Laboratório de Computadores - Projecto Processamento de Imagem (1ª parte)

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:

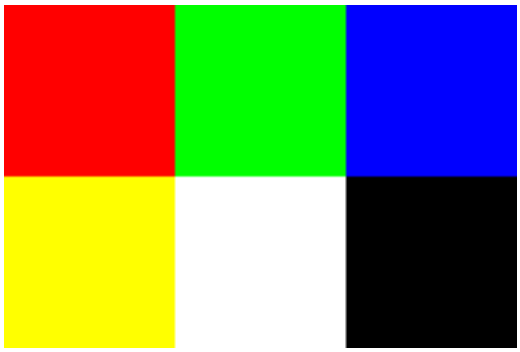

Figure 1: Ficheiro img.ppm (ampliado)

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
```

Nesta primeira parte do projeto, deverá implementar algumas operações de ro-

tação da imagem e de transformação de côr: (1) rotação horizontal, (2) rotação vertical, (3) rotação diagonal, (4) conversão para escala de cinzentos, (5) conversão para preto e branco, e (6) ajuste de componentes RGB.

Operação 1 - Rotação horizontal

Pretende-se que implemente um programa em C que faça uma rotação horizontal de 180 graus 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 o a aplicação desenvolvida se chama "):

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

deverá gerar o ficheiro *img_hflipped.ppm*:

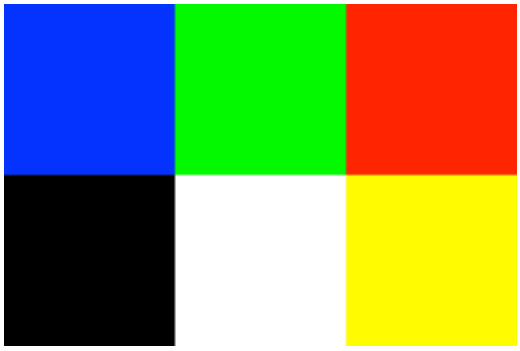

Figure 2: Ficheiro img_flipped.ppm (ampliado)

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_rot 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 de 180 graus 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**.

Figure 3: matisse.ppm

Figure 4: matisse_hflipped.ppm

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

Sinopse (assumindo que o 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: Considere o ficheiro *img.ppm* apresentado acima. A execução do programa:

```
$ ./ppm_v_flip_rot img.ppm img_vflipped.ppm
```

deverá gerar o ficheiro *img_vflipped.ppm*:

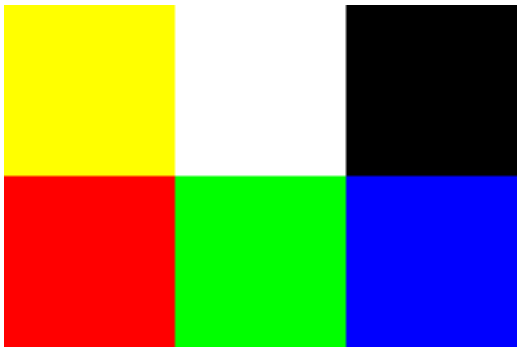

Figure 5: Ficheiro img_vflipped.ppm (ampliado)

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_rot < matisse.ppm > matisse_hflipped.ppm
```

deverá gerar o ficheiro *matisse_vflipped.ppm*:

Figure 6: 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 o 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_rot < img.ppm > img_dflipped.ppm
```

deverá gerar o ficheiro *img_dflipped.ppm*:

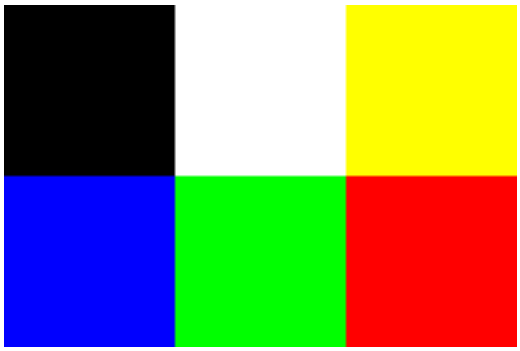

Figure 7: Ficheiro img_vflipped.ppm (ampliado)

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_rot < matisse.ppm > matisse_dflipped.ppm
```

deverá gerar o ficheiro *matisse_dflipped.ppm*:

Figure 8: 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 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 o 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*:

Figure 9: 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:

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

(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 o 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
0 0 0
255 255 255
255 255 0
0 0 255
0 255 0
255 0 0
```

Figure 10: Ficheiro img_grey.ppm (ampliado)

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 compreendico 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 o a aplicação desenvolvida se chama `ppm_bw`):

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

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

Figure 11: matisse_grey.ppm

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*:

Figure 12: matisse_bw.ppm