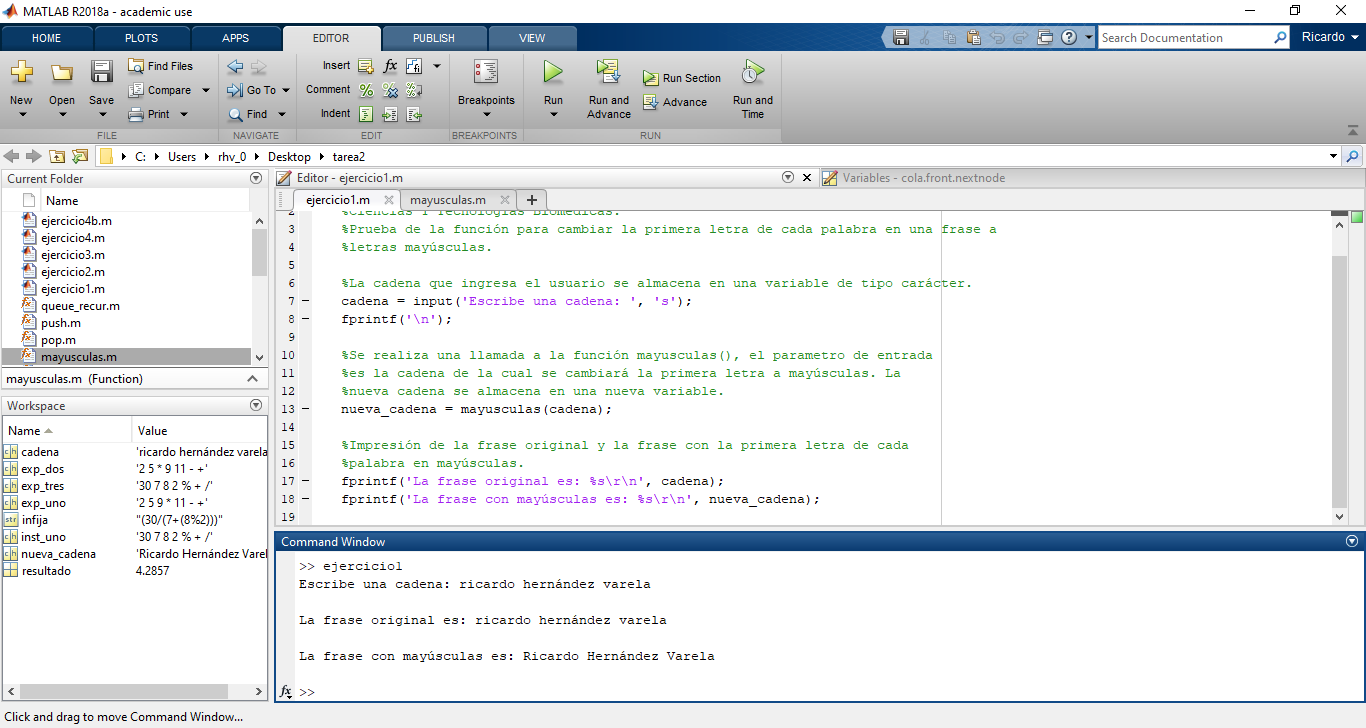
fprintf('La frase con mayúsculas es: %s\r\n', nueva\_cadena);

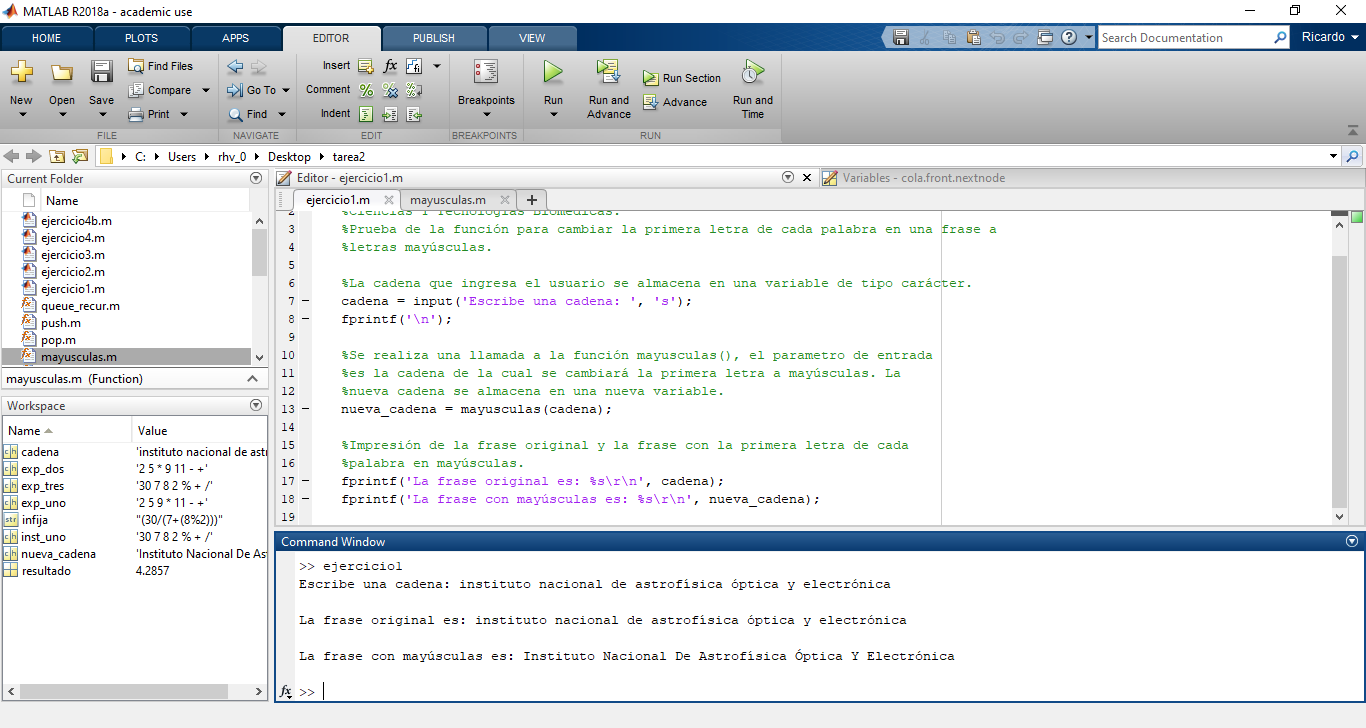
Resultados de la prueba de la función **mayusculas**().

