Etapa 2 - Criação de imagens no formato PPM

Para a realização desta etapa, você deve estar a par dos seguintes elementos:

- Implementação de classes em .h e .cpp
- Escrita em arquivos
- Alocação dinâmica

Conceitos iniciais

Imagens digitais são normalmente representadas de uma das duas formas: ou vetorial ou *raster*. A primeira usa figuras geométricas, como polígonos, circunferências entre outros, para definir objetos vetoriais a serem usados na composição da imagem. Um exemplo disso é o formato SVG.

A segunda forma é uma representação em que há uma correspondência de cada pixel da imagem a um conjunto de informações (como cor, transparência etc). Dizemos que esta última forma é uma representação “ponto-a-ponto” da imagem, normalmente realizada através de uma matriz onde cada célula contém a informação de um ponto (pixel), também conhecida como “mapa de bits” ou bitmap.

Quando desejamos armazenar, transferir ou imprimir uma imagem, é necessário armazená-la em um arquivo. Porém, a representação da imagem no arquivo não utiliza necessariamente a mesma representação da imagem em memória. Enquanto a primeira procura facilitar e tornar as operações de manipulação da imagem eficientes, a segunda é normalmente voltada à compactação da imagem (tamanho do arquivo) com ou sem perdas de qualidade. Por exemplo, imagens no formato JPEG ou PNG possuem algoritmos específicos para comprimir os dados dos pixels... e são bastante complexos. Há entretanto um formato simples para armazenar uma imagem em um arquivo, que não utiliza compressão alguma, chamado PPM (*Portable Pixel Map*).

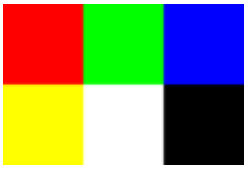
Formato PPM

O formato de imagem PPM possui duas versões: uma textual e outra binária. A versão textual, que será usada neste projeto, armazena os valores de cada pixel em um arquivo de texto, onde cada pixel é definido por três valores, indicando respectivamente a intensidade das cores vermelho, verde e azul (R, G, e B).

O arquivo PPM obedece a uma formação específica composta por um cabeçalho, com informações sobre a imagem em geral, seguido de uma sequência de triplas RGB para cada um dos pixels. O cabeçalho possui 4 informações. A primeira é um código específico (string) informando que se trata de uma imagem PPM e de sua versão (no nosso caso, será "P3"). A segunda e a terceira definem as dimensões da imagem (largura e altura, nessa

ordem). Por fim, a última informação do cabeçalho é o valor máximo dos componentes RGB dos pixels.

Para ilustrar melhor, a figura e conteúdo a seguir exemplificam uma imagem (ampliada para ser melhor exibida) com dimensões de 3x2 representada no formato PPM na versão textual (P3).



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

A primeira linha contém o identificador P3. Esse identificador é o código que indica aos programas de imagem que os próximos dados seguem o formato textual do PPM. Em seguida, os valores 3 e 2, separados por espaço, indicam que a imagem tem 3 colunas (largura) e 2 linhas (altura). Na terceira linha está o valor 255. Esse valor é usado para informar o valor máximo de cada componente do pixel. Ou seja, os valores vão de 0 a 255. Em seguida, a cada 3 valores, um pixel é definido com a cor dos valores (RGB).

Obs: Os espaços entre os valores RGB de cada pixel e os saltos de linha são puramente estéticos (facilita identificar as linhas e pixels). Para o formato, esses espaços não fazem diferença. A imagem acima pode ser representada pelo código abaixo sem perda alguma de informação.

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

Implementação

Defina um novo tipo de dado capaz de representar uma imagem. Esse tipo precisa ter as dimensões da imagem (largura e altura), bem como uma matriz de pixels. Cada pixel deve representar um ponto de cor na imagem (ou seja, um pixel deve ser do tipo Cor, definido na etapa 1 do projeto).

Sua implementação deve separar a definição da classe em um arquivo .h e a implementação dos métodos no arquivo .cpp correspondente.

O tipo imagem deve ter um construtor que recebe as dimensões da imagem e alocar dinamicamente sua matriz de pixels (com todos os valores 0, ou seja preto). O código abaixo ilustra a definição de uma imagem de tamanho 10x15. Veja que apenas as dimensões são passadas e a matriz de pixels é alocada e inicializada internamente no construtor.

```
Imagem img(10,15);
```

Lembre-se que, como o construtor realiza uma alocação de memória, seu destrutor deve liberar a memória alocada.

Além do construtor e destrutor, o tipo de imagem deve fornecer as seguintes operações (métodos):

- um método para consultar o pixel que se encontra em uma dada coordenada da matriz;
- um método para definir a cor do pixel de uma dada coordenada da matriz;
- um método para salvar a imagem no formato PPM (P3), dado o nome do arquivo.

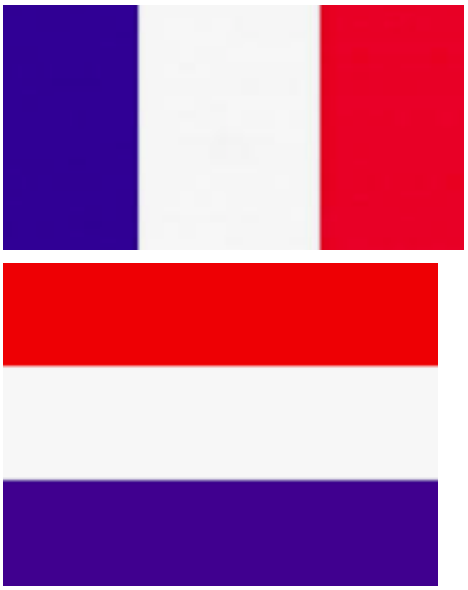
A forma que a matriz vai ser alocada internamente no tipo deve ser transparente para quem vai usar o tipo. Assim, é possível alocar tanto uma matriz propriamente dita quanto um único array. Por exemplo, para uma matriz `NxM`, pode-se alocar dinamicamente `N` linhas, cada linha contendo `M` colunas, mas também pode-se alocar um único array de tamanho `NxM` e acessar a célula (`lin`, `col`) através da fórmula: `lin * M + col`.

Testes

Para verificar se a classe de imagem está funcionando corretamente, escreva testes para verificar se:

- Após criar uma imagem 2x2, os 4 pixels estarão com valores (0, 0, 0).
- Após criar uma imagem 3x4 e setar todos os pixels com valores (255, 255, 255), a consulta aos pixels retornam de fato os valores (255, 255, 255).
- Após criar a imagem do teste anterior e salvar para um arquivo no formato PPM, é possível abrir a imagem e ver que a imagem está toda branca.

Após esses testes, você pode "brincar" com a classe criando bandeiras como a da França ou a da Holanda (abaixo).



Obs: O formato PPM é "nativo" do Linux, mas não é do Windows. Então, se você estiver desenvolvendo no Windows, não conseguirá ver imagens PPM. Você pode baixar um software de visualização como o IrfanView ou instalar no VisualCode a extensão PBM/PPM/PGM Viewer for Visual Studio Code.