

# Laboratório de Computadores - Projecto Processamento de Imagem

---

## Preâmbulo

---

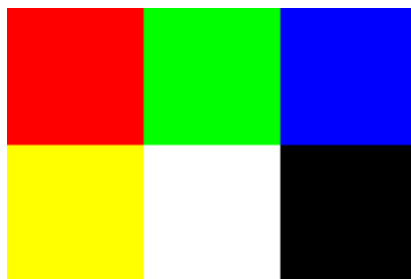
Pretende-se que implemente um conjunto de operações de processamento de imagem em formato de armazenamento *ASCII/Plain Portable Pixel Map* (PPM).

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:**

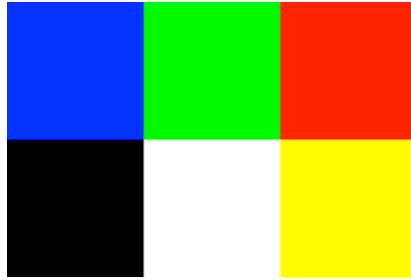
- Defina uma estrutura guardar informação das componentes de cor R,G,B de um pixel;
- Defina uma estrutura para guardar os metadados da imagem (i.e., "magic number", dimensões e valor máximo para cor), e os dados da imagem (matriz de pixels);
- Depois de ler a informação dos metadados do ficheiro de 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 pixels 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:

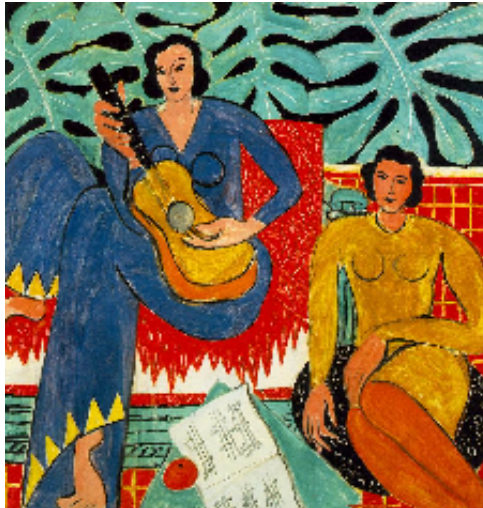
Considere o ficheiro *matisse.ppm*:



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:**

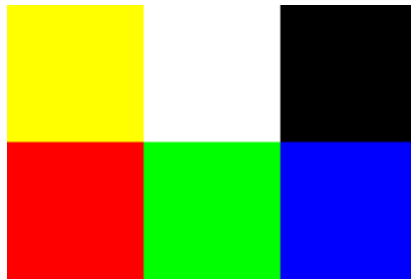
```
$ ./ppm_v_flip < input.ppm > output.ppm
$ ./ppm_v_flip input.ppm > output.ppm
$ ./ppm_v_flip input.ppm output.ppm
```

### 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
```

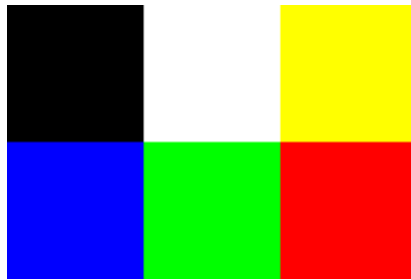
### Exemplo de execução:

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 componentes 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 triplo  $(dR, dG, dB)$ , 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](https://en.wikipedia.org/wiki/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  $G > \text{threshold}$  a cor do pixel será (MAXCOLOR, MAXCOLOR, MAXCOLOR), ou, caso contrário, a cor do pixel será (0, 0, 0).  $\text{threshold}$ , 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  $0 \leq \text{threshold} \leq \text{MAXCOLOR}$ .

**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 90° 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 90° 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_add_Border`):**

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

**Exemplos de invocação do aplicativo:**

```
$ ./ppm_add_Border < input.ppm > output.ppm
$ ./ppm_add_Border input.ppm > output.ppm
$ ./ppm_add_Border 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]
```

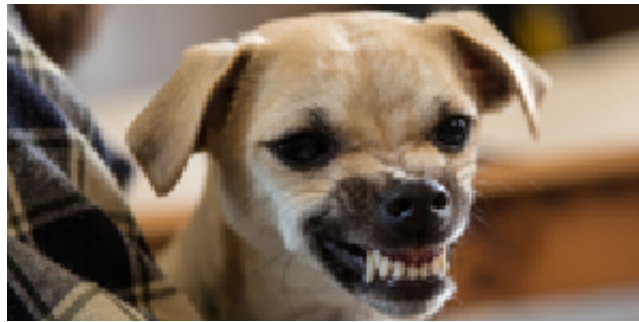
**Exemplos de invocação do aplicativo:**

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



### Exemplo de execução:

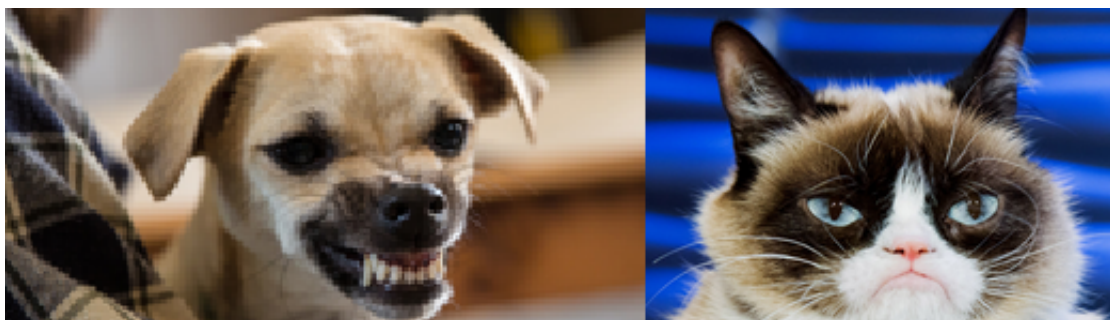
Considere os ficheiros *cao.ppm* e *gato.ppm*:



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]]
```

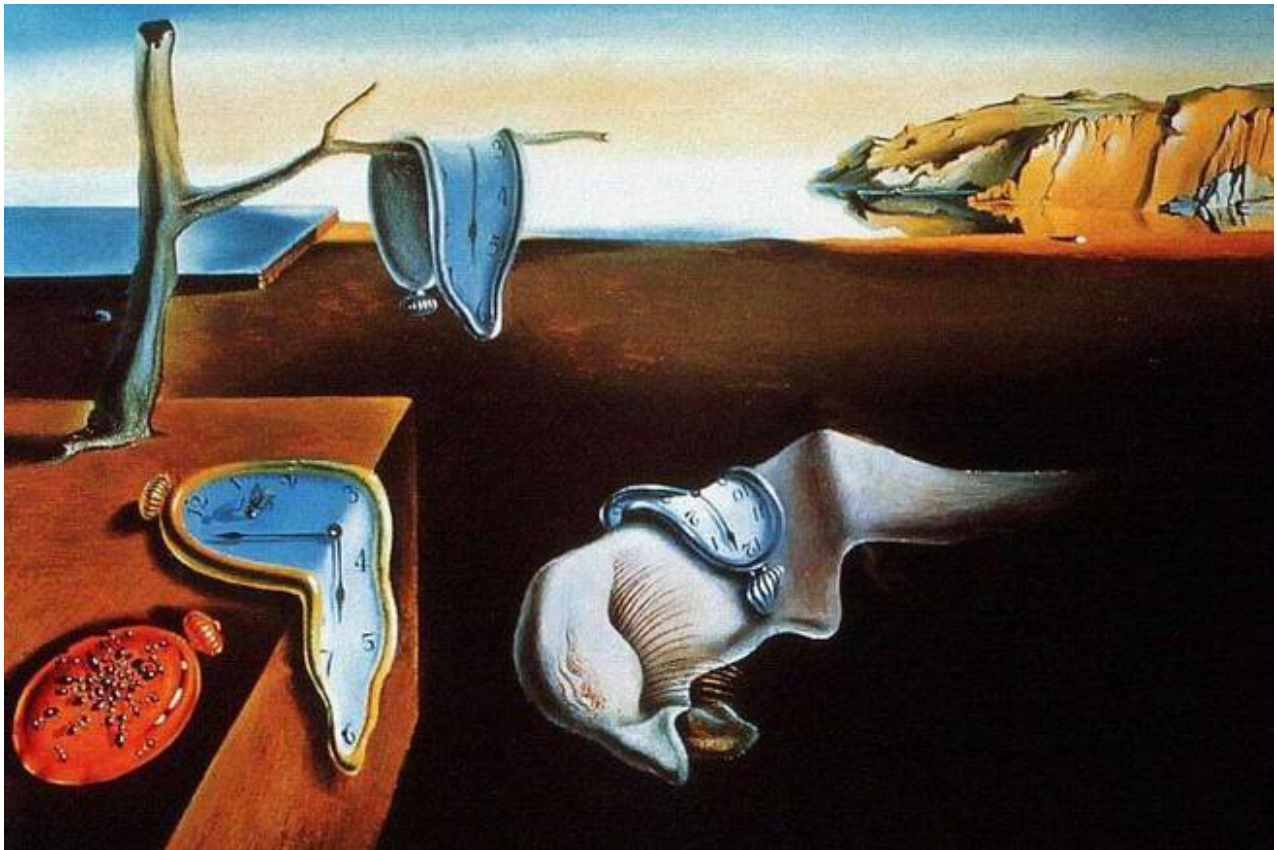
em que \$(A, B)\$ e \$(C, D)\$ correspondem às coordenadas do canto superior esquerdo e do canto inferior direito do rectângulo a recortar.

**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:**

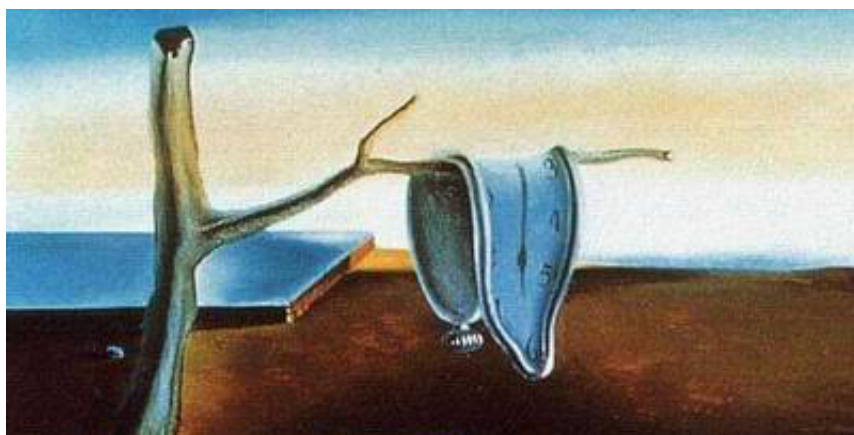
Considere o ficheiro *persistencia\_da\_memoria.ppm*:



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 imagem 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 output.ppm corresponderá à imagem resultante de sobrepor 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.ppm > output.ppm
$ ./ppm_overlay 10 20 overlay.ppm input.ppm > output.ppm
$ ./ppm_overlay 10 20 overlay.ppm input.ppm output.ppm
```

**Exemplo de execução:**

Considere os ficheiros *persistencia\_da\_memoria.ppm* indicado anteriormente, e o ficheiro *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*:



