Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0221 – Introdução à Ciência da Computação I

Trabalho: Filtro de Foto

1. Descrição

Uma famosa rede social pediu para que você crie 2 filtros de foto para eles. O primeiro é um filtro popularmente conhecido como “Foto negativa” que inverte as cores da foto e o segundo é o filtro que deixa a foto preta e branca.
Para isso você receberá uma foto no formato bmp (bitmap 256 cores) e irá criar um outro arquivo que é a foto com o filtro escolhido aplicado.
A entrada será dada pelo nome do arquivo da imagem e pelo filtro desejado, sendo que 1 é para foto negativa e 2 para foto preta e branca.

Exemplo de entrada do programa:
mario.bmp
2

2. Instruções

2.1:Entendendo o formato bmp:

Estrutura geral do formato bmp:
O arquivo bmp estará dividido em 4 partes:

a) Cabeçalho de arquivo
Contém a assinatura BM e informações sobre o tamanho e layout do arquivo BMP

b) Cabeçalho de mapa de bits
Contém as informações da imagem contida no arquivo. Define as dimensões, tipo de compressão (se houver) e informações sobre as cores da imagem.

c) Paleta de cores
Contém o valor de RGB de cada cor.

d) Área de dados da imagem
Dados que permitem a exibição da imagem propriamente dita

Estrutura detalhada do formato bmp:

a) Cabeçalho de arquivo – Tamanho 14 bytes
Campos na ordem que aparecem no arquivo:

Campo	Nº Bytes	Descrição
Assinatura	2	Assinatura do arquivo: os caracters ASCII "BM" é o que identifica o arquivo de ser bmp
Tamanho Arquivo	4	Tamanho do arquivo em bytes
Campo reservado	4	Campo reservado; Deve ser 0
Deslocamento	4	Especifica o deslocamento, em bytes, para o início da área de dados da imagem, a partir do inicio desse cabeçalho. Como a imagem usa paleta será 14+40+(4*NumeroDeCores), como usaremos BMP 256 cores será sempre 1078

b) Cabeçalho de mapa de bits – Tamanho 40 bytes
Campos na ordem que aparecem no arquivo:

Campo	Nº Bytes	Descrição
Tamanho cabecalho	4	Tamanho deste cabeçalho (40 bytes)
Largura da imagem	4	Largura da imagem em pixels
Altura da imagem	4	Altura da imagem em pixels
Número de Planos	2	Número de planos de imagem. Deve ser 1
BitPerPixel	2	Quantidade de bits por pixel. No nosso caso será 8. Pois o número máximo de cores é 2^ Bits por pixel.
Compressão	4	Compressão usada. Deve ser 0 pois não usaremos nenhuma compressão
Tamanho da imagem	4	Tamanho dos dados da imagem, no nosso caso será (tamanho arquivo - 1078)
PixelMetroHorizontal	4	Resolução em pixel por metro na horizontal
PixelMetroVertical	4	Resolução em pixel por metro na vertical
NumeroCoresUsadas	4	Número de cores usadas na imagem, se for 0 indica que usou o número máximo de cores possível que é 2^Bits por pixel.
NumeroCoresImportantes	4	Número de cores efetivamente usadas na imagem, quando zero indica que todas cores são importantes

c) Paleta de cores

Em seguida no arquivo terá a paleta de cores, como usaremos o formato 256 cores, a paleta terá 256 cores, cada cor tem os seguintes dados na ordem:

Campo	Nº Bytes	Descrição
Blue	1	Intensidade de azul da imagem. De 0 a 255
Green	1	Intensidade de verde da imagem. De 0 a 255
Red	1	Intensidade de vermelho da imagem. De 0 a 255
Reservado	1	Campo reservado deve ser sempre 0

Por exemplo nos arquivos o armazenamento da paleta seria:
Paleta[0] – RGB da cor número 0
Paleta[1] – RGB da cor número 1
.
.
.
Paleta[255] – RGB da cor número 255

A ordem das cores na paleta está em ordem de importância.

d) Área de dados da imagem

Nessa área teremos os dados da imagem que basicamente consiste em uma matriz de altura linhas por largura colunas, onde cada byte é um número que representa qual o número da cor na paleta daquela posição da imagem. Ou seja se for 0, aquela posição da imagem tem a cor RGB da paleta[0].
A matriz estará no arquivo da última para primeira linha, ou seja para uma imagem de 1920 (Largura) X 1080 (Altura) os primeiros bytes representam o número da cor na paleta da matriz[1079][0],matriz[1079][1] … até o final do arquivo que representará a matriz [0][1919].
Lembrando que cada valor da matriz representa 1 byte no arquivo;

