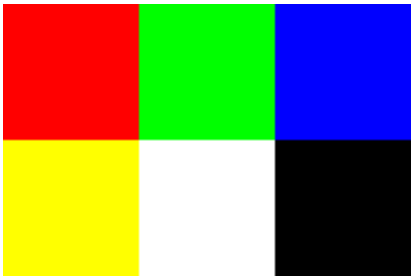
Laboratório de Computadores - Projecto Processamento de Imagem

Preâmbulo

Pretende-se que implemente um conjunto de operações de processamento de imagem em formato de armazenamento *ASCII/Plain Portable Pixel Map* (PPM).
A descrição do formato ASCII/Plain PPM está disponível em http://netpbm.sourceforge.net/doc/ppm.html, e em https://en.wikipedia.org/wiki/Netpbm.

Por exemplo, o conteúdo do ficheiro *img.ppm*, correspondente à seguinte imagem:



é:

```
P3
3 2
255
# The part above is the header
# "P3" means this is a RGB color image in ASCII
# "3 2" is the width and height of the image in pixels
# "255" is the maximum value for each color
# The part below is image data: RGB triplets
255   0   0  # red
  0 255   0  # green
  0   0 255  # blue
255 255   0  # yellow
255 255 255  # white
  0   0   0  # black
```

Operação 1 - Rotação horizontal

Pretende-se que implemente um programa em C que faça uma rotação horizontal (rotação de 180 graus em torno de um eixo vertical) de uma imagem em formato PPM. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).
A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_h_flip`):

```
$ ./ppm_h_flip [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_h_flip < input.ppm > output.ppm
$ ./ppm_h_flip input.ppm > output.ppm
$ ./ppm_h_flip input.ppm output.ppm
```

Sugestões:

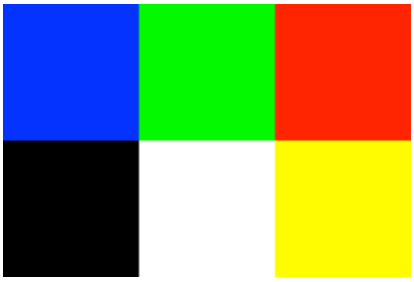
- Defina uma estrutura guardar informação das componentes de côr R,G,B de um pixel;
- Defina uma estrutura para guardar os metadados da imagem (i.e., "magic number", dimensões e valor máximo para cor), e os dados da imagem (matriz de pixeis);
- Depois de ler a informação dos metadados do ficheiro de imagem, aloque dinamicamente um "array" de dimensão igual ao número de linhas da imagem. Seguidamente, para cada linha, aloque dinamicamente um array de tamanho igual ao número de colunas para guardar a informação dos pixels de cada linha;
- Faça a leitura dos pixeis para esse array the arrays.

Exemplo de execução 1:

Considere o ficheiro *img.ppm* apresentado acima.
A execução do programa:

```
$ ./ppm_h_flip < img.ppm > img_hflipped.ppm
```

deverá gerar o ficheiro *img_hflipped.ppm*:



cujo conteúdo é:

```
P3
3 2
255
  0   0 255
  0 255   0
255   0   0
  0   0   0
255 255 255
255 255   0
```

Exemplo de execução 2:

Considere o ficheiro *matisse.ppm*:

A execução do programa:

```
$ ./ppm_h_flip matisse.ppm > matisse_hflipped.ppm
```

deverá gerar o ficheiro *matisse_hflipped.ppm*:

Operação 2 - Rotação vertical

Pretende-se que implemente um programa em C que faça uma rotação vertical (rotação de 180 graus em torno de um eixo horizontal) de uma imagem em formato PPM. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).
A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_v_flip`):

```
$ ./ppm_v_flip [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

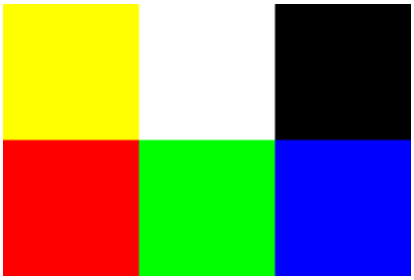
```
$ ./ppm_v_flip < input.ppm > output.ppm
$ ./ppm_v_flip input.ppm > output.ppm
$ ./ppm_v_flip input.ppm output.ppm
```

Exemplo de execução 1:

Considere o ficheiro *img.ppm* apresentado acima.
A execução do programa:

```
$ ./ppm_v_flip img.ppm img_vflipped.ppm
```

deverá gerar o ficheiro *img_vflipped.ppm*:



cujo conteúdo é:

