UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Disciplina	Turmas
Algoritmos e Estruturas de Dados II	

Professores

Erickson Rangel do Nascimento e Leonardo Barbosa e Oliveira

Data de entrega: 20/04/2017 **até as** 23:55

Trabalho Prático em Linguagem C (TP0) - 10 pontos

O objetivo deste trabalho é implementar e aplicar a operação de convolução em imagens digitais no formato PGM. Uma imagem PGM tem o formato como demonstrado abaixo, onde P2 é somente o indicativo que o formato contém uma imagem em tons de cinza, c indica o número de colunas da imagem (largura da imagem), l indica o número de linhas da imagem (altura da imagem) e v indica o valor máximo que um pixel pode assumir (neste trabalho considera-se que v será no máximo 255, para poder ser armazenado em um *unsigned char*). As entradas p(i,j) indicam o valor do pixel na linha i e coluna j da imagem.

```
P2
c l v
p(0,0) p(0,1) ... p(0,c-1)
p(1,0) p(1,1) ... p(1,c-1)
p(2,0) p(2,1) ... p(2,c-1)
...
p(l-1,0) p(l-1,1) ... p(l-1,c-1)
```

Um exemplo de imagem é dado a seguir.

```
P2
3 2 255
200 128 24
255 211 35
```

Esta imagem contém três pixels de largura e dois pixels de altura, o que equivale a uma matriz com 2 linhas e 3 colunas, e o valor máximo de um pixel é de 255. Neste exemplo, p(1,2) = 35 e p(0,0) = 200.

TAD

O aluno deverá utilizar a estrutura de dados para armazenar a imagem carregada do arquivo.

```
typedef struct {
   int c;                  // número de colunas na imagem
   int l;                  // número de linhas na imagem
   unsigned char maximo;     // valor máximo para cada pixel
   unsigned char **imagem;    // variável para armazenar os pixels da imagem (matriz)
} PGM;
```

Convolução

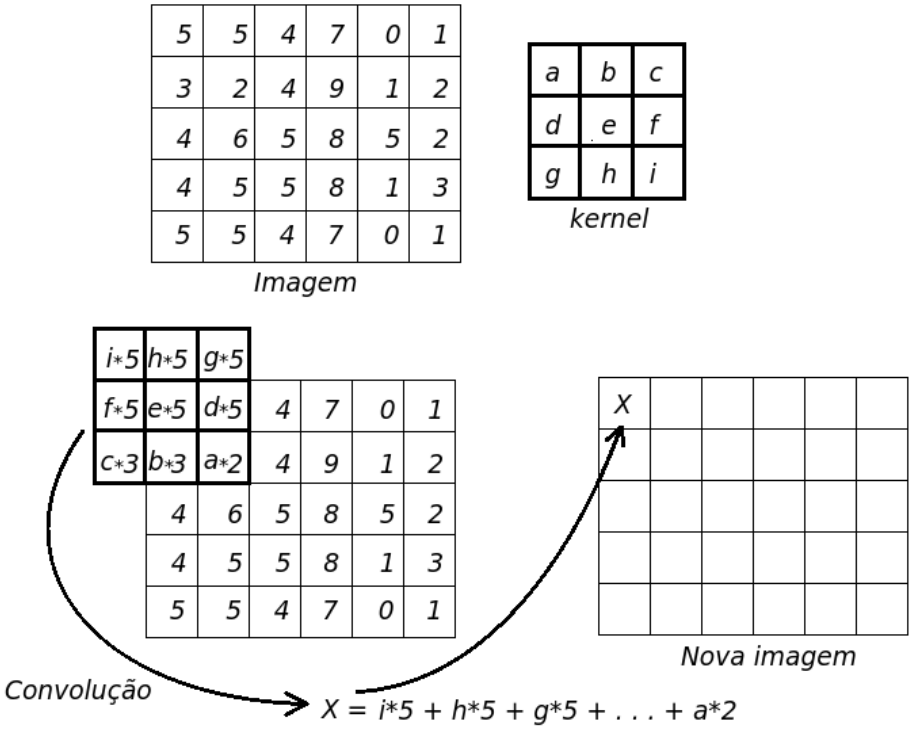
A convolução é uma operação linear entre duas funções f e g que produz uma terceira função h, que pode ser interpretada como uma versão modificada (filtrada) de f. Formalmente, para duas funções $f(x, y)$ e $g(x, y)$ de variáveis contínuas, a convolução é definida como:

$$h(x, y) = f(x, y) * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau_1, \tau_2) \cdot g(x - \tau_1, y - \tau_2) d\tau_1 d\tau_2 \tag{1}$$

Uma imagem digital é uma função bidimensional discreta *f(x,y)* que retorna a intensidade do pixel (entre 0 e 255) dadas as coordenadas x (linha) e y (coluna). Assim, a versão discreta é:

$$h[x, y] = f[x, y] * g[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2] \tag{2}$$

Para realizar a convolução em uma imagem digital precisamos de uma matriz denominada *kernel*. Aplica-se o *kernel* sobre a imagem, multiplicando cada pixel pelo valor correspondente, e a soma desses produtos é atribuída ao pixel central da nova imagem convoluída. A figura abaixo ilustra este procedimento:



Para evitar que a nova imagem seja menor do que a imagem de entrada os pixels das extremidades devem ser replicados ao realizar a convolução.

Derivadas de imagens

Em Processamento Digital de Imagens é muito comum calcular a derivada de uma imagem para encontrar regiões com variações abruptas que geralmente representam fronteiras (bordas) entre objetos e/ou o fundo da imagem. Relembrando a definição de derivada:

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x} \approx f(x + 1, y) - f(x, y) \tag{3}$$

É possível derivar uma imagem utilizando uma convolução. Para isso, basta utilizarmos o *kernel de Prewitt* representado pela matriz a seguir:

-1	0	1
-1	0	1
-1	0	1

O que deve ser feito

1. Elaborar um programa capaz de ler uma imagem no formato PGM e depois salvar a sua imagem derivada utilizando convolução.

2. Os nomes dos arquivos de entrada e saída deverão ser passados por argumentos para o programa.

   ```
   .\exec entrada.pgm saida.pgm
   ```

   onde *exec* é o nome do executável, *entrada.pgm* representa o nome do arquivo de entrada (imagem no formato PGM) e *saida.pgm* é o nome do arquivo de saída (formato PGM).

3. O programa deve ter a função `PGM *LerPGM(char* entrada)` que recebe o nome do arquivo como entrada, realiza a leitura da imagem e retorna uma instância alocada de forma *dinâmica* da estrutura de dados PGM descrita acima. É importante ressaltar que a variável imagem dentro da TAD também deverá ser alocada de forma dinâmica no tamanho exato da imagem.

4. O programa deve ter a função `void SalvarPGM(PGM *img, char* saida)` que escreve a imagem contida em *img* em um arquivo no formato PGM com o nome descrito na variável saida.

5. O programa deverá ter a função `void Convolucao(PGM *img, char **kernel, PGM *saida)` que realiza a convolução de *img* pelo *kernel* e armazena o resutlado em *saida*. Caso o somatório da convolução em algum pixel seja menor que 0 (zero) o valor desse pixel deve ser 0 (zero) na nova imagem. Caso o somatório seja maior que 255 o valor do pixel na nova imagem deve ser 255. O PGM que armazenará a imagem de saída deverá ser alocado dinamicamente.

6. Todas as funções implementadas devem possuir um cabeçalho conforme o exemplo a seguir:

```
/*-----------------------------------------------------------------
Protótipo: void Convolucao(PGM *img, char **kernel, PGM *saida)
Função: Realiza a operação ...
Entrada: estruturas PGM contendo ...
Saída: ...
-----------------------------------------------------------------*/
```

7. Seu programa não deverá ter nenhum *leak* de memória, ou seja, tudo que for alocado de forma dinâmica, deverá ser desalocado com a função "free()"antes do término da execução.

8. O programa deverá ser possível de ser compilado no Linux, portanto ele não deverá conter nenhuma biblioteca que seja específica do sistema operacional Windows.

9. Você deverá implementar o trabalho na IDE Code::Blocks (*disponível em: http://www.codeblocks.org/*). Deve ser submetido ao moodle o diretório do projeto criado no Code::Blocks em um arquivo compactado contendo os arquivos ".h"(com as declarações das funções para ler e salvar imagens PGM) e ".c"(um com as implementações das funções descritas nesse documento e outro com a função *main* (este deverá ter o nome *"main.c"*). Além disso, a submissão também deverá conter o relatório do trabalho em formato PDF.

10. O trabalho deverá ser entregue via *moodle* até as 23:55 horas do dia 20/04/2015. **Trabalhos que forem entregues fora do prazo não serão aceitos em nenhuma hipótese.**

Exemplo para teste

Para avaliar o funcionamento do seu programa disponibilizamos as imagens *teste.pgm*, *carro.pgm* e *lena.pgm*. As derivadas deverão ser:

92	99	1	8	15	67	74	51	58	40
98	80	7	14	16	73	55	57	64	41
4	81	88	20	22	54	56	63	70	47
85	87	19	21	3	60	62	69	71	28
86	93	25	2	9	61	68	75	52	34
17	24	76	83	90	42	49	26	33	65
23	5	82	89	91	48	30	32	39	66
79	6	13	95	97	29	31	38	45	72
10	12	94	96	78	35	37	44	46	53
11	18	100	77	84	36	43	50	27	59

teste.pgm

$$*\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$

4	255	248	0	0	0	48	23	38	59
0	98	218	43	0	0	23	0	43	64
0	73	193	73	0	0	0	0	73	89
0	43	218	98	0	0	0	0	98	84
0	68	98	18	0	0	0	23	43	29
4	0	0	0	23	43	18	23	0	0
84	0	0	0	148	168	23	0	0	0
89	0	0	0	168	168	0	0	0	0
64	0	0	0	168	148	0	0	0	0
0	0	0	48	143	123	0	23	0	0

derivada

carro.pgm

derivada

lena.pgm *derivada*

Documentação

Escreva um documento explicando o seu código e avaliando o desempenho de sua implementação. Separe-o em cinco seções: introdução, implementação, resultados, conclusão e referências. Seja claro e objetivo.

- **Introdução**: Faça uma breve introdução o problema, definindo-o com as *suas* palavras.
- **Implementação**: Explique quais foram as estratégias adotadas para a leitura das imagens em formato PGM, realização da operação de convolução e a escrita da imagem PGM em disco. Descreva qual o papel de cada função, seus parâmetros de entrada e de saída.
- **Resultados**: Faça testes com seu programa utilizando diversas imagens. Apresente o resultado obitido e faça uma breve discussão sobre eles. (Serão disponibilizadas duas imagens para teste)

- **Conclusão**: Explique quais foram as dificuldades encontradas durante o desenvolvimento e as conclusões obtidas.
- **Referências**: Se você utilizou informações adicionais além das especificadas neste trabalho, cite as fontes.

Avaliação

- **5 pontos** pela implementação, onde serão avaliados, além do funcionamento adequado do programa: identação correta do código, comentários das funções, alocações de desalocações dinâmicas bem feitas e modularização.
- **5 pontos** pela documentação, onde serão avaliados a clareza das seções e a discussão dos resultados obtidos.
- ***Lembramos que será utilizado um software de detecção de cópias e qualquer tipo de plágio detectado resultará em nota 0 para ambos trabalhos!***