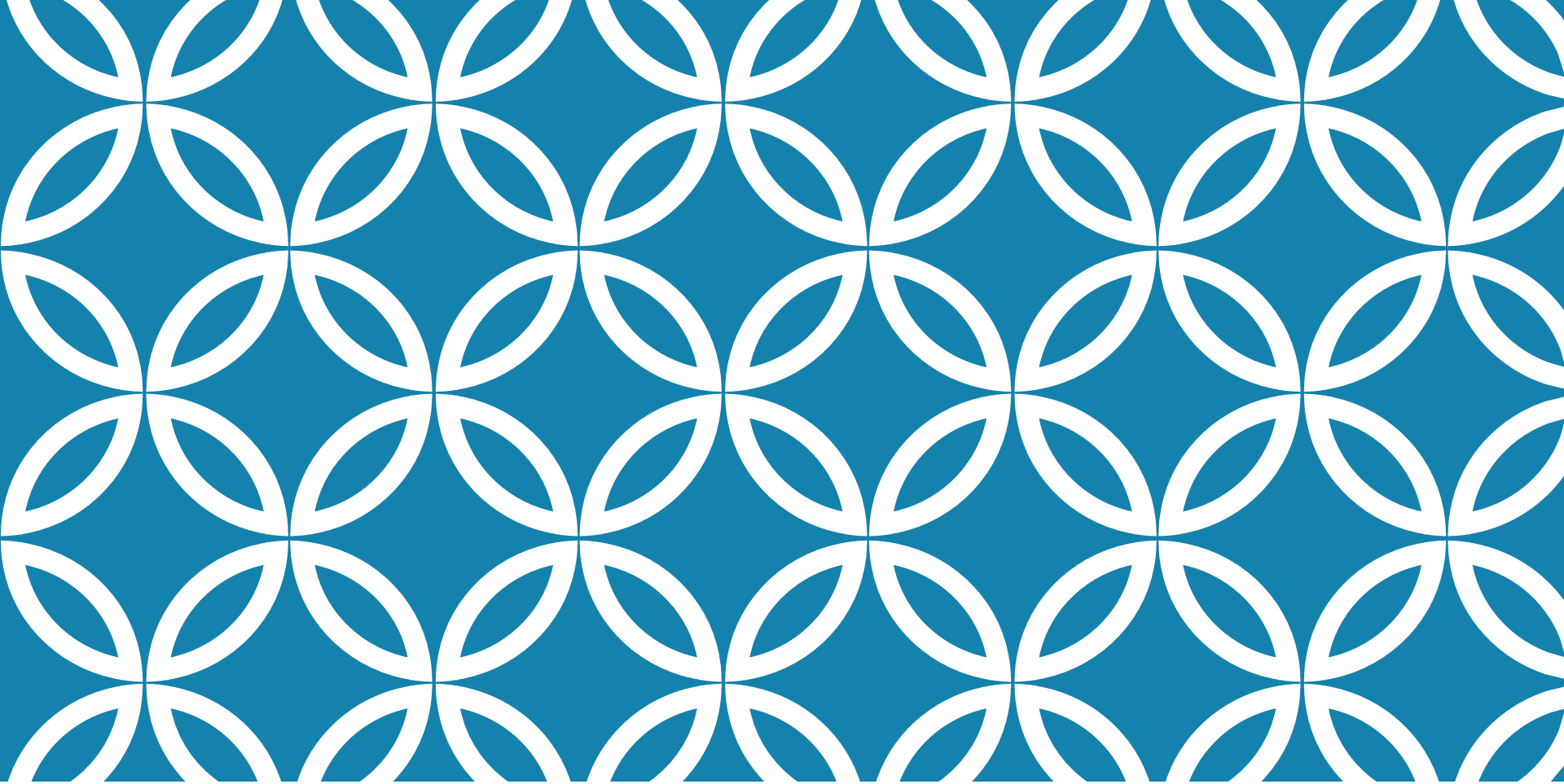


IMAGENS

Prof. André Vignatti

INSPIRAÇÃO PARA A DISCIPLINA

CI055: iremos programar somente com **números**

PORQUÊ?
PORQUÊ?
PORQUÊ?
PORQUÊ?
PORQUÊ?

IMAGENS

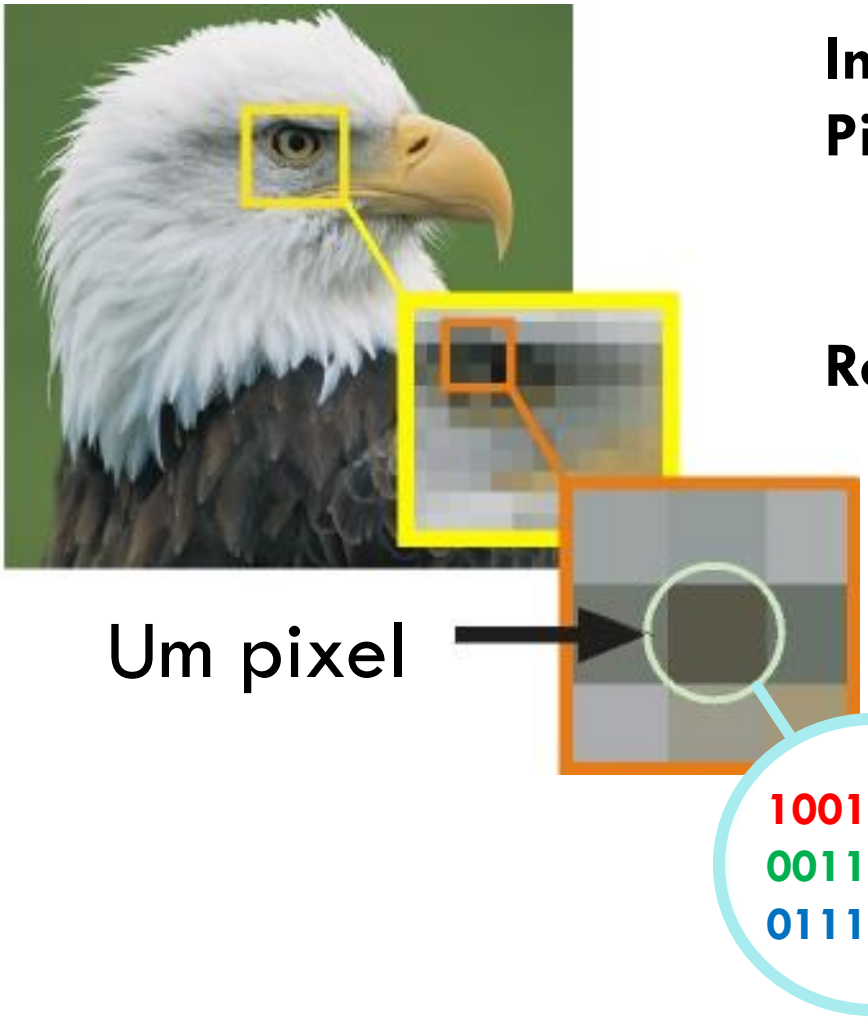


Imagem: matriz de pixels

Pixel:

- "átomo" da imagem
- 24 bits

Resolução: dimensões da matriz

8 bits para VERMELHO
8 bits para VERDE
8 bits para AZUL

FORMATO PGM

formato PGM: imagem com codificação *ASCII*, sem compactação, fácil de manipular:

- a 1ª linha contém um identificador "**P2**"
- a 2ª linha contém o número de **colunas** e de **linhas** (nesta ordem)
- a 3ª linha contém o maior valor da matriz
- o restante do arquivo contém um sequência de inteiros (bytes), cada inteiro representa um pixel da imagem em tons de cinza.

EXEMPLO DE ARQUIVO PGM

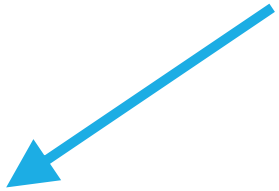
```
P2
11 10
40
40 5  5  5  5  5  5  5  5  40 0
5  20 20 5  5  5  5  5  5  5  5
5  5  20 5  5  5  0  0  0  0  0
5  5  20 20 5  5  20 20 0  0  5
5  5  5  5  5  5  0  20 0  0  0
5  5  5  5  5  5  0  20 20 0  5
5  5  5  5  11 11 11 0  0  0  0
5  5  5  5  20 20 11 5  5  5  5
5  5  5  5  11 20 11 5  5  5  0
40 5  5  5  11 20 20 5  5  40 5
```

CONVERTENDO OUTRAS IMAGENS EM PGM

Queremos “brincar” com a imagem de nosso gosto!

- na linha de comando do Linux:

```
convert original.jpg -compress none saida.pgm
```



aceita outros formatos também:
png, gif, tiff, bmp, ...

às vezes, gera linhas “comentadas”: **remover manualmente** ou **ignorar no nosso programa**

VARIÁVEIS GLOBAIS, TIPOS E CONSTANTES

Quanto de memória a matriz ocupa?

```
program pgm;

const
    X = 50;
    limiar = 50;
type
    imagem = array [1..2000 ,1..2000] of integer;
var
    O, D: imagem;
    linO, colO, maxO,
    linD, colD, maxD: integer;
```

matriz de origem (entrada)

matriz de destino (saída)

CARREGAR IMAGEM

```
procedure ler_pgm (var O : imagem; var l, c, max : integer);
var
    i , j : integer;
    s : string [2];
begin
    readln (s) ;
    if s = 'P2' then begin
        read (c, l);
        read (max) ;
        for i:= 1 to l do
            for j:= 1 to c do
                read (O[i ,j]) ;
    end
    else writeln ('Formato invalido') ;
end;
```

CARREGAR IMAGEM

P: vou ter que digitar “na mão” os valores?

R: NÃO! Utilize **redirecionamento para a entrada padrão** no Linux!

```
./nome_do_programa < arquivo.pgm
```

IMPRIMIR IMAGEM

```
procedure imprimir_pgm (var O : imagem; l, c, max : integer);
var
    i, j : integer;
begin
    writeln ('P2') ;
    writeln (c, ' ', l);
    writeln (max);
    for i:= 1 to l do begin
        for j:= 1 to c-1 do
            write (O[i,j] , ' ') ;
        writeln (O[i,c]) ;
    end;
end;
```

IMPRIMIR IMAGEM

P: vai imprimir os caracteres na tela??!!

R: SIM, mas você pode: *(1)* redirecionar para um arquivo ou *(2)* usar pipe e o comando *display*

```
./nome_do_programa < arquivo.pgm > saida.pgm
```

```
./nome_do_programa < arquivo.pgm | display
```

CLAREANDO IMAGENS

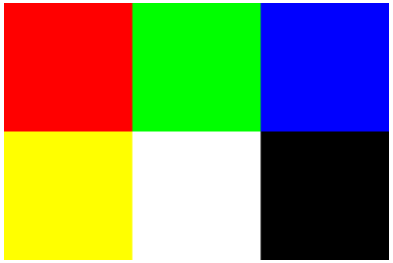
```
procedure clarear_pgm (var O : imagem; l, c, max, cte : integer);
var
    i, j : integer;
begin
    for i:= 1 to l do
        for j:= 1 to c do begin
            O[i,j] := O[i,j] + cte;
            if O[i,j] > max then
                O[i,j] := max;
        end;
end;
```

IMAGENS COLORIDAS – FORMATO PPM

Imagens **ppm** são parecidas com **pgm**, mas para imagens coloridas:

```
P3
3 2
255
# A parte acima eh o header
# "P3" significa que eh uma imagem colorida RGB em ASCII
# "3 2" eh a largura e altura da imagem em pixels
# "255" eh o valor maximo para cada cor
# A parte abaixo contem os dados da imagem: triplas RGB
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

Neste caso, seria essa imagem (obviamente, aumentada):



IMAGENS COLORIDAS – PERGUNTAS PARA PENSAR

1. Como alterar o programa Pascal para lidar com imagens coloridas ppm?

2. Como transformar uma imagem colorida em tons de cinza?

3. Como inverter as cores de uma imagem colorida?