```
P3
3 2
255
255 255 0
255 255 255
0 0 0
255 0 0
0 255 0
0 0 255
```

Exemplo de execução 2:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_h_flip < matisse.ppm > matisse_hflipped.ppm
```

deverá gerar o ficheiro *matisse_vflipped.ppm*:

Operação 3 - Rotação diagonal

Pretende-se que implemente um programa em C que faça uma rotação diagonal (topo-esquerda para fundo-direita) de uma imagem em formato PPM. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).
A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_d_flip`):

```
$ ./ppm_d_flip [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

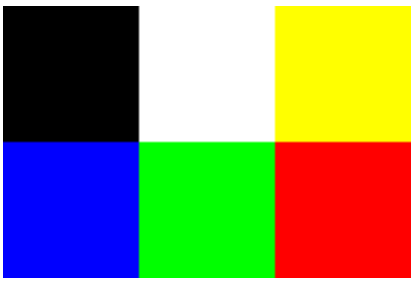
```
$ ./ppm_d_flip < input.ppm > output.ppm
$ ./ppm_d_flip input.ppm > output.ppm
$ ./ppm_d_flip input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *img.ppm* apresentado acima.
A execução do programa:

```
$ ./ppm_d_flip < img.ppm > img_dflipped.ppm
```

deverá gerar o ficheiro *img_dflipped.ppm*:



cujo conteúdo é:

```
P3
3 2
255
0 0 0
255 255 255
255 255 0
0 0 255
0 255 0
255 0 0
```

Exemplo de execução 2:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_d_flip < matisse.ppm > matisse_dflipped.ppm
```

deverá gerar o ficheiro *matisse_dflipped.ppm*:

Operação 4 - Ajuste de componentes de cor RGB

Pretende-se que implemente um programa em C que faça o ajuste das compontentes RGB uma imagem em formato PPM conforme especificado na linha de comandos. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).

Se valor de ajuste for o trip (dR, dG, dB), a nova cor de cada pixel i será (R_i+dR, G_i+dG, B_i+dB).

Note que se o novo valor calculado para uma componente de cor for inferior a 0 ou superior a $MAXCOLOR$, o valor da componente deverá ser 0 ou $MAXCOLOR$, respectivamente.

A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_addRGB`):

```
$ ./ppm_addRGB dR dG dB [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_addRGB +10 -1 +50 < input.ppm > output.ppm
$ ./ppm_addRGB -20 +30 0 input.ppm > output.ppm
$ ./ppm_addRGB 0 0 +100 input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_addRGB +20 +100 -50 matisse.ppm matisse_nRGB.ppm
```

deverá gerar o ficheiro *matisse_nRGB.ppm*:

Operação 5 - Conversão para escala de cinzentos

Pretende-se que implemente um programa em C que faça a transformação uma imagem em formato PPM para escala de cinzentos. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).

À transformação de um pixel `(R,G,B)` (em que R, G, B, correspondem componentes vermelho, verde e azul do pixel) para escala de cinzento, resulta um novo triplo `(G,G,G)`, em que G corresponde à expressão:

$$ G = 0.2126 R + 0.7152 G + 0.0722 B $$

Nota: G deverá ser arredondado para o inteiro mais próximo.

(Consultar Wikipedia - Grayscale para mais detalhes).

A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_grayscale`):

```
$ ./ppm_grayscale [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_grayscale < input.ppm > output.ppm
$ ./ppm_grayscale input.ppm > output.ppm
$ ./ppm_grayscale input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *img.ppm* apresentado acima.
A execução do programa:

```
$ ./ppm_grayscale < img.ppm > img_gray.ppm
```

deverá gerar o ficheiro *img_grey.ppm*:

cujo conteúdo é:

```
P3
3 2
255
54 54 54
182 182 182
18 18 18
237 237 237
255 255 255
0 0 0
```

Exemplo de execução 2:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_grayscale < matisse.ppm > matisse_grey.ppm
```

deverá gerar o ficheiro *matisse_grey.ppm*:

Operação 6 - Conversão para duas cores (Preto e Branco)

Pretende-se que implemente um programa em C que faça a transformação de uma imagem em formato PPM para preto e branco (duas cores). A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).

Para tal, deverá calcular para cada pixel o valor de cinzento G, e, caso $G > threshold$ a cor do pixel será (MAXCOLOR, MAXCOLOR, MAXCOLOR), ou, caso contrário, a cor do pixel será (0, 0, 0). $threshold$, será um parâmetro do programa que deverá estar compreendido entre 0 e $MAXCOLOR$. $MAXCOLOR$ é o valor máximo para cor, especificado no ficheiro ppm).

A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_bw`):

```
$ ./ppm_bw threshold [input.ppm [output.ppm]]
```

em que $0 <= threshold <= MAXCOLOR$.

Exemplos de invocação do aplicativo:

```
$ ./ppm_bw 64 < input.ppm > output.ppm
$ ./ppm_bw 77 input.ppm > output.ppm
$ ./ppm_bw 100 input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_bw 64 matisse.ppm matisse_bw.ppm
```

deverá gerar o ficheiro *matisse_bw.ppm*:

Operação 7 - Rotação 90º Direita

Pretende-se que implemente um programa em C que faça uma rotação de **90 graus para a direita** a uma imagem em formato PPM. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).
A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_rot_right`):

```
$ ./ppm_rot_right [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_rot_right < input.ppm > output.ppm
$ ./ppm_rot_right input.ppm > output.ppm
$ ./ppm_rot_right input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_rot_right matisse.ppm matisse_rot_right.ppm
```

deverá gerar o ficheiro *matisse_rot_right*:

Operação 8 - Rotação 90º Esquerda

Pretende-se que implemente um programa em C que faça uma rotação de **90 graus para a esquerda** a uma imagem em formato PPM. A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).
A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_rot_left`):

```
$ ./ppm_rot_left [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_rot_left < input.ppm > output.ppm
$ ./ppm_rot_left input.ppm > output.ppm
$ ./ppm_rot_left input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_rot_left matisse.ppm matisse_rot_left.ppm
```

deverá gerar o ficheiro *matisse_rot_left.ppm*:

Operação 9 - Adicionar moldura

Pretende-se que implemente um programa em C que adicione à imagem uma moldura. A moldura terá um tamanho fixo e um cor especificados na linha de comando.

A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos). A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada. .

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_addBorder`):

```
$ ./ppm_addBorder npixeis R G B [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_addBorder < input.ppm > output.ppm
$ ./ppm_addBorder input.ppm > output.ppm
$ ./ppm_addBorder input.ppm output.ppm
```

Exemplo de execução:

Considere o ficheiro *matisse.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_addBorder 2 255 255 0 matisse.ppm matisse_moldura.ppm
```

deverá gerar o ficheiro *matisse_moldura.ppm*:

Neste exemplo é adicionada uma moldura à imagem com o tamanho de 2 pixels em toda a sua volta. A cor especificada é R=255, G=255, B=0 (cor amarela).

Operação 10 - Justaposição de imagens

Pretende-se que implemente um programa em C que, dadas duas imagens de entrada, gere uma nova imagem que resulta da justaposição horizontal das imagens de entrada. A largura e altura da imagem resultante deverão ser iguais à soma das larguras e ao mínimo das alturas das imagens originais.

A imagem resultante deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada. .

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_concat`):

```
$ ./ppm_concat input1.ppm input2.ppm [output.ppm]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_concat input1.ppm input2.pmm > output.ppm
$ ./ppm_concat input1.ppm input2.pmm output.ppm
```

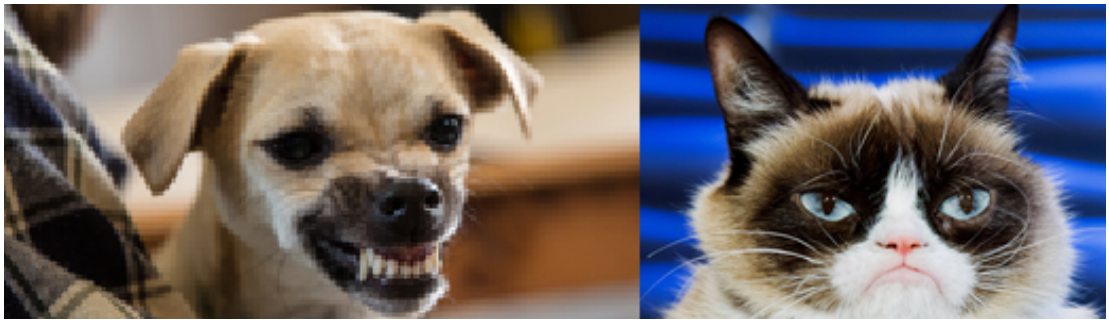
Exemplo de execução:

Considere os ficheiros *cao.ppm* e *gato.ppm*:

A execução do programa:

```
$ ./ppm_concat cao.ppm gato.ppm caoEgato.ppm
```

deverá gerar o ficheiro *caoEgato.ppm*:



Operação 11 - Recortar imagem

Pretende-se que implemente um programa em C que, dada uma imagem em formato PPM, gere uma nova imagem correspondente a um recorte rectangular da imagem original. A informação da zona da imagem a recortar será dada indicando as coordenadas (A, B) do canto superior esquerdo e (C,D) do canto inferior direito do rectângulo.

A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).

A imagem recortada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_bw`):

```
$ ./ppm_crop A B C D [input.ppm [output.ppm]]
```

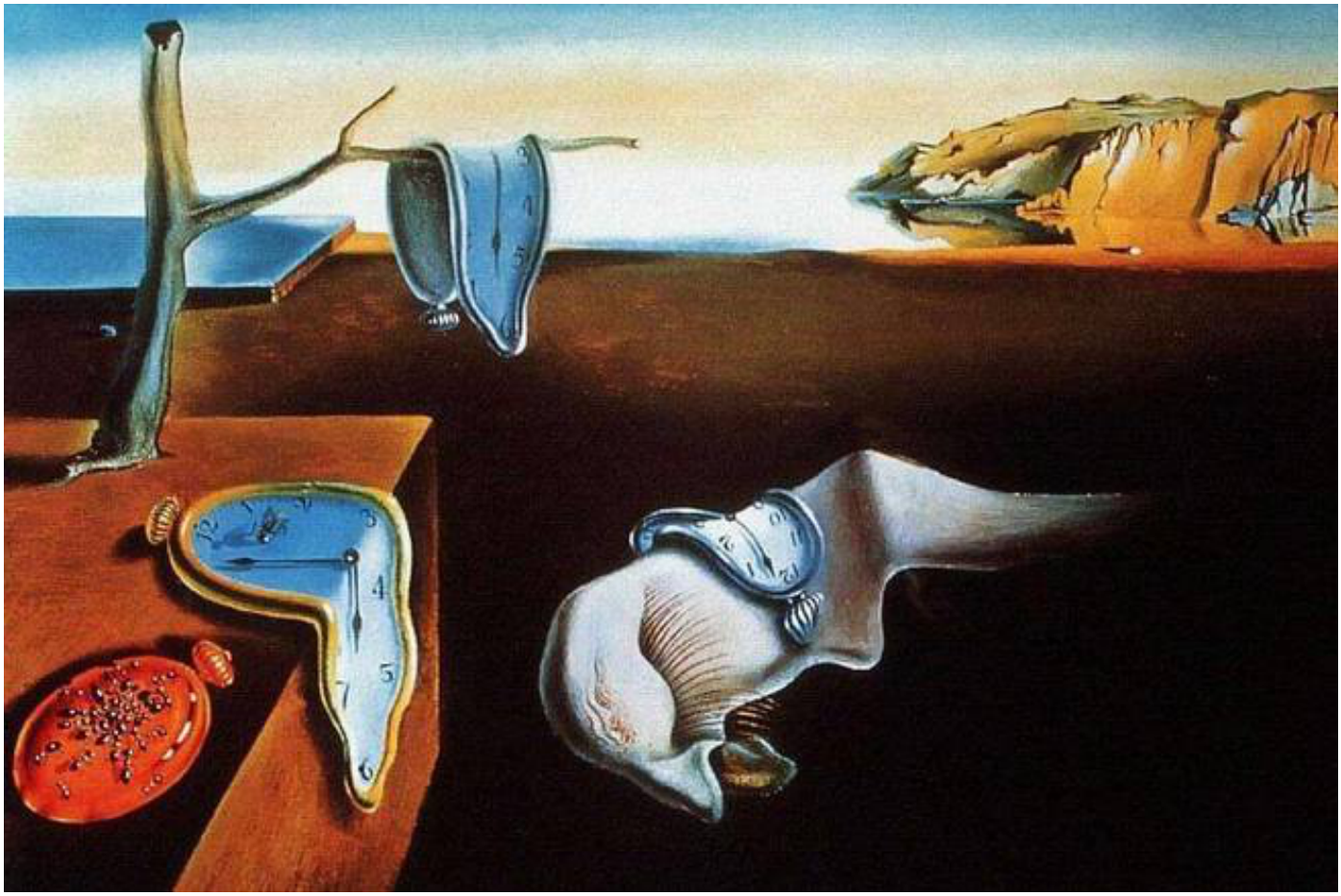
em que (A, B) e (C, D) correspondem às coordenadas do canto superior esquerdo e do canto inferior direito do rectângulo a recortar.

