Practica Inicial

Ejercicio 1:

```
%%Bacteria
disp("Bacteria:")
%Con imread le podemos pasar la direccion a una imagen para que la cargue
imgEnt = imread('../Imagenes/bacteria.tif');
%whos nos muestra informacion sobre una variable o el entorno
whos("imgEnt")
%imshow nos muestra la imagen en una ventana nueva
imshow(imgEnt)
%Con min y max podemos conseguir los valores maximos y minimos de una lista
%o matriz
disp("Valor minimo: "+min(imgEnt(:)))
disp("Valor maximo: "+max(imgEnt(:)))

%%Flowers
disp("Flowers:")
imgEnt = imread('../Imagenes/flowers.tif');
whos("imgEnt")
%Las imagenes a color reparten la intensidad de sus 3 colores(Este caso
%rgb) en 3 matrices, de esta manera podemos mostrar cada una por separado
imshow([imgEnt(:,:,1),imgEnt(:,:,2),imgEnt(:,:,3)]);
disp("Valor minimo: "+min(imgEnt(:)))
disp("Valor maximo: "+max(imgEnt(:)))
%Con imtool podemos inspeccionar la imagen
imtool('../Imagenes/flowers.tif');
```

La ejecución de este código nos devuelve:
>> p0_1
Bacteria:

Name	Size	Bytes	Class	Attributes
imgEnt	178x178	31684	uint8	

Valor mínimo: 0
Valor máximo: 239
Flowers:

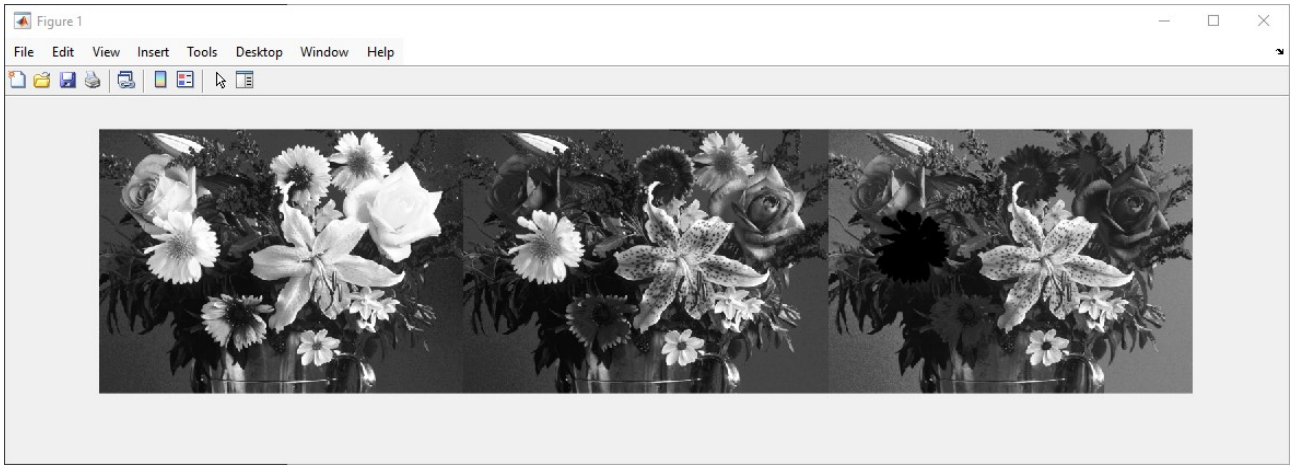
Name	Size	Bytes	Class	Attributes
imgEnt	362x500x3	543000	uint8	

Valor mínimo: 0
Valor máximo: 255

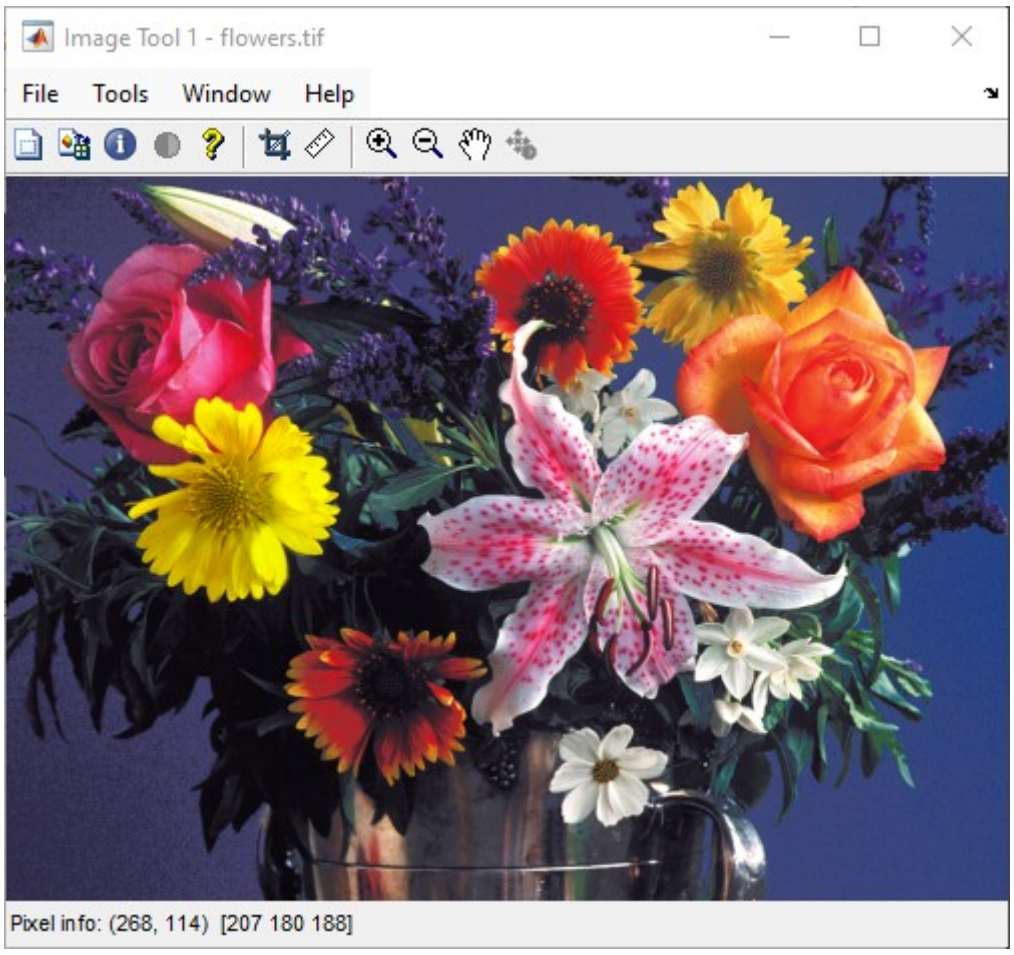
Donde podemos ver que la imagen bacteria tiene un tamaño de 178x178, un rango de grises de 256 valores indicado por la clase de valor uint8. Ademas de ver que el valor mínimo es 0 y el máximo 239.

Para la imagen de las flores en color podemos ver que el tamaño es 362x500 y el x3 nos indica que los colores están repartidos en 3 matrices, la intensidad de los colores tiene el mismo rango que la imagen anterior. Por ultimo, los valores mínimos y máximos son 255.

Ademas, nos muestra las imágenes de esta manera:



En este caso, es una captura de la imagen flores con cada uno de sus canales rgb por separado.
Si usamos el comando “imtool” para mostrar una imagen vemos lo siguiente:

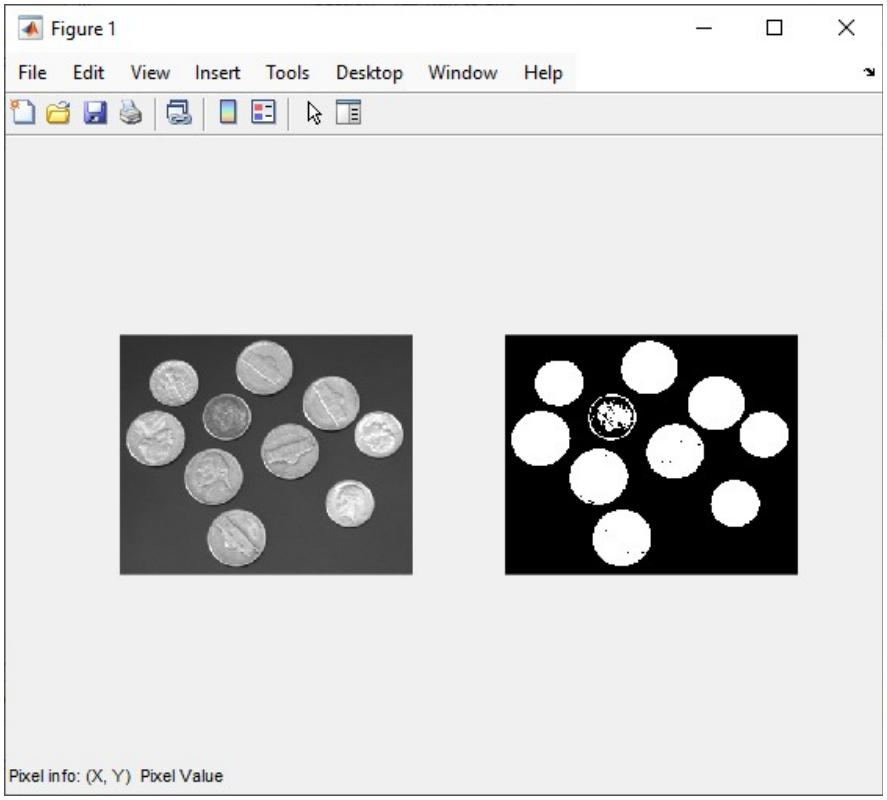


Donde podemos ver en la parte de abajo de la imagen la información del pixel al que estemos apuntando, en este caso viendo la posición y la intensidad de los valores rgb.

Ejercicio 2:

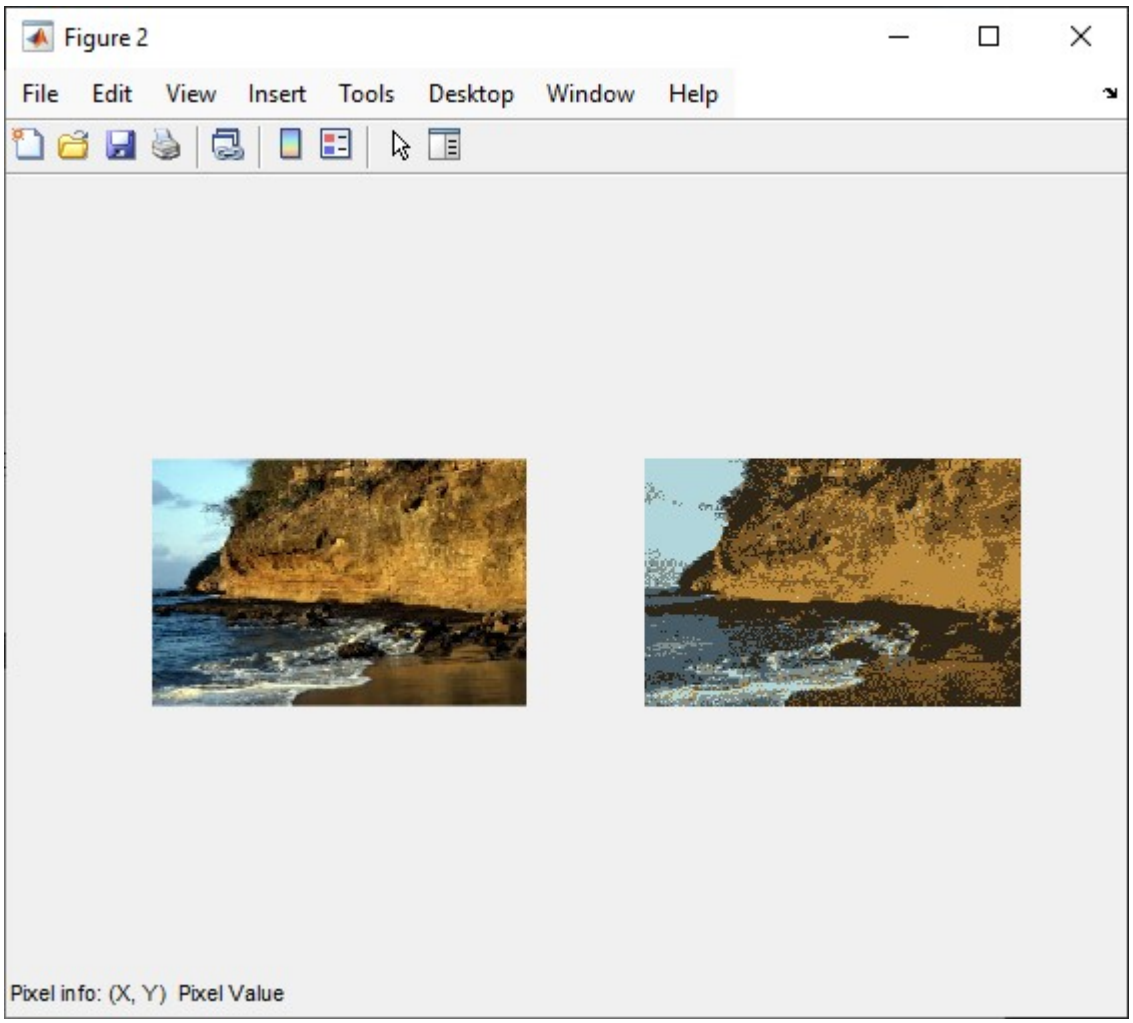
```
imgEntGris = imread(fullfile("../Imagenes/coins.png"));
figure(1);
subplot(1,2,1);
imshow(imgEntGris);
%Muestra la informacion del pixel sobre el que tengamos el raton
impixelinfo
%Convertimos la imagen a blanco y negro dependiendo de un umbra opcional.
%Los pixeles con mayor intensidad al umbral se convertiran en blanco
%Los que tengan un valor menor o igual a negro
subplot(1,2,2);
imgBW = im2bw(imgEntGris);
imshow(imgBW);
impixelinfo

figure(2);
I = imread("../Imagenes/acantilado.png");
subplot(1,2,1);
imshow(I);
numColors = 6; %Número de colores
%El comando rgb2ind nos permite convertir una imagen a una imagen indexada
%con una paleta de colores.
%Nos devuelve una matriz que sera del tamaño de la imagen pero en vez de
%guardar los colores, guardara un indice para cada pixel indicando una
%posicion en cmap
%cmap es una tabla o matriz que guarda los colores en formato rgb
[indexedImage, cmap] = rgb2ind(I, numColors);
subplot(1,2,2);
imshow(indexedImage, cmap);
impixelinfo
```

