2022_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA_TN - METATURMA

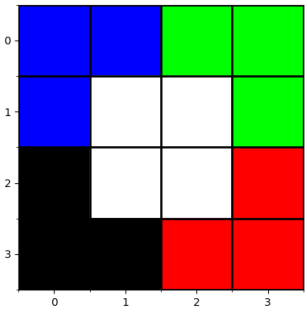
L01E08 - Imagem (4,0 pts)

Data de entrega: terça, 17 Mai 2022, 23:59
Arquivos requeridos: main.cpp, Pixel.hpp, Pixel.cpp, Imagem.hpp, Imagem.cpp (Baixar)
Tipo de trabalho: Trabalho individual

O objetivo desse exercício é praticar a parte básica de **Orientação à Objetos**.

Você deve fazer um programa que irá fazer o processamento de imagens, aplicando diferentes operações. A maneira mais simples de representar uma imagem é usando uma **grade de pixels**, onde cada posição guarda informações de cor. Uma cor pode ser representada considerando três canais: R (vermelho), G (verde), B (azul), cada canal com uma precisão de 8bits, ou seja, com valores entre 0 e 255. A figura abaixo ilustra uma imagem bem simples e os valores associais para os canais.

```
000000255 000000255 000255000 000255000
000000255 255255255 255255255 000255000
000000000 255255255 255255255 255000000
000000000 000000000 255000000 255000000
```



Você deve implementar duas **classes**: **Pixel** e **Imagem**. Cada classe deve seguir as especificações abaixo:

- **Pixel:**

1. Atributos: int _red, _green, _blue.
2. `Pixel(int red, int green, int blue)`: Método construtor para a inicialização dos atributos. Os parâmetros devem possuir valor padrão 0 (ou seja, deve ser possível instanciar um objeto sem passar parâmetros caso desejado).
3. `void set_rgb(int red, int green, int blue)`: Atribui valores específicos a cada um dos atributos da classe.
4. `std::vector<int> get_rgb()`: Retorna os atributos da classe em um vetor na ordem RED, GREEN e BLUE.
5. `void print()`: Faz a impressão dos atributos no seguinte formato: "**rrrgggbbb** ". cada atributo deve ser formatado para ser impresso considerando **3 casas** e todos **concatenados** , por exemplo, o valor '5' seria impresso como '005' e o valor '12' como '012'. Não deve-se imprimir quebra de linha ao final. Dica: veja os comandos setw/setfill.

- **Imagem:**

1. Atributos: int _width, _height. Além disso, você deve escolher a melhor maneira para guardar uma coleção de Pixels (dica: organize em uma estrutura matricial para facilitar).
2. `void initialize_image(int width, int height)`: Método que inicializa a estrutura que será utilizada para guardar todos os Pixels da imagem. A estrutura deve respeitar a largura (width) e altura (height) informados. Além disso, conforme falado, os Pixels devem ser inicializados com valores 0 para os três atributos.
3. `void fill(int row, int col, std::string pixel)`: Preenche os atributos do Pixel que está em uma determinada linha (row) e coluna (col) com o valor informado (pixel). Você deve manipular a string para recuperar cada valor individualmente.
4. `void to_grayscale()`: Transforma a imagem original colorida em uma imagem em escala de cinza. Essa operação é feita atribuindo-se o mesmo valor para todos os atributos do Pixel. Esse valor deve ser a média dos três valores de cor atuais (RGB) do pixel.

5. `void grayscale_thresholding(int lim)`: Faz a binarização de uma imagem que está em escala de cinza. Se o valor médio do Pixel for maior que um certo limiar (lim) ele recebe o valor branco (todos os atributos 255), caso contrário recebe o valor preto (todos os atributos 0).
6. `void show()`: Faz a impressão a impressão de todos os Pixels da Imagem de acordo com a largura e altura. Os valores devem ser separados com um espaço em branco (" ") e deve haver uma quebra de linha ao final de cada linha. Semelhante à entrada.

Todos os **atributos** devem ser **privados** e acessados/manipulados fora das classes apenas através de métodos. Você é livre para adicionar quaisquer outros atributos ou métodos auxiliares que julgar necessário.

Por fim, você deve implementar o arquivo **main.cpp** e adicionar toda a parte de entrada/saída que será responsável por manipular os seguintes comandos:

- '**`i width height`**': comando para inicializar uma imagem.
- '**`p`**': comando para preencher uma imagem de acordo com uma matriz de pixels que será informada. Essa matriz possui 'width' colunas e 'height' linhas, e cada pixel está no formato 'rrrgggbbb'.
- '**`s`**': exibe a imagem seguindo formato similar ao de entrada.
- '**`g`**': transforma a imagem original colorida em uma imagem em escala de cinza.
- '**`t lim`**': faz a binarização de uma imagem que está em escala de cinza de acordo com o lim informado.
- '**`b`**': deve chamar a função '**avaliacao_basica()**' implementada no arquivo "**avaliacao_basica_imagem.hpp**" (já incluído no main.cpp). Essa função faz uma avaliação do código (não apenas dos resultados).

Para ilustrar, abaixo são apresentados dois exemplos de entrada/saída:

```
input =
i 4 4
s
output =
000000000 000000000 000000000 000000000
000000000 000000000 000000000 000000000
000000000 000000000 000000000 000000000
000000000 000000000 000000000 000000000

input =
i 4 4
p
000000255 000000255 000255000 000255000
000000255 255255255 255255255 000255000
000000000 255255255 255255255 255000000
000000000 000000000 255000000 255000000
s
output =
000000255 000000255 000255000 000255000
000000255 255255255 255255255 000255000
000000000 255255255 255255255 255000000
000000000 000000000 255000000 255000000
```

Atenção: Lembre-se de fazer a correta modularização utilizando os arquivos **.hpp** e **.cpp**.

Dica 1:
O código da avaliação básica pode ser copiado aqui, caso você queira depurar algo localmente.

Dica 2:
Você pode usar os códigos dos exercícios anteriores e o da avaliação básica para lhe ajudar a fazer toda a parte de entrada/saída.

Referências:
https://www.cplusplus.com/reference/string/stoi/
https://www.cplusplus.com/reference/string/string/substr/
https://www.cplusplus.com/reference/iomanip/setw/
https://www.cplusplus.com/reference/iomanip/setfill/
https://www.cplusplus.com/reference/stl/

Arquivos requeridos

main.cpp

```cpp
// NÃO ALTERE ESSA LINHA
#include "avaliacao_basica_imagem.hpp"

int main() {


    //
    // Adicione seu código aqui e faça as demais alterações necessárias
    //


    return 0;
}
```

Pixel.hpp

Pixel.cpp

Imagem.hpp

Imagem.cpp

VPL

◄ L01E07 - Controle de versão (2,0 pts)

Seguir para...

L02E01 - Treinador Pokémon (4,0 pts) ►