Entendendo o problema do padding

Os arquivos bmp tem uma limitação em relação a largura da imagem, se a largura da imagem não for múltipla de 4, haverá um pouco de lixo em cada linha para completar a largura até ser multiplica de 4.
Como assim?
Basicamente, por exemplo, se temos uma imagem com 7 de largura, pode-se observar que a largura não é múltipla de 4 e falta 1 para ela ser múltipla de 4, logo a cada linha que será lida no arquivo haverá um byte que é lixo, que está lá apenas para a largura ser múltipla de 4.
Exemplo:
Esse é um exemplo com valores hipotéticos, mas o importante é observar que como a largura não é divisível por 4, cada linha tem lixo ($) para completá-la.
Imagem 2(largura) x 3 (altura):
Área de dados:
23 4 $ $
0 1 $ $
6 12 $ $

2.2:Foto Negativa:

O efeito conhecido como “Foto Negativa” pega as cores RGB de uma imagem e as inverte.
Como o RGB varia de 0 a 255, para achar o inverso de uma cor,basta pegar os valores de Red, Green e Blue de cada cor e tirar 255 deles, para assim ter o valor inverso dos mesmos.
Exemplos:

Cor R:255 G:23 B:0		Inverso R:0 G:232 B:255
Cor R:12 G:185 B:1		Inverso R:243 G:70 B:254

2.3:Foto Preto e Branco:

Para deixar uma foto preto e branco, devemos entender que em RGB escalas de cinza são, resumidamente ,cores onde o R, G e B tem o mesmo valor. Considerando que a

“escala de cinza” mais clara é o branco onde R:255 G:255 B:255 e a mais escura é o preto onde R:0 G:0 B:0.
Logo para transformar uma cor em sua escala de cinza respectiva basta fazer a média entre o R,G e B e colocar o valor médio no R,G e B.
Exemplos:
Cor R:255 G:23 B:0 | Escala Cinza R:92 G:92 B:92
Cor R:12 G:185 B:1 | Escala Cinza R:66 G:66 B:66

2.4:Programa a ser desenvolvido:

Agora que você já sabe como um arquivo bmp é organizado e como achar o inverso e a escala de cinza de um RGB, você deve desenvolver um programa que receba como entrada o nome do arquivo a ser aplicado o efeito e um inteiro que simboliza qual efeito será aplicado, se 1 = foto negativa, se 2 = foto preto e branca.

Criar um arquivo novo:

O seu programa deve, com essas entradas, criar um arquivo novo no formato bmp onde a paleta de cores desse novo arquivo faz esse arquivo ficar com o filtro em questão aplicado.
Exemplos:
Se temos a paleta:
Paleta[0] = R:255 G:23 B:0
.
.
.
Paleta[255] = R:12 G:185 B:1
Se o filtro em questão for o de foto negativa a paleta de cores do novo arquivo deve ser:
Paleta[0] = R:0 G:232 B:255
.
.
.
Paleta[255] = R:243 G:70 B:254
Se o filtro em questão for o de foto preto e branco a paleta de cores do novo arquivo deve ser:
Paleta[0] = R:92 G:92 B:92
.
.
.
Paleta[255] = R:66 G:66 B:66

Logo, você deve passar os cabeçalhos para o arquivo novo, passar a nova paleta de cores para o arquivo novo e por fim passar a matriz para o arquivo novo, a única parte do arquivo de entrada que deve ser diferente do arquivo novo é a paleta de cores.

Nome do arquivo novo:

O nome do arquivo novo deve seguir o seguinte padrão:
Se o filtro aplicado for o de foto negativa, o nome deve ser:
nomeOriginalNegativo.bmp
Se o filtro aplicado for o de foto preto e branca, o nome deve ser:
nomeOriginalPretoBranco.bmp
Exemplos:
Entrada:
mario.bmp
1
Nome do arquivo: marioNegativo.bmp

Entrada:
mario.bmp
2
Nome do arquivo: marioPretoBranco.bmp

Erros:

Seu programa deve detectar 3 erros:
1- Não existir o arquivo
2- O nome do arquivo de entrada não ser .bmp
3- O campo assinatura do cabeçalho de arquivo ser diferente de “BM”

Se ocorrer o erro 1 deve-se exibir a mensagem “Erro no arquivo\n” e finalizar a execução do programa.
Se ocorrer o erro 2 ou 3 deve-se exibir a mensagem
“Arquivo nao eh do formato BMP\n” e finalizar a execução do programa.

Saída do programa:

Seu programa deve imprimir na tela os seguintes dados:
Dados sobre os cabeçalhos no seguinte esquema:
“CABECALHO:\n”
“Iniciais: *Assinatura*\n”
“Tamanho do arquivo: *Tamanho Arquivo*\n”
“Reservado: *Campo reservado*\n”
“Deslocamento, em bytes, para o inicio da area de dados: *Deslocamento*\n”
“Tamanho em bytes do segundo cabecalho: *Tamanho cabecalho*\n”
“Resolucao: *Largura da imagem* x *Altura da imagem*\n”
“Numero de planos: *Número de Planos*\n”
“Bits por pixel: *BitPerPixel*\n”
“Compressao usada: *Compressão*\n”
“Tamanho imagem: *Tamanho da imagem*\n”
“Resolucao horizontal: *PixelMetroHorizontal* pixel por metro\n”
“Resolucao Vertical: *PixelMetroVertical* pixel por metro\n”

“Numero de cores usadas: *NumeroCoresUsadas*\n”
“Numero de cores importantes: *NumeroCoresImportantes*\n”
Em seguida dados sobre a paleta do arquivo lido, no seguinte esquema:
“PALETA ORIGINAL:\n”
“Paleta[0]: R:%d G:%d B:%d\n”
“Paleta[1]: R:%d G:%d B:%d\n”
.
.
.
“Paleta[255]: R:%d G:%d B:%d\n”
Em seguida dados sobre a nova paleta no seguinte esquema:
“PALETA NOVA\n:”
“Paleta[0]: R:%d G:%d B:%d\n”
“Paleta[1]: R:%d G:%d B:%d\n”
.
.
.
“Paleta[255]: R:%d G:%d B:%d\n”
Em seguida deve-se imprimir a soma de cada linha da matriz no seguinte esquema:
A soma de cada linha da matriz deve ser a soma de cada elemento da linha e para cada lixo na linha deve-se subtrair 1 a soma.
Exemplo:
Linha 0: 1 2 3 $ → Soma[0] = 5 (1+2+3-1)

A impressão deve seguir esse formato:
“Soma linha 0: %lld\n”
“Soma linha 1: %lld\n”
.
.
.
“Soma linha *Altura da imagem-1*: %lld\n”

Por fim deve-se imprimir o nome do arquivo gerado, da seguinte maneira:
“*nomeDoArquivo*\n”

Atenção
Lembre-se que no arquivo a matriz começa da última para a primeira linha, mas na impressão das somas das linhas deve-se imprimir a soma da primeira linha para última.

Avaliação do trabalho:
A correção levará em consideração 2 avaliações principais:
1- Automática (Run Codes)
As saídas baterem com o Run Codes como os outros trabalhos

2- Manual
Ver manualmente se o arquivo gerado pelo programa é realmente a imagem com o filtro corretamente aplicado.
Além de que espera-se que o aluno utilize todos os conceitos aprendidos nas aulas, como structs, alocação dinâmica,funções e afins.
Mas como ter certeza que o filtro está corretamente aplicado?
Seguindo a especificação do trabalho é para o filtro ser aplicado corretamente, para conferir basta abrir o arquivo gerado pelo seu programa e ver a imagem gerada, se ele não abrir ou não parecer correto, está errado.
Meu arquivo abre mas não como vou saber que a minha imagem gerada é a imagem com o filtro corretamente aplicado?
Todos os casos testes, arquivos de entrada e o arquivo esperado de saída serão disponibilizados para vocês compararem a imagem de saída com a imagem esperada de saída.

3. Instruções Complementares
Todos os dados dos arquivos estarão em binário, devem ser lidos e escritos em binário, para isso no fopen utilize os modos de abertura “rb” e “wb” que leem ou escrevem em arquivos em binário.
Quando um dos erros ocorrerem é esperado que nenhum arquivo seja criado.
Alguns arquivos de entrada já estão com o efeito de foto negativa, nesse caso após aplicar o efeito de foto negativa a foto deve voltar ao normal, pois o inverso do inverso de uma cor é a própria cor.
Os valores de RGB são números inteiros.