Para la primera parte, la ejecución del código nos muestra lo siguiente:



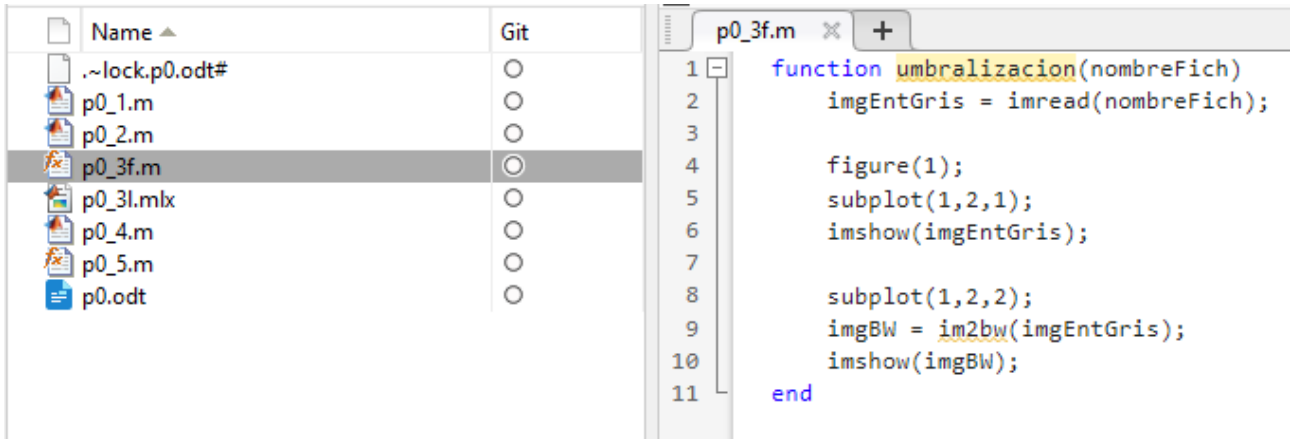
Que son las imágenes antes y después de aplicarle “im2b”

Para la segunda parte vemos:



Donde tenemos la imagen normal y en la segunda parte la imagen indexada.

Ejercicio 3:



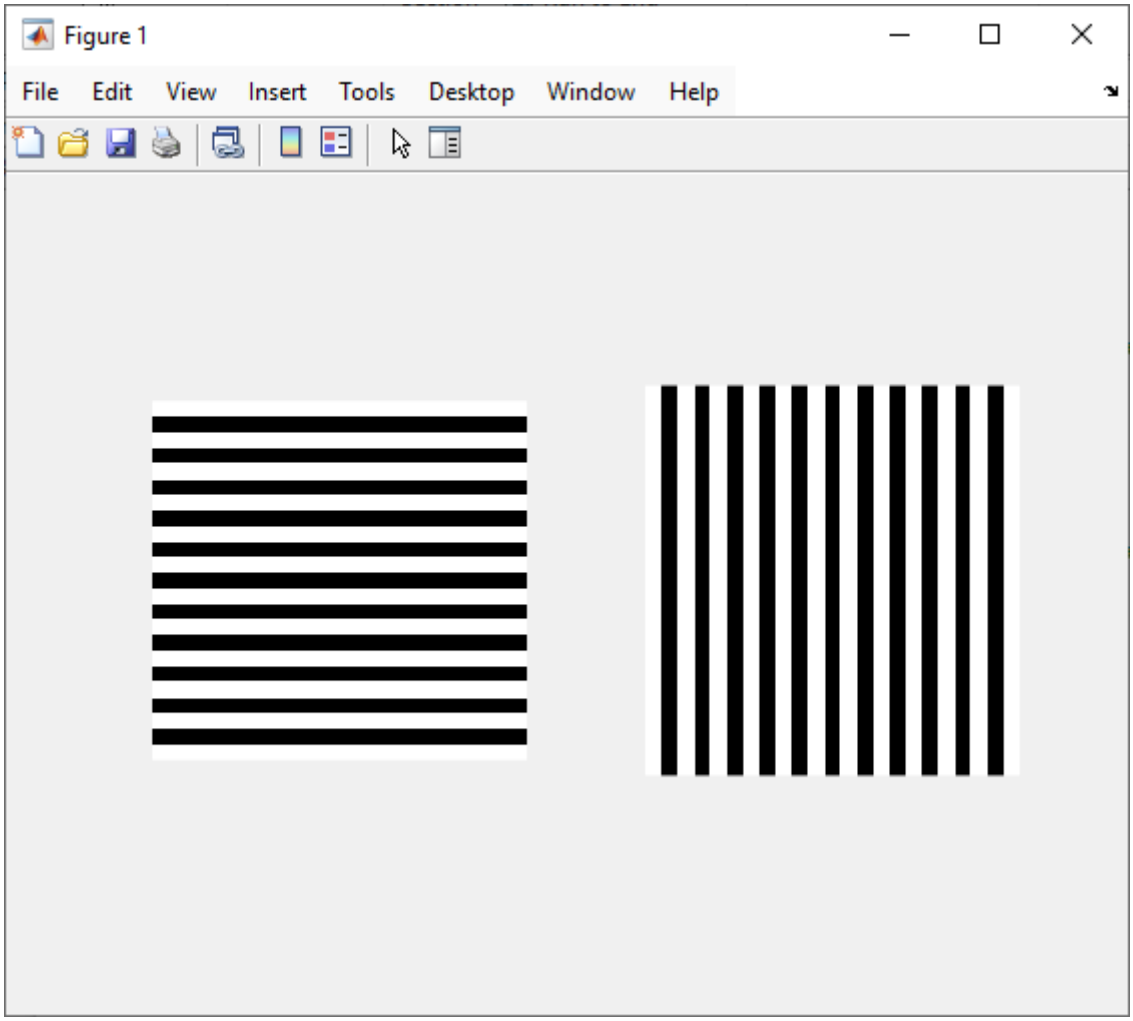
Creo un archivo .m y creo la función que usara comandos del ejercicio anterior. De esta manera, se podrá llamar a esa función desde otros archivos para reutilizar código.

Lo siguiente que hay que hacer es crear un archivo .mlx. Este tipo de archivos son muy útiles al funcionar parecidos a cuadernos de jupyter de manera que podemos escribir código, texto y tener el output en el mismo archivo. En este caso, tengo cuadros con el código y debajo de cada “imshow” se puede ver el output de ese comando.

Ejercicio 4:

```
%%Asi creamos una matriz de ceros del tamaño que queramos
imgBW = false([400,600])
%Tenemos un bucle for que va desde 1 hasta 600 de 50 en 50
for i=1:50:600
    %%Con esto ponenmos las filas de i hasta i+25 y todas las
columnas a 1
    imgBW (i:i+25,:)=true;
end
figure(1);
subplot(1,2,1);
imshow(imgBW);
%%Una manera de invertirlo es usar el operando " ' " que nos devuelve
la
%%traspuesta de una matriz
subplot(1,2,2);
imshow(imgBW')
```

Este código nos da como resultado:



Ejercicio 5:

```
function franjasBlancoNegro()
    %%Creamos una matriz de ceros y lo casteamos a uint8 para que
    %%cada pixel tenga un rango de 256 grises
    imgBW = uint8(zeros([256,256]));
    %%Tenemos un bucle que recorra toda la imagen
    for i=1:1:256
        %%Igualamos cada fila completa al valor de i-1 de manera
        %%que vaya desde 0 hasta 255
        imgBW (i,:)=i-1;
    end
    figure(1);
    subplot(1,2,1);
    imshow(imgBW);
    title('Franjas horizontales');
    %%Una manera de invertirlo es usar el operando " ' "
    %%que nos devuelve la traspuesta de una matriz
    subplot(1,2,2);
    imshow(imgBW');
    title('Franjas verticales');
end
```

Que nos da como resultado: