TRABALHO DA 1ª UNIDADE – COMPUTAÇÃO GRÁFICA.

Relatório do trabalho de implementação dos métodos de manipulação de imagens.

Marcos André Azevedo de Assis.

Departamento de Computação – Universidade do Estado do Rio Grande do Norte (UERN)
Av. Dr.João Medeiros Filho, n° 3419 – Natal – RN – Brasil.

marcosandreazevedo78@gmail.com

- Os códigos dos métodos para manipulação de imagens podem ser encontrados em meu repositório no GitHub: https://github.com/MarcosAndre5/Computacao-Grafica

1. **Binarização:**
   Nesta técnica, se obtém como saída uma imagem com apenas dois níveis de luminância: preto e branco. Aqui estou dividindo os pixels, aqueles que possuem tonalidade menor ou igual à 126 será atualizado para preto (0, 0, 0), e os pixels com tonalidade maior que 126 serão atualizados para branco (255, 255, 255).

```
for(i = 0; i < alt; i++)
    for(j = 0; j < larg; j++){
        if(RGB[i][j].r <= 126){
            RGB[i][j].r = 0;
            RGB[i][j].g = 0;
            RGB[i][j].b = 0;
            fprintf(novaImagem, "\n%d ", RGB[i][j].r);
            fprintf(novaImagem, "%d ", RGB[i][j].g);
            fprintf(novaImagem, "%d ", RGB[i][j].b);
        }else if(RGB[i][j].r > 126){
            RGB[i][j].r = 255;
            RGB[i][j].g = 255;
            RGB[i][j].b = 255;
            fprintf(novaImagem, "\n%d ", RGB[i][j].r);
            fprintf(novaImagem, "%d ", RGB[i][j].g);
            fprintf(novaImagem, "%d ", RGB[i][j].b);
        }
    }
```

   Resultado:

2. **Negativo:**
   O Negativo é usado para inverter as cores de uma imagem.

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        fprintf(novaImagem, "%d ", 255 - RGB[i][j].r);
        fprintf(novaImagem, "%d ", 255 - RGB[i][j].g);
        fprintf(novaImagem, "%d ", 255 - RGB[i][j].b);
    }
    fprintf(novaImagem, "\n");
}
```

Resultado:

3. **Adição:**

A adição é usada para fazer adição em cada pixel de uma imagem com os pixels de uma outra imagem, para gerar uma só imagem resultante.

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        if((RGB[i][j].r + RGB2[i][j].r) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].r + RGB2[i][j].r) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].r + RGB2[i][j].r));

        if((RGB[i][j].g + RGB2[i][j].g) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].g + RGB2[i][j].g) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].g + RGB2[i][j].g));

        if((RGB[i][j].b + RGB2[i][j].b) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].b + RGB2[i][j].b) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].b + RGB2[i][j].b));
    }
    fprintf(novaImagem, "\n");
}
```

Resultado:

4. **Subtração:**
   A subtração é usada para subtrair cada pixel de uma imagem pelos pixels de outra, para gerar uma só imagem resultante.

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        if((RGB[i][j].r - RGB2[i][j].r) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].r - RGB2[i][j].r) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].r - RGB2[i][j].r));

        if((RGB[i][j].g - RGB2[i][j].g) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].g - RGB2[i][j].g) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].g - RGB2[i][j].g));

        if((RGB[i][j].b - RGB2[i][j].b) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].b - RGB2[i][j].b) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].b - RGB2[i][j].b));
    }
    fprintf(novaImagem, "\n");
}
```

   Resultado:

5. **Multiplicação:**
   A multiplicação é usada para multiplicar cada pixel de uma imagem pelos pixels de outra, para gerar uma só imagem resultante.

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        if((RGB[i][j].r * RGB2[i][j].r) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].r * RGB2[i][j].r) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].r * RGB2[i][j].r));

        if((RGB[i][j].g * RGB2[i][j].g) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].g * RGB2[i][j].g) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].g * RGB2[i][j].g));

        if((RGB[i][j].b * RGB2[i][j].b) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].b * RGB2[i][j].b) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].b * RGB2[i][j].b));
    }
    fprintf(novaImagem, "\n");
}
```

Resultado:

6. **Divisão:**
   A divisão é usada para dividir cada pixel de uma imagem pelos pixels de outra, para gerar uma só imagem resultante.

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        if((RGB[i][j].r / RGB2[i][j].r) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].r / RGB2[i][j].r) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].r / RGB2[i][j].r));

        if((RGB[i][j].g / RGB2[i][j].g) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].g / RGB2[i][j].g) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].g / RGB2[i][j].g));

        if((RGB[i][j].b / RGB2[i][j].b) > 255){
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
        }else if((RGB[i][j].b / RGB2[i][j].b) < 0){
            aux = 0;
            fprintf(novaImagem, "%d ", aux);
        }else
            fprintf(novaImagem, "%d ", (RGB[i][j].b / RGB2[i][j].b));
    }
    fprintf(novaImagem, "\n");
}
```

7. **E Lógico:**

   No E Lógico é feito uma comparação pixel a pixel das duas imagens, quando os pixels de ambas as imagens e mesma posição tem tonalidades iguais e tonalidade é gravada na nova imagem, já quando as tonalidades são diferentes a tonalidade na imagem nova é gravada como branco.

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        if((RGB[i][j].r == RGB2[i][j].r) && (RGB[i][j].g == RGB2[i][j].g) && (RGB[i][j].b == RGB2[i][j].b)){
            fprintf(novaImagem, "%d ", RGB[i][j].r);
            fprintf(novaImagem, "%d ", RGB[i][j].g);
            fprintf(novaImagem, "%d ", RGB[i][j].b);
        }else{
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
            fprintf(novaImagem, "%d ", aux);
            fprintf(novaImagem, "%d ", aux);
        }
    }
    fprintf(novaImagem, "\n");
}
```

Resultado:

8. **OU Lógico:**

```
for(i = 0; i < alt; i++){
    for(j = 0; j < larg; j++){
        if((RGB[i][j].r == RGB2[i][j].r) || (RGB[i][j].g == RGB2[i][j].g) || (RGB[i][j].b == RGB2[i][j].b)){
            fprintf(novaImagem, "%d ", RGB[i][j].r);
            fprintf(novaImagem, "%d ", RGB[i][j].g);
            fprintf(novaImagem, "%d ", RGB[i][j].b);
        }else{
            aux = 255;
            fprintf(novaImagem, "%d ", aux);
            fprintf(novaImagem, "%d ", aux);
            fprintf(novaImagem, "%d ", aux);
        }
    }
    fprintf(novaImagem, "\n");
}
```

Resultado:

9. **NÃO Lógico:**

No NÃO Lógico é feito uma inversão pixel a pixel de uma imagem. Aqui eu usei uma imagem binária onde a imagem resultante obteve branco onde antes era preto e preto onde antes era branco.

```
for(i = 0; i < alt; i++)
    for(j = 0; j < larg; j++){
        if(RGB[i][j].r == 255){
            RGB[i][j].r = 0;
            RGB[i][j].g = 0;
            RGB[i][j].b = 0;
            fprintf(novaImagem, "\n%d ", RGB[i][j].r);
            fprintf(novaImagem, "%d ", RGB[i][j].g);
            fprintf(novaImagem, "%d ", RGB[i][j].b);
        }else if(RGB[i][j].r == 0){
            RGB[i][j].r = 255;
            RGB[i][j].g = 255;
            RGB[i][j].b = 255;
            fprintf(novaImagem, "\n%d ", RGB[i][j].r);
            fprintf(novaImagem, "%d ", RGB[i][j].g);
            fprintf(novaImagem, "%d ", RGB[i][j].b);
        }
    }
```

Resultado: