

```
% exercise 01 for the image processing 1 class
close all;
clear;
```

%% exercise 1a

```
close all;
clc
```

```
imgFilename = 'image1.png';
```

```
%loading the image
img = my_loadImage(imgFilename);
```

```
%showing the image
my_showImage(img);
```

```
%solution
exercise_1a(imgFilename);
```

```
function [ img ] = my_loadImage( filename )
%MY_LOADIMAGE Loads an image with the given file.
%The image data type needs to be converted double.
% the range needs to be converted from [0-255] to [0.0-1.0]
img = imread(filename);
img = double(img); %integer in double
img = img ./255; %skalieren auf 0.0 bis 1.0
end
```

```
function my_showImage( img )
%MY_SHOWIMAGE Presents the image to the user.
imagesc(img);
axis equal;
set(gca, 'Visible', 'off');
colormap gray;
end
```

%% exercise 1b

```
close all;  
clc
```

```
%splitting the RGB channels of the color image  
[imgR, imgG, imgB] = my_RGBSplit(img);
```

```
%plotting the image and each RGB channel  
my_plotRGBSplit(img, imgR, imgG, imgB);
```

```
%solution  
exercise_1b(img);
```

```
function [ imgR, imgG, imgB ] = my_RGBSplit( imgRGB )  
%my_RGBSplit Splits the RGB channels of the images in separate images.  
  
R = imgRGB(:,:,1); % Rot-Kanal --> 512x512  
G = imgRGB(:,:,2); % Grun-Kanal --> 512x512  
B = imgRGB(:,:,3); % Blau-Kanal --> 512x512  
h = zeros(size(imgRGB, 1), size(imgRGB, 2)); % 512x512 Matrix erstellen  
% zum cat-Befehl: C = cat(dim, A1, A2, A3, A4, ...)  
%concatenates all the input arrays (A1, A2, A3, A4, and so on)  
%along array dimension dim.  
imgR = cat(3, R, h, h); % der Blau-Kanal und der Grun-Kanal werden 0 gesetzt --  
-> nur Rot  
imgG = cat(3, h, G, h); % der Rot-Kanal und der Grun-Kanal werden 0 gesetzt --  
> nur Grun  
imgB = cat(3, h, h, B); % der Rot-Kanal und der Blau_Kanal werden 0 gesetzt --  
> nur Blau  
end  
  
function my_plotRGBSplit( imgRGB, imgR, imgG, imgB )  
%MY_PLOTRGBSPLIT Plots the original RGB image and  
%its splitted R,G,B channels in a single figure.  
subplot(1,4,1) % 1 Zeile, 4 Spalten, 1. Position  
my_showImage(imgRGB)  
subplot(1,4,2)  
my_showImage(imgR)  
subplot(1,4,3)  
my_showImage(imgG)  
subplot(1,4,4)  
my_showImage(imgB)  
  
end
```

%% exercise 1c

```
close all;
clc
```

```
%gamma correct the image
imgGC = my_gammaCorrection(img);
```

```
my_showImage(imgGC);
```

```
%solution
exercise_1c(img);
```

```
function [ imgGC ] = my_gammaCorrection( img )
% MY_GAMMACORRECTION calculate a gamma corrected version of the image
```

```
% Bei Gammawerten grosser als 1 steigt der Anstiegswinkel der Kurve
% uber 45 grad --> reproduzierte Bilder werden dadurch heller dargestellt
% als im Originalbild
```

```
imgGC = img.^(2.2); % angenommener Gamma-Wert aus Vorlesung 2
```

```
end
```

%% exercise 1d

```
close all;
clc
```

```
%convert the RGB image to grayscale image
```

```
imgGray = my_RGB2Gray(img);
my_showImage(imgGray);
```

```
%solution
exercise_1d(img);
```

```
function [ imgGray ] = my_RGB2Gray( img )
%MY_RGBTOGRAY converts the RGB image to a grayscale image.
% before the conversion a gamma correction is performed.
```

```
%Gamma-Korrektur
```

```
imgGC= my_gammaCorrection(img);
```

```
%Gewichtung der RGB-Kanäle, damit optisch ein gutes Grauwertbild
%herauskommt
```

```
imgGray = (0.299*imgGC(:,:,1) + 0.5870*imgGC(:,:,2)+...
0.1140*imgGC(:,:,3)); % Gewichtungen aus "Digitale Bildverarbeitung
% Eine algorithmische Einführung mit Java". Autor: Wilhelm Burger. 3.
% Auflage.
end
```