Exemplos de invocação do aplicativo:

```
$ ./ppm_crop 0 0 100 50 < input.ppm > output.ppm
$ ./ppm_crop 20 10 30 30 input.ppm > output.ppm
$ ./ppm_crop 100 100 200 200 input.ppm output.ppm
```

Exemplo de execução:

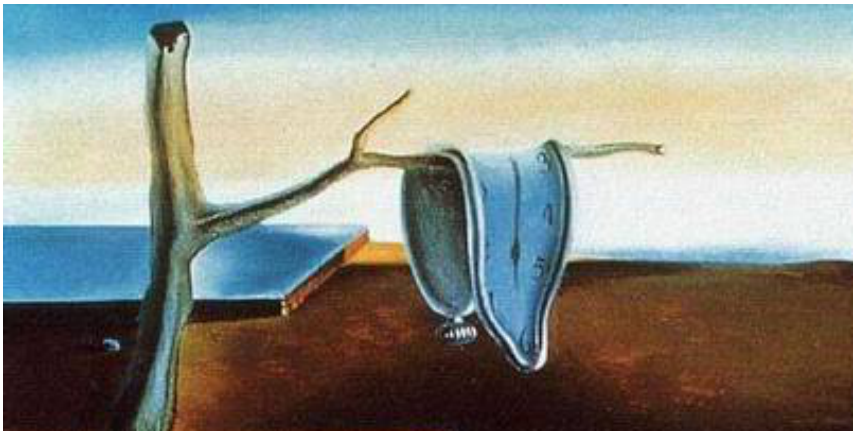
Considere o ficheiro *persistencia_da_memoria.ppm*:



A execução do programa:

```
$ ./ppm_crop 0 0 400 200 persistencia_da_memoria.ppm pm_crop_0_0_400_200.ppm
```

deverá gerar o ficheiro *pm_crop_0_0_400_200.ppm*:



Operação 12 - Sobreposição de imagens

Pretende-se que implemente um programa em C que dadas duas imagens em formato PPM, gere uma nova imagem correspondente à sobreposição de uma imagem sobre a outra a começando num dado ponto da (X, Y) primeira imagem.

No input, serão indicados X, Y e o nome do ficheiro da imagem a sobrepor. A imagem base poderá ser lida da entrada padrão, ou de um ficheiro indicado como argumento. A imagem resultante deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada. .

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_overlay`):

```
$ ./ppm_overlay X Y overlay.ppm [input.ppm [output.ppm]]
```

Explicação: O ficheiro ouput.ppm corresponderá à imagem resultante de sobrebor a imagem overlay.ppm sobre a imagem input.ppm no ponto (X,Y) de input.ppm.

Exemplos de invocação do aplicativo:

```
$ ./ppm_overlay 10 20 overlay.ppm < input.pmm > output.ppm
$ ./ppm_overlay 10 20 overlay.ppm input.pmm > output.ppm
$ ./ppm_overlay 10 20 overlay.ppm input.pmm output.ppm
```

