# Laboratório 1

```
begin
    using Images ✓
    using PlutoUI ✓
    using FileIO ✓
    import PlutoUI: combine
end
```

## Parte I – Leitura e Gravação de Arquivos de Imagens

Para essa primeira parte simplesmente vamos fazer uma cópia de uma imagem.

```
begin
    src = load("./img/Gramado_22k.jpg")
    dst = copy(src)
    #save("./img/copia_de_Gramado_22k.jpg", dst)
end
```

Ao fazermos um *load* da imagem em `Julia`, salvamos na memória uma `matriz de RGB`. Por isso, quando salvamos em um arquivo o *encoder* não sabe qual era o tamanho da imagem original, muito menos que ela nem foi alterada. Já que `JPEG` é um método de compressão *lossy*, algumas informções serão realmente perdidas.

Por tudo isso, o arquivo original era de 21,7 kB foi para 32,1 kB.

# Parte II – Leitura, Exibição e Operações sobre Imagens

## a) Espelhamento horizontal e vertical da imagem original

```
h_mirror (generic function with 1 method)
```

```
function h_mirror(img)
    result = similar(img)                  # cria uma nova imagem de
dimensões
                                           # iguais ao input
    height=size(img, 1)                    # altura da imagem
    width=size(img, 2)                     # largura da imagem

    for y in 1:height                      # loop sobre a imagem original
        for x in 1:width
            result[y, x] = img[y, width-x+1] # andamos sobre o y, porém
pegamos o
        end                                # x do outro lado
    end
    return result
end
```

```
v_mirror (generic function with 1 method)
```

```
function v_mirror(img)
    result = similar(img)                    # cria uma nova imagem de
dimensões
                                             # iguais ao input
    height=size(img, 1)                      # altura da imagem
    width=size(img, 2)                       # largura da imagem

    for y in 1:height                        # loop sobre a imagem original
        for x in 1:width
            result[y, x] = img[height-y+1, x] # andamos sobre o x, porém
pegamos o
        end                                  # y do outro lado
    end
    return result
end
```

# b) Converta uma imagem colorida para tons de cinza (luminância).

Para essa função resolvi atuar sobre uma cor ao invés de fazer para imagem inteira, assim ela fica mais genérica

```
luminance (generic function with 1 method)
```

```
function luminance(color)
    l=0.299*(color.r)+0.587*(color.g)+0.114*(color.b)
    return RGB(l, l, l)
end
```

```
luminance_image (generic function with 1 method)
```

```
function luminance_image(img)
    return luminance.(img)
end
```

# c) Implemente um processo de quantização (de tons) sobre as imagens em tons de cinza.

```
quantize (generic function with 1 method)
```

```
function quantize(img, n)
    q=n                                     # passo necessário uma vez que
julia
    n=n/256                                 # trabalha com imagens codificadas
em float

    t1=img[1,1].r                           # menor tom
    t2=img[1,1].r                           # maior tom

    for y in 1:size(img, 1)                 # loop sobre a imagem original
        for x in 1:size(img, 2)             # para achar o maior e o menor tom
            if t2<img[y,x].r
                t2=img[y,x].r
            end
            if t1>img[y,x].r
                t1=img[y,x].r
            end
        end
    end

    tam_int = t2-t1                          # calculamos o tamanho do
intervalo e
    if tam_int <= n                          # se o n informado for maior que
o tam_int
        return img                           # retornamos a imagem original
    end

    tb = tam_int/q

    result = similar(img)                    # cria uma nova imagem de
dimensões
                                             # iguais ao input

    for y in 1:size(img, 1)                  # loop sobre a imagem original
(de novo)
        for x in 1:size(img, 2)
            pixel = img[y,x].r               # pegamos um pixel (r, já que
r=g=b)
            b=floor((pixel-t1)/(tam_int)*n*255) # deslocamos esse pixel t1,
                                             # multiplamos por 1/tam_int
para
                                             # deixar entre 0 e tam_int,
                                             # multiplicamos por n e
depois por 255
            result[y, x] = (t1+b*tb+tb/2)    # o resultado será multiplicado
por tb
```

```
quantize_color (generic function with 1 method)
```

# PhotoChop.jl

Para começar, selecione um arquivo do seu computador

Escolher arquivo Nenhum arquivo selecionado

Agora, selecione o filtro que você quer usar

Nenhuma

Escolha uma imagem

Escolha uma imagem

Escolha o formato do arquivo para salvar

- (•) PNG
- ( ) JPEG

Escolha uma imagem