ICC2 – Trabalho 5

Entrega: 09/09/2013 as 23h59
Sistema de Submissão de Programas: https://ssp.icmc.usp.br
Data máxima para encaminhamento de dúvidas sobre o trabalho: 04/09/2013

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
<valor do pixel em um unsigned char>...<valor do pixel em um unsigned char>
```

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.

Após essa parte o arquivo ***deve ser lido byte a byte, no formato binário***. Isso porque o arquivo tem o cabeçalho ASCII e o conteúdo binário.

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) Realize um processamento pixel a pixel para gerar uma nova imagem com base num valor inteiro que será lido como entrada. Pixels com ***valor maior ou igual*** a esse valor inteiro deverão assumir o valor do pixel máximo (no exemplo acima, 255) na nova imagem. Pixels com ***valor menor deverão assumir valor 0*** na nova imagem. Essa operação é conhecida como ***filtro de Threshold***.

Ou seja, sendo f(x) a imagem de entrada, e g(x) uma imagem de saída, com x representando a coordenada de cada pixel da imagem, e T o valor inteiro, que é chamado limiar (threshold):

$$g(x) = \begin{cases} max, & \text{se } f(x) >= T \\ 0, & \text{caso contrario} \end{cases}$$

2) Produza um arquivo de saída cujo nome será passado por parâmetro para o programa, que contenha a nova imagem, após aplicação do ***filtro de Threshold***. Esse arquivo deve ter como cabeçalho:

```
P5
# Created by Trabalho 05
<largura da imagem> <altura da imagem>
<valor máximo da escala de cinza>
<bytes que correspondem aos pixels da imagem>
```

Após o cabeçalho, os valores da imagem resultante devem ser gravados byte a byte.

3) Utilizando a imagem após a aplicação do filtro de Threshold, **assuma que:**

- **a imagem será sempre quadrada**
- **que o tamanho da imagem será sempre uma potência de 2,**
- **toda imagem terá ao menos 4x4 pixels**

Assim, a imagem processada pelo filtro de Threshold pode ser dividida em 16 subimagens, conforme esquema abaixo:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Então, o programa deverá:

- contar quantos pixels de valor 0 (zero) ocorrem em cada subimagem (1 a 16) da imagem após realização do filtro de Threshold
  - *Mas o que ocorre se a imagem contiver 2^3 linhas e 2^3 colunas? Nesse caso você terá 8 linhas e 8 colunas que deverão ser particionadas em 16 subimagens conforme o exemplo acima.*
- imprimir o identificador numérico (ID) de cada subimagem e sua respectiva quantidade de zeros (CONTAGEM), de forma crescente pela CONTAGEM
- **caso haja empate, desempatar pelo identificador da subimagem, i.e., priorizando a subimagem de menor ID**
  - **Note que ID começa em 1 e vai até 16**

Entrada padrão:

```
<nome do arquivo da imagem>
<inteiro>
<nome do arquivo de saida>
```

Saída padrão:

```
<id_subimagem_com_menor_quantidade_de_zeros>\t<quantidade_de_zeros>\n
…
<id_subimagem_com_maior_quantidade_de_zeros>\t<quantidade_de_zeros>\n
```

Exemplo de saída (fictício, apenas para ilustração):

```
16    0
7     15
8     23
15    30
1     110
4     200
10    201
13    202
14    202
9     205
2     250
3     1000
12    1020
11    1200
5     1500
6     2000
```

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.