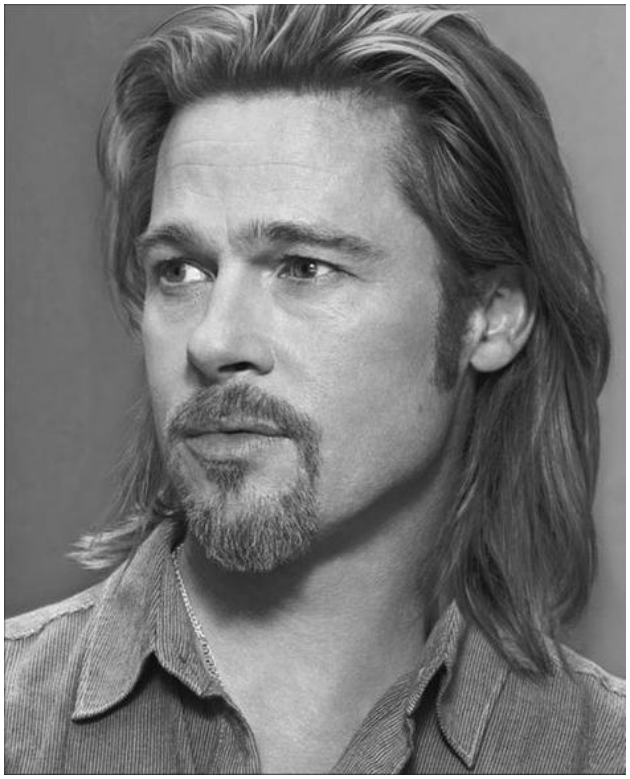
INF100 – Introdução à Programação I
Roteiro Prática 27 a 29 de junho de 2017

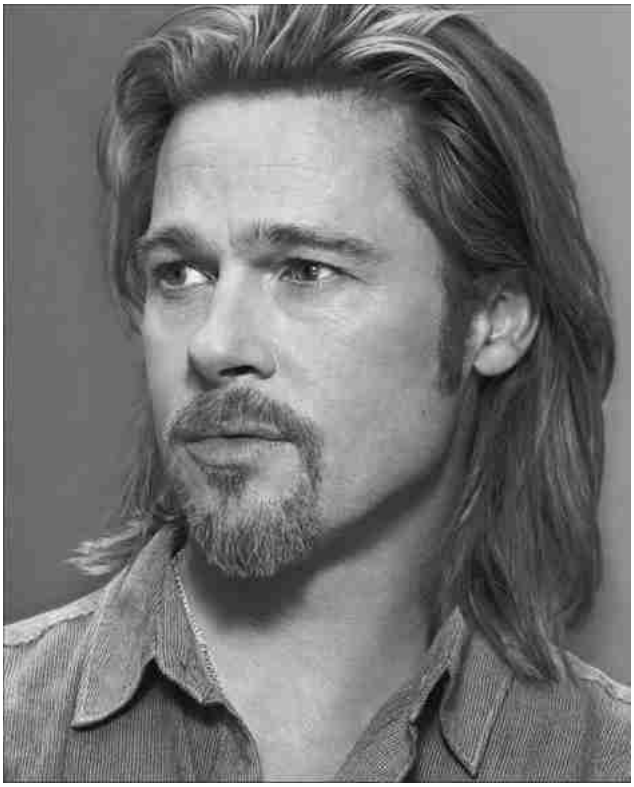
Faça o download do arquivo **imagens.pyc** e das imagens (***.jpg**) para dentro da "Pasta pessoal".

Introdução

Atualmente, o formato mais popular para armazenar imagens fotográficas em arquivos digitais é o JPEG. Os arquivos nesse formato são facilmente identificados pela extensão .jpg ou .jpeg. Essa sigla vem do "*Joint Photographic Experts Group*", grupo fundado em 1986 e que criou o padrão em 1992. Apesar de existirem outros formatos mais novos e mais eficientes, o JPEG possivelmente continuará sendo usado por muitos anos.

No fomato JPEG, a imagem perde parte de sua informação, permitindo que o seu tamanho em bytes possa ser reduzido, mas não a ponto de prejudicar a sua visualização. A fotografia abaixo ilustra isso. À esquerda vemos a foto original, sem compressão. Na foto da direita a imagem foi comprimida a uma taxa de 10:1. Ou seja, o arquivo resultante ocupa apenas cerca de 10% do tamanho do arquivo original.





Ao visualizar os dois arquivos lado a lado, é possível ver as diferenças, especialmente se dermos um *zoom* nas imagens. Outra forma de identificarmos as diferenças entre as imagens é através da diferença entre as matrizes que compõem os *pixels* de ambas.

Nesta prática, você fará um programa para produzir uma imagem que é a diferença entre duas imagens dadas. Mas, em vez de lidar com imagens coloridas, faremos isso com uma imagem em tons de cinza. Isso significa que cada *pixel* é formado por um único valor entre 0 (preto) e 255 (branco), em vez dos três componentes R, G, B que usamos na Prática 14. Com isso, podemos usar comandos mais simples para "ler" e "escrever" um *pixel* em uma posição (*i*,*j*) qualquer:

```
p = im[i][j]
...
im[i][j] = p
```

Roteiro de Prática

Uma visão geral do programa (algoritmo de alto nível) segue abaixo:

1. Leitura e exibição da imagem im1
2. Leitura e exibição da imagem im2

3. Cálculo da diferença entre as imagens im1 e im2, obtendo a imagem im3
4. Exibição da imagem im3
5. Cálculo do valor máximo de pixel presente na imagem im3
6. Alargamento de contraste da imagem im3
7. Exibição da imagem im3 com novo contraste

O algoritmo acima já está implementado dentro do esqueleto do programa. Para que ele funcione direito, você deverá preencher a implementação das seguintes funções:

`difImagens( im1, im2, im3 )`

Essa função deve implementar a operação matricial **im3** = | **im1** – **im2** |. Considerando que todas as imagens possuem o mesmo tamanho, isso deve ser feito assim:

```
para i = 0 até im1.altura-1:
    para j = 0 até im1.largura-1:
        im3[i][j] = abs( im1[i][j] - im2[i][j] )
```

A função deve considerar que a imagem **im3** já foi criada previamente, ou seja, ela só precisa ser preenchida como mostrado acima.

A imagem **im3** resultante desse cálculo será aparentemente uma tela toda preta. Os *pixels* pretos representam os pontos que são idênticos entre as imagens **im1** e **im2**. No entanto, olhando com mais cuidado, você verá alguns pontinhos claros, parecendo uma "sujeira" na imagem. Esses pontos serão mais visíveis na parte inferior. Para visualizar melhor esses pontos teremos que fazer um "alargamento de contraste" nessa imagem...

`maxPixel( img )`

Essa função deve percorrer a imagem **img** e determinar o maior valor de *pixel* encontrado nela, retornando esse valor.

`alargaContraste( img )`

Essa função deve fazer o seguinte:

- Obtém mx = maxPixel(img);

- Para cada pixel pertencente à imagem **img**, alterar seu valor usando a seguinte expressão:

```
img[i][j] = round( img[i][j] * 255 / mx )
```

A função `round()` serve para arredondar o valor para o número inteiro mais próximo.

O propósito dessa operação é aumentar o brilho dos pixels que representam diferenças entre a imagem original e a comprimida, permitindo uma melhor visualização dessas diferenças.

`contaZeros( img )`

Essa função deve percorrer a imagem **img** e determinar quantos *pixels* iguais a zero aparecem nela, retornando essa contagem. Esse número de zeros irá representar a quantidade de pixels que são idênticos nas duas imagens.

Segue abaixo a saída esperada pelo programa:

```
Calculando a diferença entre as duas imagens...
Calculando o "alargamento de contraste" da diferença...
Calculando a semelhança entre as imagens...
As imagens im1 e im2 são idênticas em 8.76% dos pixels.
```

☞ Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p17.py**) através do sistema do LBI.