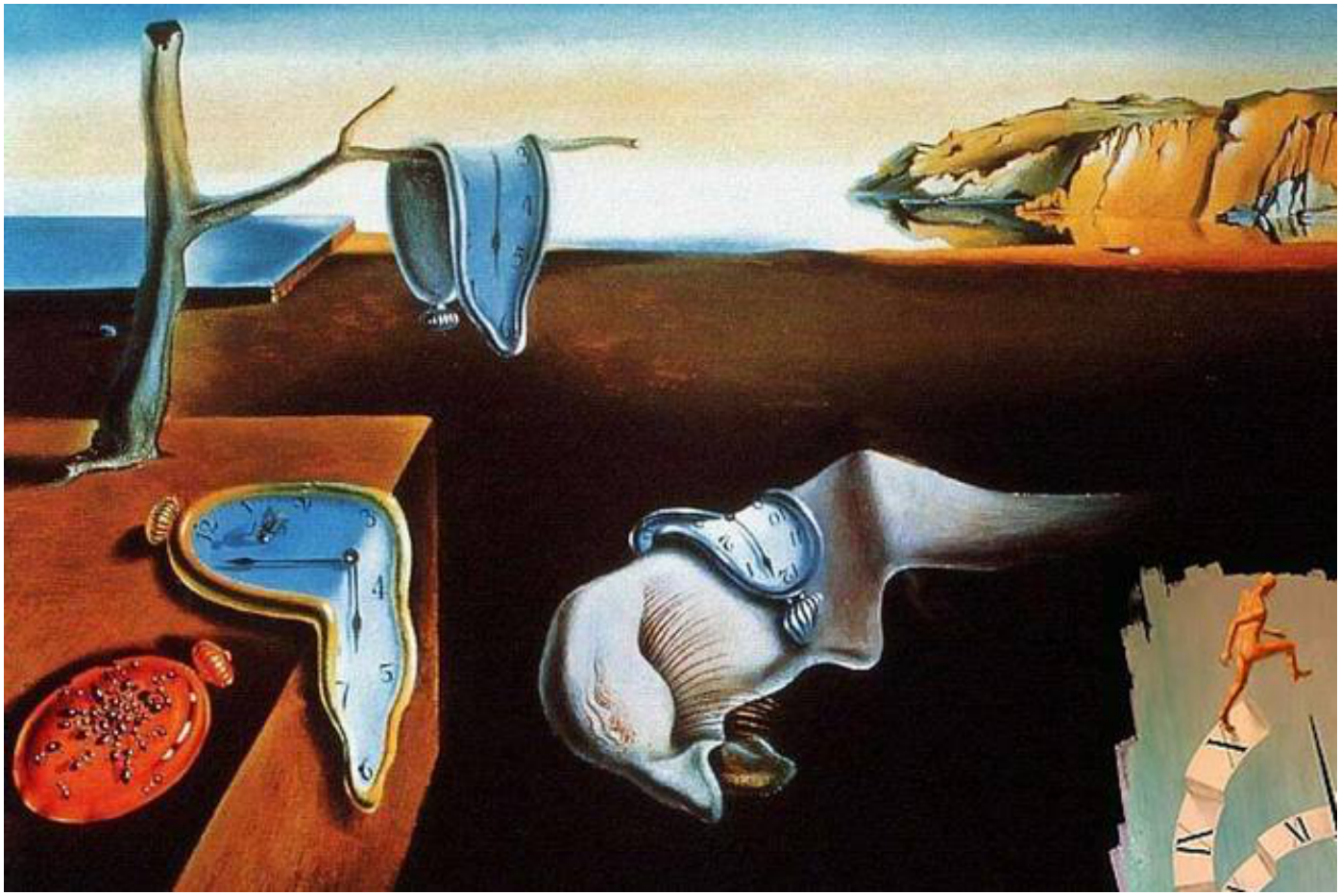
Exemplo de execução:

Considere os ficheiros *persistencia_da_memoria.ppm* indicado anteriormente, e o ficheiro e *clock.ppm*:

A execução do programa:

```
$ ./ppm_overlay 540 270 clock.ppm < persistencia_da_memoria.ppm > pm_clock.ppm
```

deverá gerar o ficheiro *pm_clock.ppm*:



Operação 13 - Sobreposição de imagens com efeito transparência

Pretende-se que implemente um programa em C que dadas duas imagens em formato PPM, gere uma nova imagem correspondente à sobreposição, com transparência, de uma imagem sobre a outra a começando num dado ponto da (X, Y) primeira imagem. A sobreposição das imagens deverá implementar o efeito de transparência: cada componente C_R de cor do pixel resultante da sobreposição corresponderá a uma média pesada das componentes de cor C_1 e C_2 dos pixeis correspondentes das imagens originais.

$$C_R = \alpha C_1 + (1-\alpha) C_2$$

em que $0 <= \alpha <= 1$.

No input, serão indicados α, X, Y, e o nome do ficheiro da imagem a sobrepor. A imagem base poderá ser lida da entrada padrão, ou de um ficheiro indicado como argumento. A imagem resultante deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada. .

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_owt`):

```
$ ./ppm_owt alpha X Y overlay.ppm [input.ppm [output.ppm]]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_owt 0.2 10 20 overlay.ppm < input.pmm > output.ppm
$ ./ppm_owt 0.2 10 20 overlay.ppm input.pmm > output.ppm
$ ./ppm_owt 0.2 10 20 overlay.ppm input.pmm output.ppm
```

Operação 14 - Pesquisa em imagens

Pretende-se que implemente um programa em C que dadas duas imagens I_1, I_2, em formato PPM, procure todas as ocorrências de I_2 em I_1. Para cada ocorrência, o programa deverá devolver as coordenadas (X, Y) do canto superior esquerdo onde se inicia a sobreposição de I_2 com I_1.

Para este enunciado, para considerarmos que uma imagem I_2 está contida em I_1, todos os pixeis de I_2, a partir de uma dado pondo (X,Y) terão que ser aproximadamente (a menos de uma de uma tolerância δ) iguais aos correspondente de I_1. Ou seja, dadas as componentes de cor de dois pixeis C_1 e C_2, $C_2- \delta <= C_1 <= C_2 + \delta$;

No input, serão indicados δ e o nome do ficheiro da imagem a procurar. A imagem base poderá ser lida da entrada padrão, ou de um ficheiro indicado como argumento. As coordenadas de cada ponto onde se detetou encontrou a imagem a procurar deverá ser impressos na saída padrão.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada. .

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_search`):

```
$ ./ppm_search delta image2.ppm [image1.ppm]
```

Exemplos de invocação do aplicativo:

```
$ ./ppm_overlay 5 image2.ppm < image1.pmm
$ ./ppm_overlay 30 image2.ppm image1.pmm
```

Operação 15 - Equalização de histograma

Pretende-se que implemente um programa em C que, dada uma imagem em formato PPM, gere uma nova imagem com um ajuste de contraste utilizando a técnica de equalização de histograma (ver https://en.wikipedia.org/wiki/Histogram_equalization).

As imagens utilizadas estão em escala de cinza (i.e as componente R=G=B). Para simplificar pode apenas considerar a informação de um dos canais/côr para o cálculo. Para aplicar o algoritmo de equalização de histograma deverá seguir os seguintes passos:

1. Calcular o histograma da imagem utilizando apenas um dos canais de cor;
2. Calcular a função de distribuição acumulada a partir do histograma da imagem;
3. Calcular os valores v para o histograma equalizado utilizando a seguinte equação:

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

   Em que:

   - Cdfmin é o menor valor, diferente de zero, da função de distribuição acumulada
   - M x N é o número de píxeis da imagem
   - L é o valor MaxVal especificado no cabeçalho do ficheiro PPM.

4. Calcular as cores da imagem tendo em conta o histograma equalizado.

Para mais informações visitar o seguinte link: https://en.wikipedia.org/wiki/Histogram_equalization

A imagem poderá ser lida da entrada padrão ou de um ficheiro de input (caso seja especificado na linha de comandos).

A imagem transformada deverá ser enviada para a saída padrão, ou então, para um ficheiro de output (caso seja especificado na linha de comandos). O ficheiro PPM de output deverá conter apenas **um pixel por linha**, e **não deverá ter comentários**.

Nota: Caso não seja possível abrir o(s) ficheiro(s), deverá ser gerada uma mensagem de erro adequada.

Sinopse (assumindo que a aplicação desenvolvida se chama `ppm_eh`):

```
$ ./ppm_eh [input.ppm [image1.ppm]]
```

Exemplos de invocação do aplicativo:

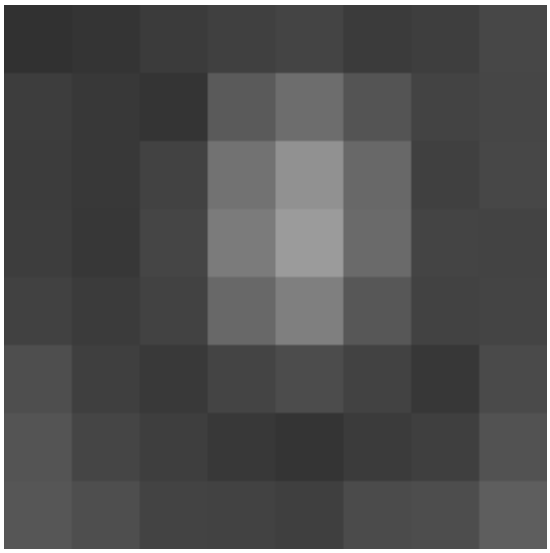
```
$ ./ppm_eh < input.ppm > output.ppm
$ ./ppm_eh input.ppm > output.ppm
$ ./ppm_eh input.ppm output.ppm
```

Exemplo:

Este exemplo foi retirado do wikipedia:

(https://en.wikipedia.org/wiki/Histogram_equalization)

Considere a imagem (8x8 píxeis):



A que correspondem os seguintes valores de cinzento:

52	**55**	**61**	**59**	**79**	**61**	**76**	**61**
62	59	55	104	94	85	59	71
63	65	66	113	144	104	63	72
64	70	70	126	154	109	71	69
67	73	68	106	122	88	68	68
68	79	60	70	77	66	58	75
69	85	64	58	55	61	65	83
70	87	69	68	65	73	78	90

