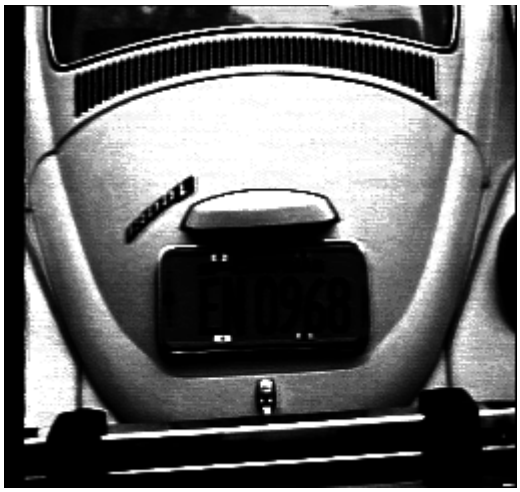
Atividade para as férias– Lab ICC 1 (SCC-222)
(conta ponto pra ir pro céu!)

CSI – A missão

Finalmente, após inúmeras tentativas frustradas, Dick Vigarista pôde finalmente comemorar a sua primeira vitória à frente de Peter Perfeito, Irmãos Rocha, Professor Aéreo, Rufus Lenhador, Penélope Charmosa, Quadrilha da Morte, dentre outros intrépidos corredores.

Aconteceu, porém, um corredor participante, de nome Max Vestápi, resolveu denunciar o nosso astro-vilão da "Corrida Maluca" por excesso de velocidade! Sim. A corrida vencida por Dick Vigarista tinha por regra não ultrapassar a velocidade máxima de 180 km/h. Max Vestápi, com livre trânsito na FIA, conseguiu uma imagem de radar que, segundo ele, comprovaria o delito !!! Veja com seus próprios olhos...



Acontece que a câmera é de quinta categoria e a placa do carro não pode ser vista a olho nu. Entretanto, a imagem revela claramente uma traseira de fusca (mesmo modelo e cor do carro de Dick Vigarista !) e o radar que disparou a câmera, indica a incrível velocidade de 295 km/h. (Sim… Dick tinha sob o capô de seu fusquinha 66, um motor V8 turbinado, movido a fusão nuclear!)

Em meio a toda celeuma, a Polícia Federal (PF) contratou você, para desvendar este mistério: se a placa do automóvel for revelada, Dick pode, mais uma vez, ver seu sonho de vencer uma corrida ir por água abaixo. E Penelope Charmosa, a segunda colocada, levantará o troféu de vencedora!

Só que a PF quer que você faça um programa, que possa ser aplicado em casos futuros de fraude. A entrada consiste do nome do arquivo (imagem) que você terá que modificar para revelar, finalmente, informações ocultas.

A Operação: é o tipo da função que você deve aplicar sobre a imagem: operação de log. (ver detalhes mais adiante)

NomeArquivo é nome da imagem no formato <*.pgm> onde '*' é qualquer cadeia de caracteres (sem espaços).

O formato de imagem PGM (http://netpbm.sourceforge.net/doc/pgm.html) é um dos mais simples que existem e armazena imagens em nível de cinza apenas, com pixels de profundidade 1, isto é, cada pixel tem exatamente um BYTE podendo ter valores entre 0 – 255 (em C, qual o tipo mais "econômico" possível para representá-los??). O cabeçalho da imagem (mais conhecido como ***header***) tem o seguinte formato (sempre 4 linhas de texto):

P<**n**>
Comentario qualquer: normalmente o software que criou a imagem !
XDIM YDIM
MaxVal

Onde **n** pode ser == 2: imagem é armazenada textualmente, ou seja, no formato ASCII.
== 5: imagem é armazenada no formato binário.

 # qualquer coisa……. É uma linha de comentário, indicando o "fabricante" da imagem !

XDIM é um inteiro que representa a largura, em pixels, da imagen
YDIM é um inteiro que representa a altura da imagem.

XDIM e YDIM podem ser separados ou por um espaço em branco ou então um valor por linha.

MaxVal é o valor do maior pixel encontrado na imagem (normalmente 255).

Após o cabeçalho de 4 linhas, inalmente temos os pixels da imagem. Uma coisa muito interessante é que os pixels da imagem podem ser armazenados tanto no formato ASCII, quanto binário (mesmo que o header sempre seja ASCII)!

Caso a imagem seja do tipo P2 (ASCII), na linha abaixo de **MaxVal** teremos os pixels da imagem (valores inteiros entre 0 e 255) UM EM CADA LINHA.

Caso a imagem seja P5 (binária), na linha abaixo de **MaxVal** teremos um STREAM de BYTES (binário) de tamanho XDIM * YDIM, que contém todos os pixels da imagem.

Veja exemplos de cada uma destes formatos em:

Fusca no formato ASCII (http://www.lcad.icmc.usp.br/~jbatista/scc221/fusca.pgm)
Fusca no formato Binário (http://www.lcad.icmc.usp.br/~jbatista/scc221/fuscaBIN.pgm)

Faça o download de ambas as imagens e visualize no viewer de sua preferência. São exatamente iguais, apenas armazenadas de formas diferentes! Agora abra-as em um editor de texto. Tente entender por que a imagem binária aparece como "lixo" no editor! Qual a explicação ????

A Saída

Seu programa deve gerar como saída uma imagem processada no formato PGM, SEMPRE NO FORMATO ASCII (mesmo que a imagem original seja no formato binário).

O comentário na segunda linha do cabeçalho da imagem que você deverá gerar deverá ter exatamente o seguinte conteúdo:

"# CREATOR: Image Generator SCC-222 – Lab ICC I" (sem as aspas)

Sobre os requisitos do trabalho:

a) Utilize uma **struct** para organizar o seu tipo "imagem": esta **struct** deve conter: as dimensões X e Y da imagem; os valores do pixel de maior valor e o de menor valor da imagem (estes serão úteis para realizar as operações de *log* e *stretching*); o tipo da imagem (binário ou ASCII) e claro, os pixels de sua imagem – QUE DEVER SER ALOCADA DINAMICAMENTE! Você pode usar tanto uma estrutura bi-dimensional quanto unidimensional.

b) utilize funções para tudo: operações tais como loadimage, saveImage, calcula min e max, libera imagem, e as operações solicitadas devem ser funções !!! Tente manter o seu código em main() o mais curto possível, sempre chamando funções para realizar as operações necessárias. Mantenha o número de argumentos reduzido, passando a sua struct como um dos argumentos.

c) evite variáveis globais. Não há necessidade de usar NENHUMA variável global.

Sobre a operação de Log:

consulte o link http://homepages.inf.ed.ac.uk/rbf/HIPR2/

e selecione: worksheets →Point Operations. Lá você encontrará links para (Contrast stretching e Logarithm Operation). São operações MUITO simples de entender e implementar.

Log: deve obedecer a seguinte equação **y = c * log(1+y)** com **c = 255 / log(1+max),** onde:
y é todo pixel da nova imagem; x é todo pixel da imagem original e max é o pixel de maior valor da imagem original.

NOTA: sugiro que você, após processar a sua imagem, abra-a no seu visualizador de preferência (pegue a saida do RunCodes, renomei o .out como um arquivo .pgm) e desvende o mistério da placa do carro com um visualizador de sua preferência!!! VC acabou de fazer o seu primeiro código de processamento de imagens !!!!
NOTA2: perceba que a operação de log NÃO resolve sempre o problema para qualquer imagem. Por que?

Bom divertimento!