ICC2 – Trabalho 4

Entrega: 02/09/2013 à meia-noite
Sistema de Submissão de Programas: https://ssp.icmc.usp.br
Data máxima para encaminhamento de dúvidas sobre o trabalho: 23/08/2013

Faça um programa para ler um arquivo de imagem no formato PGM. Esse formato é composto por um cabeçalho que tem duas linhas iniciais (que você pode desprezar) e, em seguida, há uma linha de texto com o número de pixels de largura e de altura da imagem. A linha seguinte contém o valor máximo de escala de cinza para a imagem. Posteriormente são listados os bytes que compõem a imagem. Cada byte representa um pixel.

Exemplo:

```
P2
# CREATOR: GIMP PNM Filter Version 1.1
800 600
255
1
92
97
94
...
93
```

A primeira linha tem o texto P2, o qual deve ser descartado. Em seguida, descarte a segunda linha, a qual apresenta detalhes do software que gerou a imagem. A terceira linha contém o número de pixels que define a largura da imagem (neste caso 800 pixels de largura) e, em seguida, a altura da imagem (neste caso 600 pixels). Na linha seguinte há o valor máximo de escala de cinza para a imagem, o qual, neste caso, é 255.

O valor máximo 255 indica que a imagem contém pixels com valor 0 até 255. Em que o 0 indica a cor preta e 255 representa a cor branca. Demais valores intermediários representam diferentes níveis de cinza.

Você deve ler o cabeçalho desse arquivo PGM e os bytes que compõem a imagem. Em seguida deve atender às tarefas abaixo listadas.

Tarefas:

1) Imprima a largura da imagem em pixels;

2) Imprima a altura da imagem em pixels;

3) Imprima o valor máximo de escala de cinza da imagem;

4) Imprima o número total de pixels da imagem (largura x altura da imagem);

5) Imprima o número de pixels cujo valor é igual a um inteiro A definido na entrada;

6) Imprima o número de ocorrências para cada bin do histograma segundo um inteiro B definido na entrada;

- O que é um histograma?

É uma representação na qual um conjunto de dados é agrupado em intervalos. Por exemplo, considere o seguinte conjunto de dados **Pixels = {25, 0, 255, 254, 28, 2, 125, 124, 126}**. Suponha que damos como entrada o número de bins igual a 3. Isso quer dizer que precisamos dividir, em tamanhos iguais, o intervalo [0, 255], ou seja:

- De 0 a 255 temos 256 valores
- Assim dividimos 256 / 3 = 85,3333...
- Utilize a parte inteira da divisão nesse caso 85 para montar cada intervalo do histograma (*bin*)
- Assim os intervalos do histograma ou bins do histograma serão [0, 84], [85, 169], [170, 255]
- Contamos quantos valores do conjunto Pixels estão em cada um dos intervalos:
    - [0, 84] → temos {25, 0, 28, 2} → o que totaliza 4 elementos
    - [85, 169] → temos {125, 124, 126} → o que totaliza 3 elementos
    - [170, 255] → temos {255, 254} → o que totaliza 2 elementos
- Assim imprimimos:
    [0, 84] 4
    [85, 169] 3
    [170, 255] 2

Repare que, nesse caso o último bin terá um intervalo maior, de 86 valores por causa do resto da divisão. Você deverá implementar seu programa para que esse resto seja sempre inserido no último intervalo do histograma.

Encontrar o histograma de uma imagem é útil para muitas aplicações de processamento de imagens, incluindo: melhorar a qualidade de imagens, classificar imagens segundo os objetos da cena, compreender a distribuição de tons da imagem, etc.

Leituras adicionais:

http://pt.wikipedia.org/wiki/Histograma
http://www.cambridgeincolour.com/pt-br/tutorials/histograms1.htm

Entrada:

<nome do arquivo da imagem>
<inteiro A: número de pixels com valor igual ao inteiro A>
<inteiro B: número de bins do histograma>

Saída

```
<largura>\n
<altura>\n
<max_cinza>\n
<tamanho>\n
<ocorrencias_valor_A>\n
<bin1> <bin2> <bin3> ...<binB> \n
```

Obs: há um espaço depois da impressão do valor de cada *bin*

Exemplo

Imagem de entrada (`img0.pgm`):

```
P2
# CREATOR: GIMP PNM Filter Version 1.1
3 3
255
0
47
99
99
138
178
201
229
255
```

Entrada do programa:

```
img0.pgm
99
3
```

Saída do programa

```
3
3
255
9
2
2 3 4
```

Informações importantes (LEIA COM ATENÇÃO)

- Sobre a avaliação

    1. Um dos objetivos da disciplina de ICC2 é o **aprendizado individual** dos conceitos de programação.

A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos, nem codificar em conjunto. Portanto, compartilhem idéias, soluções, modos de resolver o problema, mas não o código.

- O plágio vai contra o código de ética da USP.
- Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação.
- O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e idéias. Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que para poupar memória basta usar uma certa estrutura de dados, etc. O que você **não** deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado do seu colega:
  - depois de o seu colega ter visto o seu código, será muito mais difícil para ele imaginar uma solução original e própria;
  - o seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.
- Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela.
- O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2. Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e **os trabalhos com similaridade detectada terão suas notas zeradas**, tanto aqueles relativos ao código de origem quanto do código copiado.
   - O plágio pode ser detectado **mesmo em pequenos trechos de código,** quando esses trechos tiverem papel importante no desenvolvimento do trabalho.