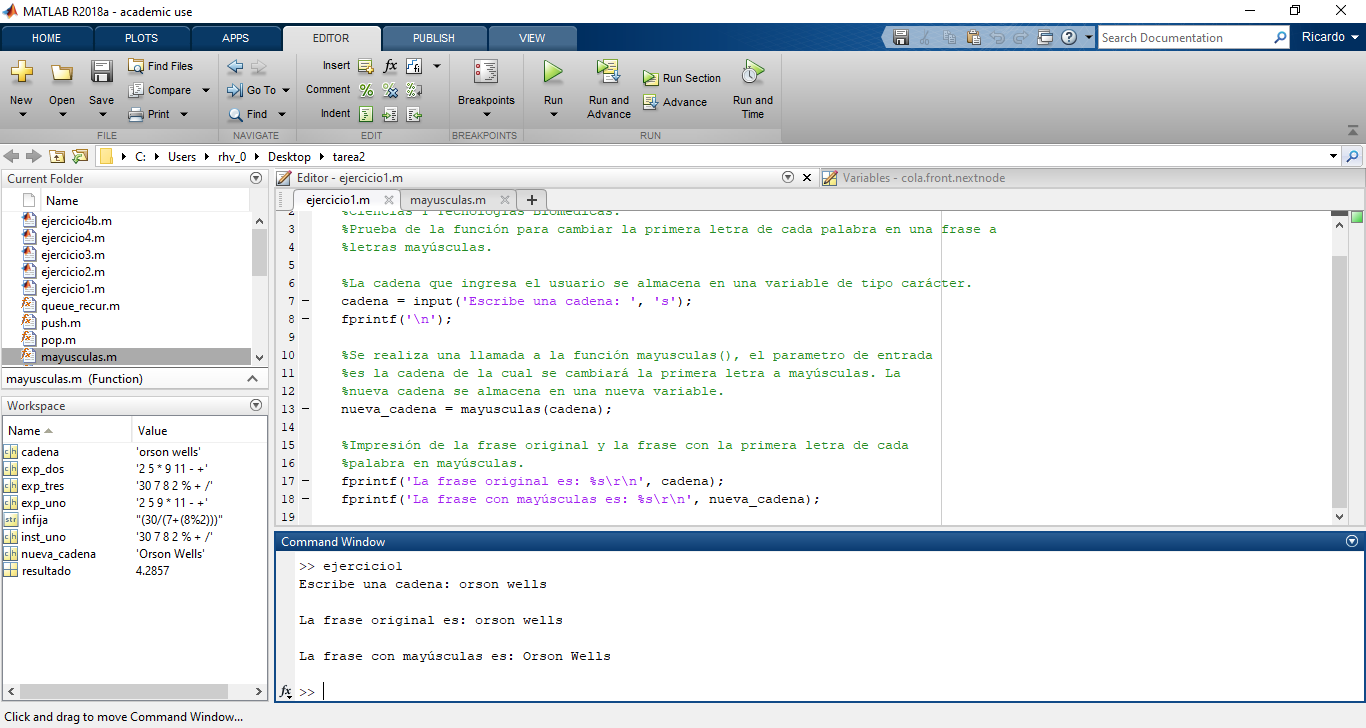
Cadenas procesadas:

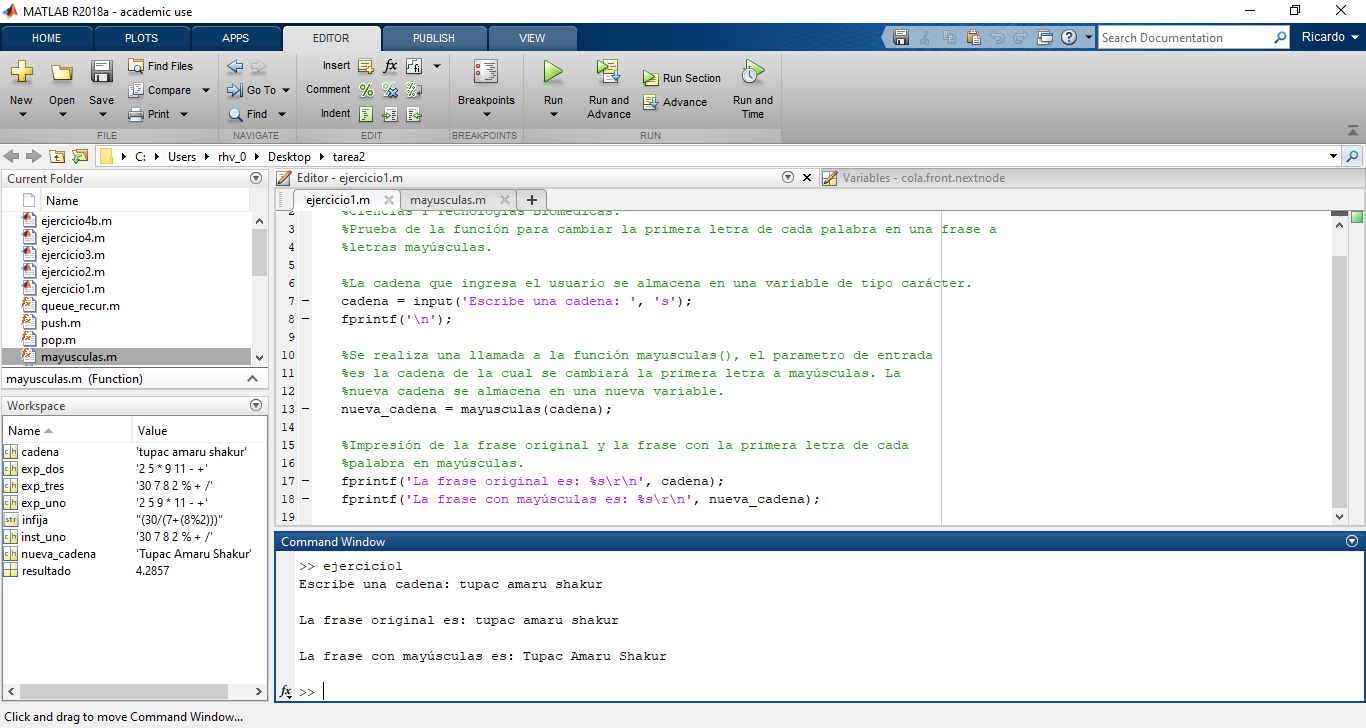
1. ciencias y tecnologías biomédicas
2. ricardo hernández varela
3. instituto nacional de astrofísica óptica y electrónica
4. orson wells
5. tupac amaru shakur











1. Implementa las funciones en Matlab necesarias para procesar la siguiente lista de instrucciones efectuadas sobre una pila inicialmente vacía, donde una letra es una operación de inserción y un asterisco es una operación de extracción, indica cual es el estado final de la pila.

Para implementar la solución de este problema se utilizan dos tipos de objeto, el nodo para crear un nuevo elemento que apilar y la pila para almacenar este nodo. En una pila los elementos nuevos ingresan por el tope y solo puede extraerse el último valor de la pila.

Definición de la clase **nodo**.

Código en Matlab.

classdef node

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Definición de un objeto tipo nodo.

%Las propiedades de este objeto son el id o valor del nodo, y el apuntador

%al nodo siguiente.

properties

id

nextnode

end

methods

function obj = node(id\_)

obj.id = id\_;

end

end

end

Las propiedades de este objeto son el id o valor del nodo, y el apuntador al nodo siguiente. Al crear un nodo nuevo se define como parámetro el valor que almacenará, la propiedad **id** permite tener acceso a este valor. La propiedad **nextnode** es un apuntador al siguiente elemento de la pila.

Definición de la clase **stack**.

Código en Matlab.

classdef stack

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Definición de un objeto tipo pila.

%Las propiedades de este objeto son el tope, tamaño actual y tamaño máximo

%de la pila.

properties

top

size

max\_size

end

methods

function obj = stack(max\_tam\_)

obj.max\_size = max\_tam\_;

obj.size = 0;

end

end

end

Las propiedades de este objeto son el tope, tamaño actual y tamaño máximo de la pila.

Para insertar un nuevo elemento en la pila se utiliza la función **push()**. Con un condicional se verifica si la pila está llena, si está llena no se almacena nada en la pila, sino se genera un nuevo nodo con el valor de entrada, el apuntador hacía el elemento siguiente, se inserta el nuevo elemento al tope de la pila y el tamaño de la pila se incrementa en una unidad.

Código en Matlab.

function stack = push(stack, value\_)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para insertar un elemento en la pila.

%Con un condicional se verifica si la pila está llena.

if isFull(stack)

disp('Push: Full stack'); %Si está llena no se almacena nada en la pila.

else

n = node( value\_ ); %Si no está llena se genera un nuevo nodo con el valor de entrada.

n.nextnode = stack.top ; %Se crea el apuntador hacía el elemento siguiente.

stack.top = n; %Se inserta el nuevo elemento al tope de la pila.

stack.size = stack.size+1; %Se incrementa el tamaño de la pila en una unidad.

end

end

La función **isFull()** se utiliza para verificar si una pila o cola está llena, si la pila o cola está llena se retorna el valor verdadero, si la pila o cola no está llena se retorna el valor falso.

Código en Matlab.

function value = isFull(stack)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para verificar si una pila o cola está llena.

if stack.size == stack.max\_size

value = true; %Si la pila o cola está llena se retorna el valor verdadero.

else

value = false; %Si la pila o cola no está llena se retorna el valor falso.

end

end

Para extraer el elemento del tope de la pila se utiliza la función **pop()**, con un condicional se verifica si la pila está vacía. Cuando está vacía no se puede extraer nada, si no está vacía el tope de la pila se asigna al elemento siguiente del que fue extraído y el tamaño de la pila disminuye en una unidad.

Código en Matlab.

function [n, stack] = pop(stack)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para extraer el elemento del tope de la pila.

%Con un condicional se verifica si la pila está vacía.

if isEmpty(stack)

disp('Pop: Empty stack') ; %Si está vacía no se puede extraer nada.

n = [];

else

n = stack.top.id; %Si no está vacía se puede extraer el tope de la pila.

stack.top = stack.top.nextnode; %El tope de la pila se asigna al elemento siguiente del que fue extraído.

stack.size = stack.size-1; %El tamaño de la pila disminuye en una unidad.

end

end

La función **isEmpty ()** se utiliza para verificar si una pila o cola está vacía, Si la pila o cola está vacía se retorna el valor verdadero, si la pila o cola no está vacía se retorna el valor falso.

Código en Matlab.

function value = isEmpty(stack)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para verificar si una pila o cola está vacía.

if stack.size == 0

value = true; %Si la pila o cola está vacía se retorna el valor verdadero.

else

value = false; %Si la pila o cola no está vacía se retorna el valor falso.

end

end

Para procesar las listas de instrucciones para insertar o extraer elementos de una pila la función **push()** se ejecuta cuando en la lista de instrucciones se encuentra una letra y **pop**() se ejecuta cuando en la lista se encuentra un carácter específico.

La función **apilar()** se utiliza para procesar una lista de instrucciones sobre una pila inicialmente vacía, al principio se crea una pila con un tamaño máximo introducido como parámetro, con un ciclo **for** se recorre la lista de instrucciones, en una variable se almacena el valor de la cadena correspondiente a cada iteración y con un condicional se verifica si el valor se encuentra dentro de los caracteres indicados en el parámetro **toPop**, si se cumple la condición se saca un elemento de la pila, sino ese valor se introduce en la pila.

Con un ciclo for se recorren los valores almacenados en la pila para incluirlos en la cadena del estado final de la pila. En cada iteración se accesa al valor al que apunta cada elemento de la pila para extraerlo. Para mostrar los elementos en el orden en que entraron a la pila se invierte el orden de la cadena del estado final de la pila.

Código en Matlab.

function resultado = apilar(max, cadena, toPop)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para procesar una lista de instrucciones sobre una pila

%inicialmente vacía.

%Creación de la pila, el tamaño máximo de la pila se define con el

%parámetro max.

pila = stack(max);

%La lista de instrucciones es la cadena que se introduce como parámetro.

inst\_uno = cadena;

%La función retorna los valores almacenados en la pila en una variable tipo

%string.

resultado = "";

%Con un ciclo for se recorre la lista de instrucciones, en una variable se

%almacena el valor de la cadena correspondiente a cada iteración.

for i=1:length(inst\_uno)

caracter = inst\_uno(i);

%Con un condicional se verifica si el valor se encuentra dentro de los caracteres

%indicados en el parámetro toPop, si se cumple la condición se saca un

%elemento de la pila, sino ese valor se introduce en la pila.

if contains(toPop, caracter)

[~, pila] = pop(pila); %La función pop() saca el elemento del tope de la pila.

else

pila = push(pila, inst\_uno(i)); %La función push() introduce un elemento al tope de la fila.

end

end

%El valor del tope de la pila se incluye en la cadena del estado final de la pila.

resultado = resultado + pila.top.id;

caracter = pila.top.nextnode;

%Con un ciclo for se recorren los valores almacenados en la pila para

%incluirlos en la cadena del estado final de la pila. En cada iteración se

%accesa al valor al que apunta cada elemento de la pila para extraerlo.

for i=2:pila.size

valor = caracter.id;

resultado = resultado + valor;

caracter = caracter.nextnode;

end

%Para mostrar los elementos en el orden en que entraron a la pila se

%invierte el orden de la cadena del estado final de la pila.

resultado = reverse(resultado);

end

En el script **ejercicio2** se almacenan las dos listas de instrucciones y para cada una se realiza una llamada a la función **apilar**().

Código en Matlab.

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Prueba de la función apilar() con dos listas de instrucciones diferentes.

%Primera lista de instrucciones.

inst\_uno = 'EJE\*R\*CICI\*\*\*O\*F\*ACIL\*';

%Llamada a la función apilar() con los parámetros de entrada 20 para el

%tamaño de la pila, la lista de instrucciones a procesar y el carácter para

%la operación de extracción. El valor que retorna la función se almacena en

%una variable.

resultado1 = apilar(20, inst\_uno, '\*');

%Impresión del estado final de la pila.

fprintf('El estado final de la pila es: %s\r\n', resultado1);

%Segunda lista de instrucciones.

inst\_dos = 'LA\*STI\*N\*FIR\*ST\*\*O\*F\*ACIL';

resultado2 = apilar(20, inst\_dos, '\*');

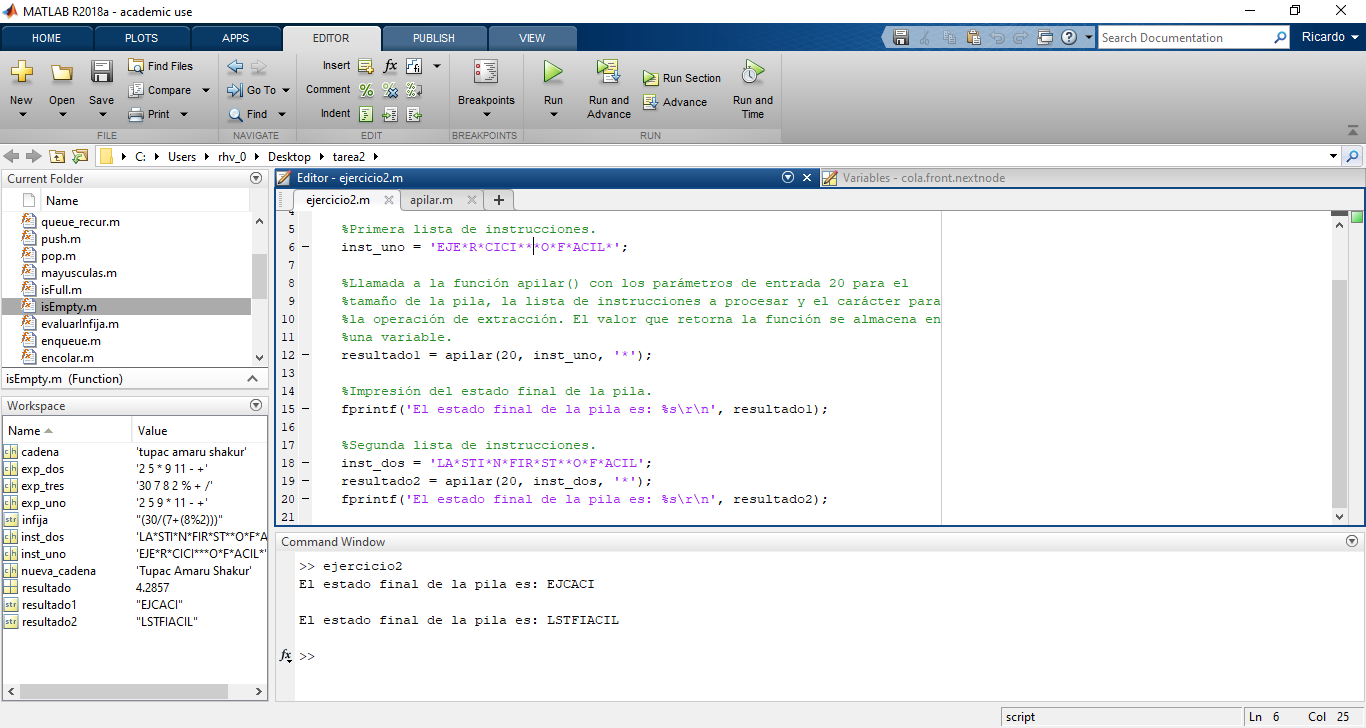
fprintf('El estado final de la pila es: %s\r\n', resultado2);

Resultados de la prueba de la función **apilar**().

Listas de indicaciones procesadas:

a) EJE\*R\*CICI\*\*\*O\*F\*ACIL\*

b) LA\*STI\*N\*FIR\*ST\*\*OU\*T\*\*\*\*\*\*



1. Implementa una función en Matlab que convierta las siguientes expresiones a notación infija y las evalúe:
2. 2 5 9 \* 11 - +
3. 2 5 \* 9 11 - +
4. 30 7 % 8 + 2 /
5. 30 7 8 2 % + /

La función **evaluarInfija()** convierte expresiones en notación posfija a notación infija y evalúa las operaciones para calcular el resultado.

Código en Matlab.

function [infija, resultado] = evaluarInfija(inst\_uno, tam1, tam2)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para convertir expresiones en notación posfija a notación infija y

%evaluar las operaciones para calcular el resultado.

separado = split(string(inst\_uno)); %Separación de los caracteres dentro de la cadena.

operadores = '%/\*-+'; %Carácteres para realizar las operaciones con los elementod de la pila.

infija = ""; %Variabe en la que se almacenan la expresión en notación infija.

pila = stack(tam1); %Creación del primer objeto tipo pila.

pila2 = stack(tam2); %Creación del segundo objeto tipo pila.

%Ciclo for para recorrer la pila que se utiliza para evaluar la expresión.

for i=1:length(separado)

valor = separado(i); %En una variable se almacena el carácter actual.

switch valor %Con un switch se verifica si el carácter es un operador.

case '-'

[n1, pila] = pop(pila); %Si es un operaor de resta se sacan los dos últimos elementos y se restan.

[n2, pila] = pop(pila);

resultado = n2-n1;

pila = push(pila, resultado); %El resultado se inserta en la pila.

case '\*'

[n1, pila] = pop(pila); %Si es un operador de multiplicación se sacan los dos últimos elementos y se multiplican.

[n2, pila] = pop(pila);

resultado = n2\*n1;

pila = push(pila, resultado); %El resultado se inserta en la pila.

case '+'

[n1, pila] = pop(pila); %Si es un operador de suma se sacan los dos últimos elementos y se suman.

[n2, pila] = pop(pila);

resultado = n2+n1;

pila = push(pila, resultado); %El resultado se inserta en la pila.

case '/'

[n1, pila] = pop(pila); %Si es un operador de división se sacan los dos últimos elementos y se dividen.

[n2, pila] = pop(pila);

resultado = n2/n1;

pila = push(pila, resultado); %El resultado se inserta en la pila.

case '%'

[n1, pila] = pop(pila); %Si es un operador de módulo se sacan los dos últimos elementos y se calcula el módulo.

[n2, pila] = pop(pila);

resultado = mod(n2,n1);

pila = push(pila, resultado); %El resultado se inserta en la pila.

otherwise

pila = push(pila, str2double(valor)); %Si el carácter es un número se inserta en la pila.

end

if contains(operadores, valor) %Con un condicional if se verifica si el carácter es un operador.

[n1, pila2] = pop(pila2); %Si lo es, se sacan los dos últimos elementos de la segunda pila.

[n2, pila2] = pop(pila2);

infija = '(' + n2 + valor + n1 + ')'; %Se construye una cadena con los números y el operador correspondiente.

pila2 = push(pila2, infija); %Se introduce en la pila el resultado de concatenar la cadena.

else

pila2 = push(pila2, valor); %Si el carácter es un número se inserta en la pila.

end

resultado = pila.top.id; %El resultado de evaluar la operación se retorna.

end

Ciclo for para recorrer la pila que se utiliza para evaluar la expresión y en una variable se almacena el carácter actual, con un switch se verifica si el carácter es un operador, si es un operador se sacan los dos últimos elementos y se realiza la operación correspondiente, si el carácter es un número se inserta en la pila.

Con un condicional if se verifica si el carácter es un operador, si lo es, se sacan los dos últimos elementos de la segunda pila y se construye una cadena con los números y el operador correspondiente y el resultado de concatenar la cadena se introduce en la pila. El resultado de evaluar la operación se retorna.

En el script **ejercicio3** se almacenan las cuatro expresiones en notación posfija y para cada una se realiza una llamada a la función **evaluarInfija().**

Código en Matlab.

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Expresión en notación posfija.

exp\_uno = '2 5 9 \* 11 - +';

%Llamado a la función evaluarInfija(), indicando la expresión en notación

%posfija y el tamaño para la pila que se utiliza para evaluar la expresión

%y la segunda pila para convertir la expresión a notación infija.

[infija, resultado] = evaluarInfija(exp\_uno, 20, 20);

%Impresión de la expresión en notación infija y el resultado de evaluar la expresión.

fprintf('La notación infija de la expresión "%s" es %s\r\n y el resultado es: %0.2f\r\n', exp\_uno, infija, resultado);

%Expresión en notación posfija.

exp\_dos = '2 5 \* 9 11 - +';

[infija, resultado] = evaluarInfija(exp\_dos, 20, 20);

fprintf('La notación infija de la expresión "%s" es %s\r\n y el resultado es: %0.2f\r\n', exp\_dos, infija, resultado);

%Expresión en notación posfija.

exp\_tres = '30 7 % 8 + 2 /';

[infija, resultado] = evaluarInfija(exp\_tres, 20, 20);

fprintf('La notación infija de la expresión "%s" es %s\r\n y el resultado es: %0.2f\r\n', exp\_tres, infija, resultado);

%Expresión en notación posfija.

exp\_tres = '30 7 8 2 % + /';

[infija, resultado] = evaluarInfija(exp\_tres, 20, 20);

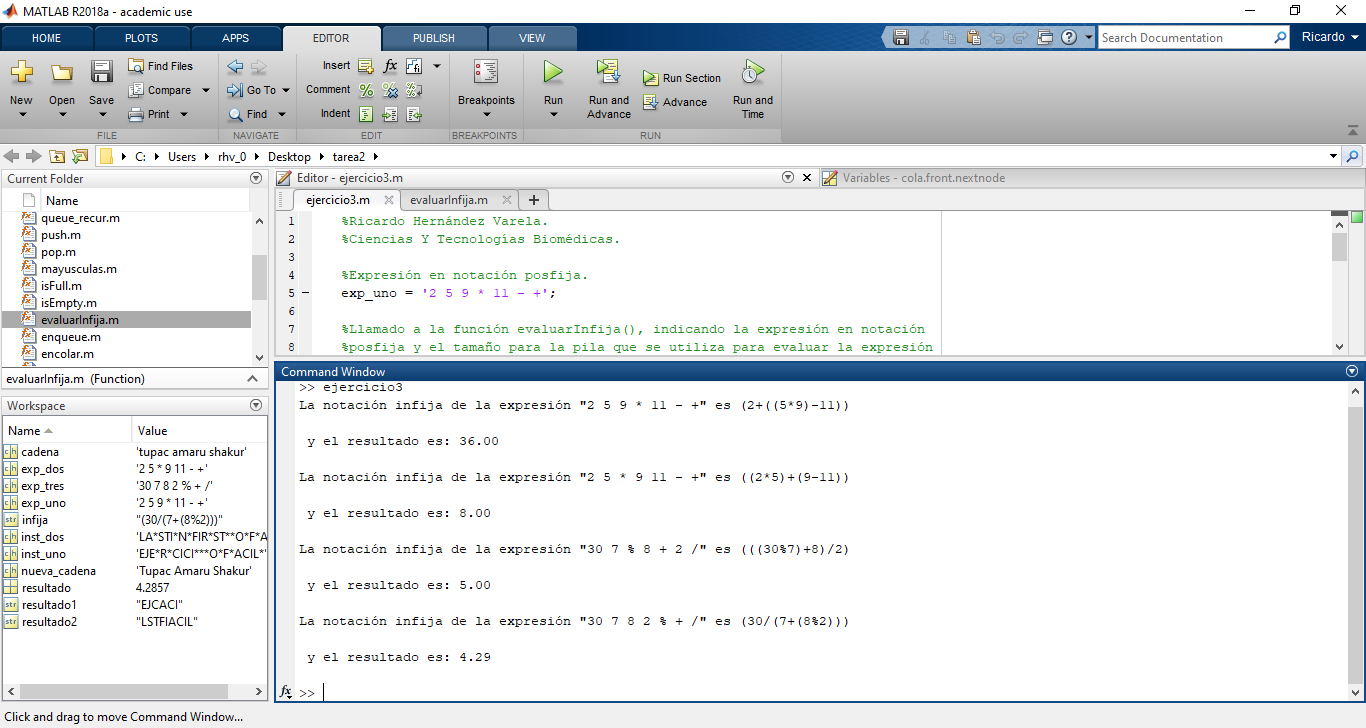
fprintf('La notación infija de la expresión "%s" es %s\r\n y el resultado es: %0.2f\r\n', exp\_tres, infija, resultado);

Finalmente se realiza la impresión de la expresión en notación infija y el resultado de evaluar la expresión.

Resultados de la prueba de la función **evaluarInfija().**

Listas de indicaciones procesadas:

1. 2 5 9 \* 11 - +
2. 2 5 \* 9 11 - +
3. 30 7 % 8 + 2 /
4. 30 7 8 2 % + /



1. Asumiendo que la cola **q** está vacía inicialmente, dibuja su estado después de ejecutar las siguientes operaciones. Realiza dicho estudio para cada una de las dos implementaciones vistas en clase (vectores y nodos enlazados). En el caso de la implementación vectorial, asume que el tamaño del vector es de 8 elementos. Muestra también el valor de la variables x, y, z.

encolar(q,10); desencolar(q);

encolar(q,20); encolar(q,50);

encolar(q,30); encolar(q,60);

x=cabeza(q); x=cabeza(q);

desencolar(q); desencolar(q);

y=cabeza(q); encolar(q,x);

desencolar(q); encolar(q,y);

encolar(q,40);

z=cabeza(q);

**Implementación con vectores.**

Para implementar la solución de este problema utilizando vectores se utilizan tres funciones:

* **encolar**(), introduce un elemento al final de la cola.
* **desencolar**(), elimina el elemento del frente de la cola.
* **cabeza**(), retorna el elemento del frente de la cola.

Código en Matlab.

function cola = encolar(cola, valor)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para introducir un elemnto en la cola.

%Se verifica si la longitud de la cola es igual a 8.

if length(cola) == 8

disp('La cola está llena'); %Si es igual a 8 no se inserta nada en la cola.

else

cola = [cola valor]; %Si la cola aun no esta llena se inserta el valor indicado.

end

end

Código en Matlab.

function cola = desencolar(cola)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para eliminar el elemento del frente de la cola.

%Con un condicional se verifica si la cola está vacía.

if length(cola) == 0

disp('La cola está vacía'); %Si la cola esta vacía no se puede sacar ningún elemento.

else

cola(1) = []; %Si la cola no está vacía se elimina el elemento del frente de la cola.

end

Con un condicional se verifica si la cola está vacía, si la cola está vacía no se puede sacar ningún elemento, sino se elimina el elemento del frente de la cola.

Código en Matlab.

function valor = cabeza(cola)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Función para retornar el valor del frente de la cola.

valor = cola(1);

end

En el script **ejercicio4** se realizan las operaciones sobre una cola inicialmente vacía y se imprimen los valores almacenados en la cola al final de las operaciones realizadas y se imprimen también los valores almacenados en las variables x, y, z.

Código en Matlab.

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Operaciones realizadas en una cola implementada como un vector.

%encolar(), introduce un elemento al final de la cola.

%desencolar(), elimina el elemento del frente de la cola.

%cabeza(), retorna el elemento del frente de la cola.

q\_vector = [];

q\_vector = encolar(q\_vector, 10);

q\_vector = encolar(q\_vector, 20);

q\_vector = encolar(q\_vector, 30);

x = cabeza(q\_vector);

q\_vector = desencolar(q\_vector);

y = cabeza(q\_vector);

q\_vector = desencolar(q\_vector);

q\_vector = encolar(q\_vector, 40);

z = cabeza(q\_vector);

q\_vector = desencolar(q\_vector);

q\_vector = encolar(q\_vector, 50);

q\_vector = encolar(q\_vector, 60);

x = cabeza(q\_vector);

q\_vector = desencolar(q\_vector);

q\_vector = encolar(q\_vector, x);

q\_vector = encolar(q\_vector, y);

%Impresión de los valores almacenados en la cola al final de las

%operaciones realizadas.

fprintf('Los valores dentro de la cola son: %d %d %d %d\r\n', q\_vector);

%Impresión de los valores almacenados en las variables x, y, z.

fprintf('Los valores finales de x, y, z son: x = %d, y = %d, z = %d\r\n', x, y, z);

Resultados de las operaciones sobre la cola.

encolar(q,10); desencolar(q);

encolar(q,20); encolar(q,50);

encolar(q,30); encolar(q,60);

x=cabeza(q); x=cabeza(q);

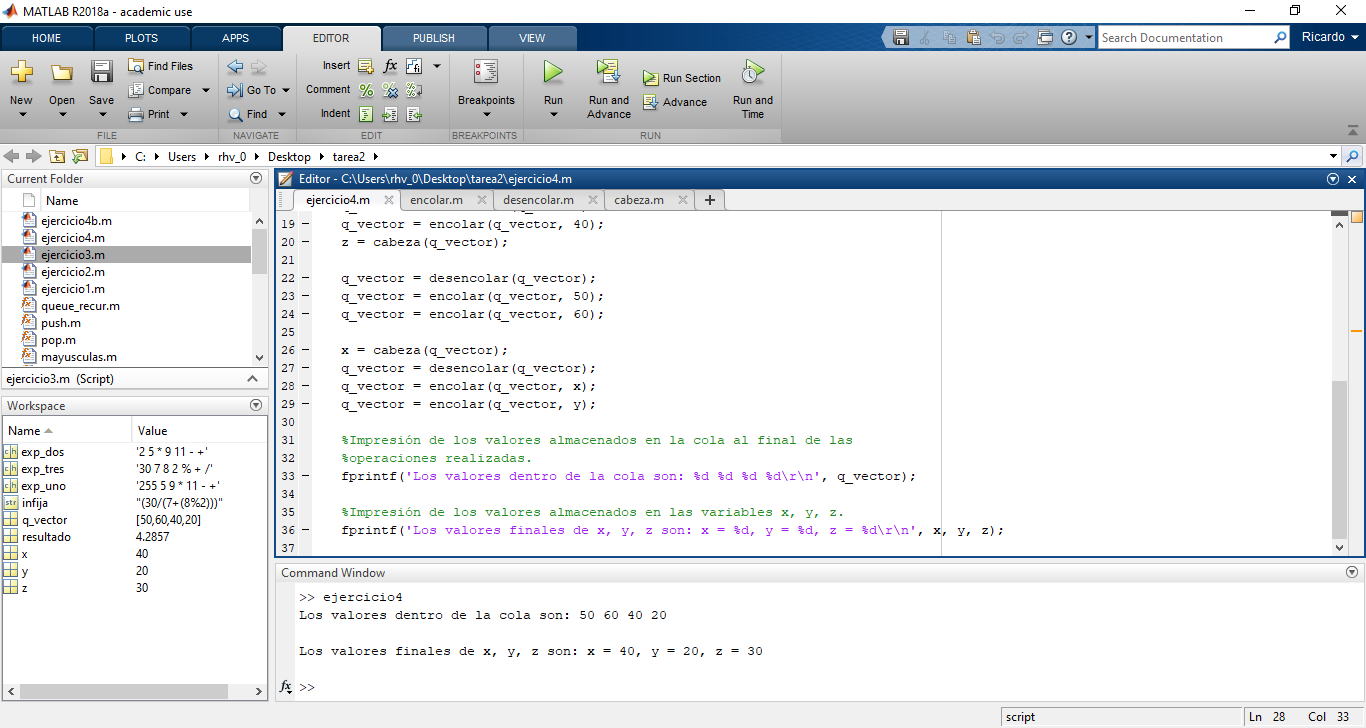
desencolar(q); desencolar(q);

y=cabeza(q); encolar(q,x);

desencolar(q); encolar(q,y);

encolar(q,40);

z=cabeza(q);



**Implementación con nodos enlazados.**

Para implementar la solución de este problema utilizando nodos enlazadosse utilizan dos funciones y un objeto:

* **queue**(), crea un objeto tipo cola, es necesario indicar el tamaño máximo.
* **enqueue**(), introduce un elemento al final de la cola.
* **dequeue**(), elimina el elemento del frente de la cola.
* **queue\_recur()**, busca el elemento del frente de la cola.

classdef queue

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Definición de un objeto tipo cola.

%Las propiedades de este objeto son el frente, tamaño actual y tamaño máximo

%de la cola.

properties

front

size

max\_size

end

methods

function obj = queue(max\_tam\_)

obj.max\_size = max\_tam\_;

obj.size = 0;

end

end

end

Las propiedades del objeto **queue** son el frente, tamaño actual y tamaño máximo de la cola, para acceder al elemento del frente de la cola se utiliza la propiedad front del objeto **queue** y la propiedad id de este.

Código en Matlab.

function queue = enqueue(queue,value\_)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

if isFull(queue)

disp('Enqueue: Full queue') %Si la cola está llena no se puede insertar elemento alguno.

else

if isEmpty(queue) %Si la cola no está llena se crea un nuevo nodo y se inserta en la cola.

newnode = node(value\_);

queue.front = newnode;

queue.size = queue.size+1;

else

queue.front = queue\_recur(queue.front,value\_); %Con la función queue\_recur() se realiza una busqueda por el primer elemento de la cola.

queue.size = queue.size+1;

end

end

end

Código en Matlab.

function [id, queue] = dequeue(queue)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

if isEmpty(queue)

disp('Dequeue: Empty stack'); %Si la cola está vacía no se puede extraer elemento alguno.

id = [] ;

else

id = queue.front.id ; %Si la cola no está vacía se extrae el elemento al frente de la cola.

queue.front = queue.front.nextnode; %Se asigna la cabeza al elemento siguiente del que se extrajo.

queue.size = queue.size-1;

end

end

Código en Matlab.

function cnode = queue\_recur(cnode, value)

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Con la función queue\_recur() se realiza una busqueda por el primer elemento de la cola.

if ~isempty(cnode.nextnode)

cnode.nextnode = queue\_recur(cnode.nextnode,value);

else

newnode = node(value);

cnode.nextnode = newnode;

end

end

En el script **ejercicio4b** se crea un objeto tipo **queue** y se realizan las operaciones sobre la cola inicialmente vacía y se imprimen los valores almacenados en la cola al final de las operaciones realizadas y se imprimen también los valores almacenados en las variables x, y, z.

Código en Matlab.

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Ricardo Hernández Varela.

%Ciencias Y Tecnologías Biomédicas.

%Operaciones realizadas en una cola implementada como nodos enlazados.

%queue(), crea un objeto tipo cola, es necesario indicar el tamaño máximo.

%enqueue(), introduce un elemento al final de la cola.

%dequeue(), elimina el elemento del frente de la cola.

%para accesar al elemento del frente de la cola se utiliza la propiedad front del objeto queue y la propiedad id de este.

cola = queue(20);

cola = enqueue(cola,10);

cola = enqueue(cola,20);

cola = enqueue(cola,30);

x = cola.front.id;

[id, cola] = dequeue(cola);

y = cola.front.id;

[id, cola] = dequeue(cola);

cola = enqueue(cola,40);

z = cola.front.id;

[id, cola] = dequeue(cola);

cola = enqueue(cola,50);

cola = enqueue(cola,60);

x = cola.front.id;

[id, cola] = dequeue(cola);

cola = enqueue(cola,x);

cola = enqueue(cola,y);

%Con un ciclo se recorre la cola para extrar sus elementos y almacenarlos

%en un vector.

num = cola.front;

valores = [];

for i=1:cola.size

elemento = num.id;

valores = [valores elemento];

num = num.nextnode;

end

%Impresión de los valores almacenados en la cola al final de las

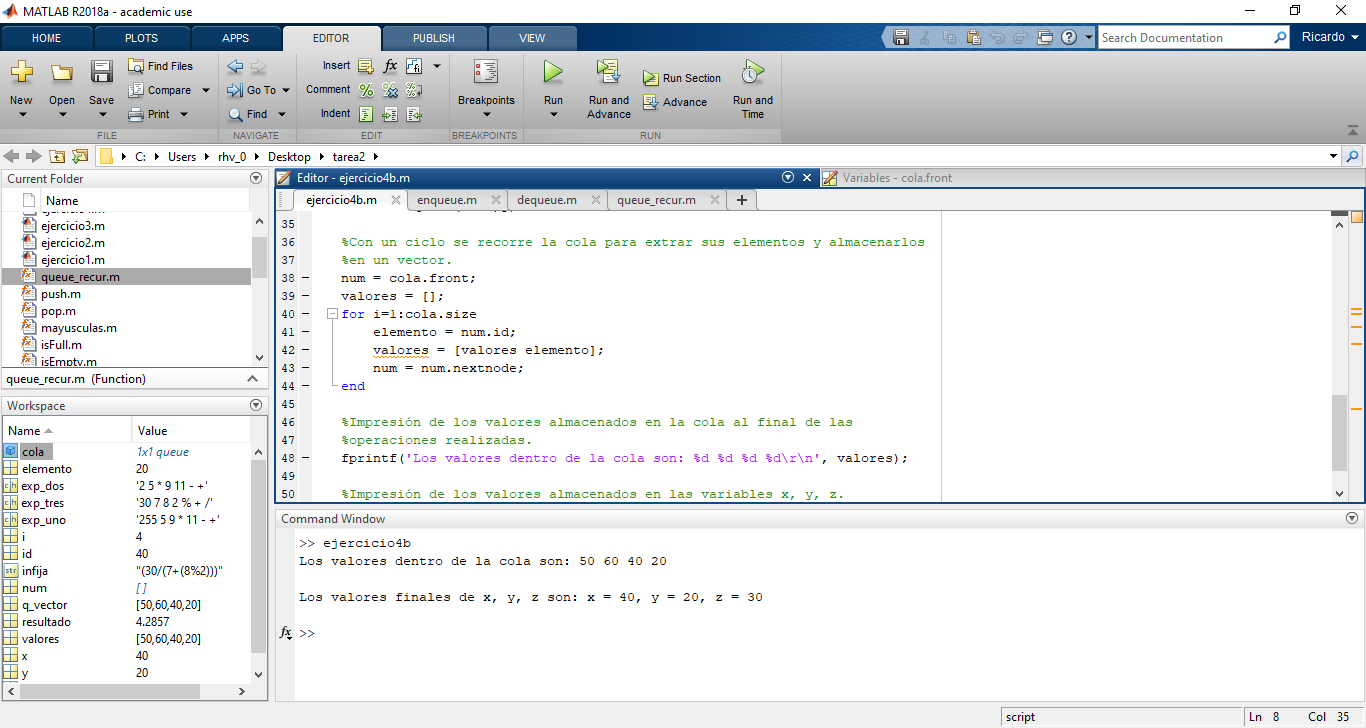
%operaciones realizadas.

fprintf('Los valores dentro de la cola son: %d %d %d %d\r\n', valores);

%Impresión de los valores almacenados en las variables x, y, z.

fprintf('Los valores finales de x, y, z son: x = %d, y = %d, z = %d\r\n', x, y, z);

Resultados de las operaciones sobre la cola.



III. COMENTARIOS

El desarrollo de las funciones para implementar los periodogramas requirió implementar diferentes pasos para los cuales se desarrolló una función particular para poder dividir el proceso en tareas que resultaran más fáciles de verificar ya que el periodograma de Bartlett y el de Welch involucran diferentes cálculos y procesos relacionados con ajustar los datos de entrada para los cálculos.

Las funciones principales que se llaman para generar los diferentes periodogramas incluyen llamadas a otras funciones con sus respectivas subtareas, estas funciones principales incluyen el código para generar las gráficas de cada señal con los parámetros de entrada para realizar cada una de las pruebas, este código se implementó para que pudiera ajustar los ejes y el título de la gráfica de forma adecuada.

1. [↑](#footnote-ref-1)