A tabela seguinte apresenta o histograma correspondente (os valores de pixel que têm uma contagem zero estão excluídos):

Value	Count	Value	Count	Value	Count	Value	Count	Value	Count
52	1	64	2	72	1	85	2	113	1
55	3	65	3	73	2	87	1	122	1
58	2	66	2	75	1	88	1	126	1
59	3	67	1	76	1	90	1	144	1
60	1	68	5	77	1	94	1	154	1
61	4	69	3	78	1	104	2		
62	1	70	4	79	2	106	1		
63	2	71	2	83	1	109	1		

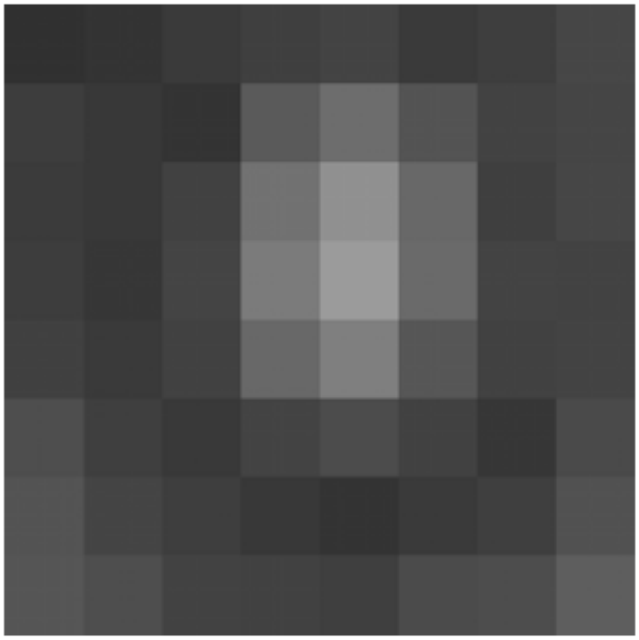
A função de distribuição cumulativa, bem como o resultado da função de mapeamento são mostrados na seguinte tabela (os valores de côr sem ocorrências na imagem original não são apresentados):

v, Pixel Intensity	cdf(v)	h(v), Equalized v
52	1	0

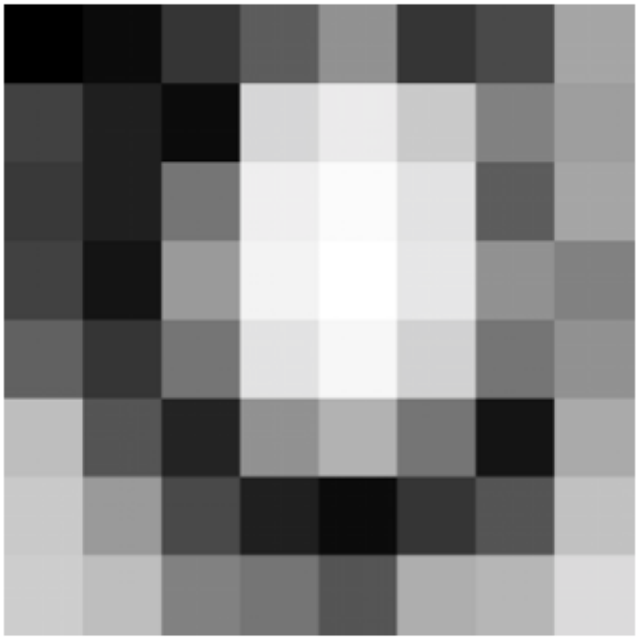
55	4	12
58	6	20
59	9	32
60	10	36
61	14	53
62	15	57
63	17	65
64	19	73
65	22	85
66	24	93
67	25	97
68	30	117
69	33	130
70	37	146
71	39	154
72	40	158
73	42	166
75	43	170
76	44	174
77	45	178
78	46	182
79	48	190
83	49	194
85	51	202
87	52	206
88	53	210
90	54	215
94	55	219
104	57	227
106	58	231
109	59	235
113	60	239
122	61	243
126	62	247
144	63	251
154	64	255

Os valores da imagem equalizada são obtidos da tabela de mapeamento acima:

0	12	53	32	190	53	174	53
57	32	12	227	219	202	32	154
65	85	93	239	251	227	65	158
73	146	146	247	255	235	154	130
97	166	117	231	243	210	117	117
117	190	36	146	178	93	20	170
130	202	73	20	12	53	85	194
146	206	130	117	85	166	182	215



Original



Equalized