<u>5ª exercício avaliativo (EA4)</u>
Programação II (INF16153) - UFES
04 de Julho de 2023

O exercício consiste em ler dados de uma imagem armazenados em um **arquivo binário** e mostrar na tela um histograma dos pixels. Como ilustrado na Figura 1, imagens em tons de cinza (não coloridas) podem ser vistas como matrizes de números em que valores grandes representam cores claras e valores próximos de zero representam cores escuras. Como ilustrado na Figura 2, o histograma de uma imagem conta quantos pixels possuem valores dentre de determinados intervalos. O histograma do exemplo informa que a imagem possui 1052 pixels com valores no intervalo [0, 52), 1760 pixels com valores no intervalo [52, 104), 820 pixels com valores no intervalo [104, 156), 650 no intervalo [156, 208) e 433 no intervalo [208, 256).

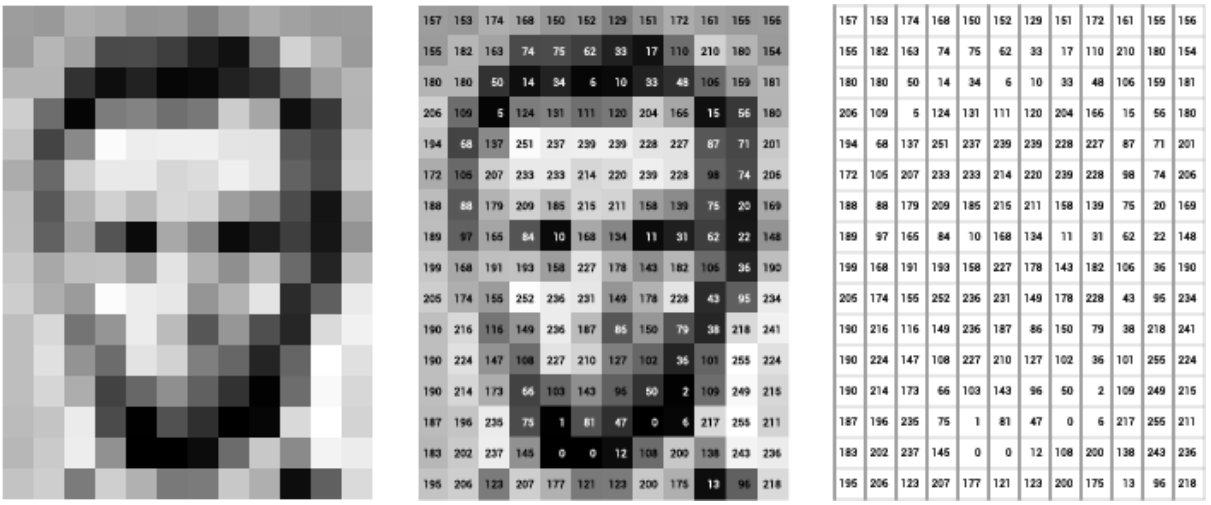

Figura 1: Imagens em tons de cinza como matrizes de números.

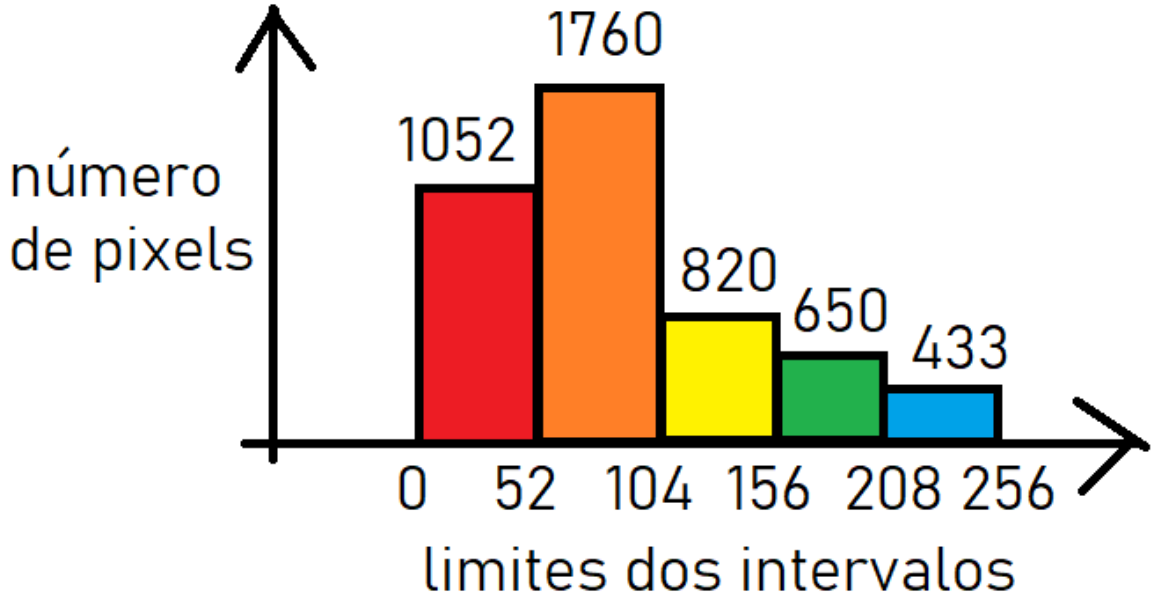

Figura 2: Exemplo de histograma.

Entradas, Formato dos Dados e Regras

O nome do arquivo contendo a imagem deverá ser digitado pelo usuário do programa. Alguns arquivos de exemplo foram disponibilizados junto com esta especificação. Em cada arquivo, os dados da imagem estão organizados da seguinte forma:

- Dois inteiros de 4 bytes contendo o número de linhas M e de colunas N da imagem
- Um inteiro de 4 bytes representando o tipo que pode ser 0 ou 1, sendo que 0 representa FLOAT e 1 representa INT.
- M x N valores dos tipos INT ou FLOAT, dependendo do campo anterior, cada um com 4 bytes, representando as cores dos pixels. Os dados estão em formato linearizado, i.e., as linhas da matriz foram colocadas uma na frente da outra de forma que a estrutura resultante tenha a forma de um vetor com 1 linha e M * N colunas. Se a imagem for do tipo INT, os valores dos pixels estarão entre 0 e 255, sendo que 0 representa preto e 255 representa branco. Se a imagem for do tipo FLOAT, os valores estarão entre 0.0 e 1.0, sendo que 0.0 representa preto e 1.0 representa branco. Em ambos os casos, valores intermediários entre os limites mínimo e máximo são tons de cinza.

Deverá ser escrita uma estrutura Imagem contendo como atributos o número de linhas e colunas da imagem, o tipo e um void* para armazenar os valores dos pixels. Deverão ser escritas ainda funções para ler a imagem e para destruir a imagem (veja o formato da `main` no arquivo em anexo).

Deverá ser criada uma estrutura Histograma contendo um int* para armazenar os números de pixels em cada intervalo e dois inteiros indicando o número de intervalos e o tamanho de cada intervalo. O número de intervalos deverá ser digitado pelo usuário. Deverão ser escritas funções para calcular um histograma de uma imagem, mostrar um histograma na tela e destruir um histograma. No cálculo do histograma, se a imagem for do tipo float, os valores dos pixels devem ser convertidos de [0.0, 1.0] para [0, 255]. Para isso, multiplique o valor float por 255 e converta o resultado para o tipo int.

Atenção: O tamanho dos intervalos deverá ser calculado a partir do número de intervalos digitado pelo usuário e sabendo que existem 256 valores possíveis de pixels (0 a 255). Como o tamanho dos intervalos é inteiro, se você concluir que cada intervalo deverá armazenar 2.35 valores diferentes, tenha em mente que na verdade eles deverão armazenar 3 valores diferentes cada.

Dica: Para testar se o cálculo do histograma está correto, verifique se a soma das contagens é igual ao número de colunas vezes o número de linhas da imagem.

Atenção: Não é válido converter os valores da imagem para int na leitura e armazenar na estrutura da imagem um int* ao invés de um void*.

Exemplos de Entrada e Saída

O programa deverá mostrar na tela os limites e as quantidades de pixels em cada intervalo.

Entrada	Saída
img-1-float.bin 5	[0, 52): 12679 [52, 104): 28799 [104, 156): 43482 [156, 208): 30335 [208, 260): 25705

img-1-int.bin 4	[0, 64): 17607 [64, 128): 42821 [128, 192): 47969 [192, 256): 32603
img-1-int.bin 10	[0, 26): 1686 [26, 52): 10993 [52, 78): 11995 [78, 104): 16804 [104, 130): 20732 [130, 156): 22750 [156, 182): 18404 [182, 208): 11931 [208, 234): 12530 [234, 260): 13175
img-2-float.bin 6	[0, 43): 18613 [43, 86): 28852 [86, 129): 32444 [129, 172): 28761 [172, 215): 24519 [215, 258): 30971
img-2-float.bin 2	[0, 128): 79186 [128, 256): 84974

Regras gerais (as de sempre)

- A atividade é **individual**. Todas as questões serão testadas e plágio não será tolerado
- Seu programa deve implementar TADs com **encapsulamento total (opaco)**
- Toda lógica de negócio deve ser implementada fora da sua `main()`. Em outras palavras, você só deve fazer chamadas de funções do seu TAD (como realizado em exercícios na sala de aula)
- Seu programa será testado com o Valgrind para detectar vazamento de memória e erros de alocação. É sua responsabilidade liberar toda memória alocada.
- **Números de ponto flutuante devem ter precisão simples e apenas duas casas decimais devem ser impressas**
- Você deve fornecer um **Makefile** que gere um arquivo executável chamado **EA5**
- **O seu programa será executado da seguinte forma:**
  - ./EA5 < entrada > saida
- Haverá **correção automática**, portanto, siga os padrões de saída corretamente
  - O corretor ignora espaços e quebras de linha
  - Porém, se você escrever informações na tela, ele retornará um erro na saída (por exemplo: “digite um time”). Portanto, escreva somente o que foi solicitado na tela
- Organização, modularização e boas práticas de programação são critérios fundamentais de avaliação.
- A submissão da atividade será realizada via Github de acordo com as instruções já conhecidas
  - Criar uma pasta EA4 e submeter a atividade dentro dela

Anexo: Formato da main.c

```c
#include <stdio.h>
#include "image.h"
#include "histogram.h"

int main()
{
    char path[1024];  // nome do arquivo que contem a imagem
    int n_buckets;  // numero de intervalos

    Image *image;
    Histogram *histogram;

    scanf("%s", path);
    scanf("%d", &n_buckets);

    image = image_read(path);
    histogram = histogram_compute(image, n_buckets);
    histogram_show(histogram);
    histogram_destroy(histogram);

    return 0;
}
```