MAC 110 — Introdução à Computação

BCC — Primeiro Semestre de 2007

Quarto Exercício-Programa — Prazo de entrega: até **29 de junho de 2007.**

Transformações Sobre Imagens e o Filtro da Mediana

Neste exercício você escreverá um programa simples para manipulação de imagens. Seu programa permitirá que o usuário abra um arquivo contendo uma imagem, aplique uma ou mais transformações sobre a imagem e salve a imagem modificada em um arquivo de saída. Ele lidará apenas com imagens cujos pontos podem assumir tonalidades de cinza que variam do branco ao preto. Sobre tais imagens, o programa deverá ser capaz de aplicar as seguintes transformações:

- inversão horizontal (*horizontal flip*);
- inversão vertical (*vertical flip*);
- rotação de 90 graus no sentido horário;
- rotação de 90 graus no sentido anti-horário;
- rotação de 180 graus;
- filtragem de ruído impulsivo (filtro da mediana).

As transformações por rotação fazem exatamente o que seus nomes dizem. A inversão horizontal é a transformação que converte um $\rightharpoonup$ num $\leftharpoondown$. (Ela não deve ser confundida com a rotação de 180 graus, que converte um $\rightharpoonup$ num $\leftharpoonup$.) A inversão vertical é a transformação que converte um $\rightharpoonup$ num $\rightharpoondown$. A filtragem de ruído impulsivo é feita com o chamado filtro da mediana, que será explicado na seção 1.

1 Filtro da mediana

Filtro da mediana é uma transformação bastante comum para suavizar ruídos do tipo impulsivo em sinais e imagens digitais.

Definição de vizinhança. Seja A uma matriz de inteiros positivos com m linhas e n colunas, e sejam p e q dois inteiros positivos ímpares. Dada uma coordenada (i, j) em A, a vizinhança de tamanho $p \times q$ em torno de (i, j) é a submatriz $A_{i,j}$ de A com p linhas e q colunas e centro em (i, j).

Por exemplo, dada a seguinte matriz 5×5

$$A = \begin{bmatrix} 9 & 4 & 5 & 0 & 8 \\ 10 & 3 & 2 & 1 & 7 \\ 9 & 1 & 6 & 3 & 15 \\ 0 & 3 & 8 & 10 & 1 \\ 1 & 16 & 9 & 12 & 7 \end{bmatrix}$$

a vizinhança 3×3 em torno de $(1, 1)$ é a submatriz

$$A_{1,1} = \begin{bmatrix} 9 & 4 & 5 \\ 10 & 3 & 2 \\ 9 & 1 & 6 \end{bmatrix}$$

Note que a vizinhança não é bem definida em algumas coordenadas (por exemplo, em $(0, 0)$).

Definição da transformação. Uma imagem digital pode ser representada por uma matriz. Dada uma matriz A de inteiros positivos com m linhas e n colunas, e dois inteiros positivos e ímpares, p e q, o filtro da mediana calcula uma matriz Med com o mesmo tamanho de A, de forma que $Med(i, j)$ contém a mediana dos números em $A_{i,j}$ (a vizinhança $p \times q$ em torno de (i, j)).

No caso do exemplo anterior, os números em torno de $(1, 1)$ são $9, 4, 5, 10, 3, 2, 9, 1, 6$. Logo, $Med(1, 1) = 5$. Quando a vizinhança de uma coordenada (i, j) não estiver bem definida, usaremos a convenção $Med(i, j) = 0$.

No caso da matriz-exemplo da seção anterior, o resultado do filtro da mediana com uma vizinhança 3×3 é a seguinte matriz:

$$Med = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & 5 & 0 \\ 0 & 3 & 3 & 6 & 0 \\ 0 & 6 & 8 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2 Formato PGM

Neste EP utilizaremos o formato PGM (*portable graymap*) para armazenar imagens em arquivos. Este formato tem duas variações, uma binária (o PGM "normal" ou *raw*) e outra textual (o PGM ASCII ou *plain*) Em ambos os casos, o arquivo deve conter um cabeçalho e a matriz correspondente à imagem. O exemplo a seguir mostra um arquivo PGM textual:

```
P2
5 4
16
9  4  5 0  8
10 3  2 1  7
9  1  6 3  15
1  16 9 12 7
```

A primeira linha do arquivo contém obrigatoriamente uma palavra-chave, que deve ser "`P2`" no caso de um arquivo PGM textual e "`P5`" no caso de um arquivo PGM binário. A segunda linha contém dois números inteiros que indicam o número de colunas e o número de linhas da matriz, respectivamente. A terceira linha contém um número inteiro positivo *maxval*, que deve ser igual ao maior elemento da matriz. Na definição do formato PGM, *maxval* não pode ser maior que 65535. Para fins deste EP, entretanto, *maxval* é no máximo 255. Os demais números do arquivo são os elementos de uma matriz de inteiros com os tons de cinza de cada ponto da imagem. Cada tom de cinza é um número entre 0 e *maxval*, com 0 indicando "negro" e *maxval* indicando "branco".

O formato PGM também permite colocar comentários. Todo o texto que vai desde um caractere '`#`' até (e inclusive) o próximo fim de linha (caractere '`\n`') é um comentário e deve ser ignorado. Este é um exemplo de arquivo PGM textual com um comentário:

```
P2
# feep.pgm
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

O formato PGM binário tem cabeçalho análogo ao do PGM textual, usando a palavra chave "`P5`" em vez da "`P2`". O que muda é o modo como é armazenada a matriz de tons de cinza. No formato PGM textual, essa matriz é guardada como uma seqüência de caracteres ASCII contendo as representações decimais das tonalidades de cinza. No formato PGM binário, a matriz é guardada como uma seqüência de bytes, sendo o valor de cada byte (de 0 a 255) uma tonalidade de cinza. O número de bytes da sequência é exatamente igual ao número de elementos da matriz[1]. Este é um exemplo de arquivo PGM binário:

```
P5
# feep.pgm
24 7
15
```
... (seqüência de 24×7 bytes com as tonalidades de cinza)

Existem dois outros formatos de arquivo muito semelhantes ao PGM: o PBM (portable bitmap), para imagens monocromáticas (só preto e branco, sem tons de cinza), e o PPM (portable pixmap), para imagens coloridas. No primeiro, os elementos da matriz podem assumir apenas os valores 0 e 1. No segundo, os elementos da matriz são triplas de números inteiros positivos correspondentes às intensidades das cores vermelha, verde e azul nos pontos da imagem. Quem quiser saber mais detalhes sobre esses formatos, visite as seguintes páginas:

- `http://en.wikipedia.org/wiki/Portable_pixmap`
- `http://netpbm.sourceforge.net/doc/pbm.html`
- `http://netpbm.sourceforge.net/doc/pgm.html`
- `http://netpbm.sourceforge.net/doc/ppm.html`

3 O que o seu programa deve fazer

Seu programa deve pedir que o usuário digite o nome do arquivo de entrada, que poderá ser um arquivo PGM textual ou binário, mostrar na tela a imagem contida nesse arquivo e apresentar ao usuário um menu com as escolhas que ele poderá fazer interativamente. Após cada escolha, o programa efetuará a operação solicitada e, caso a opção escolhida não tenha sido "sair", mostrará na tela uma nova imagem. Este é um exemplo de uma possível utilização do programa:

[1] Na verdade, essa descrição se aplica apenas a arquivos PGM com $maxval \leq 255$, que são os considerados neste EP. No caso $256 \leq maxval \leq 65535$, a matriz é guardada como uma seqüência de pares de bytes e o número de bytes da sequência é o dobro do número de elementos da matriz.

```
Digite o nome do arquivo de entrada: exemplo.pgm
Para prosseguir, feche a janela do visualizador...

Opcões:
  0 - ler nova imagem                    5 - rotação de 180 graus
  1 - inversão horizontal                6 - filtro da mediana
  2 - inversão vertical                  7 - salvar em arquivo PGM normal
  3 - rotação horária de 90 graus        8 - salvar em arquivo PGM ASCII
  4 - rotação anti-horária de 90 graus   9 - sair

Escolha uma opção: 6
Digite a semi-variação horizontal: 1
Digite a semi-variação vertical  : 1
Para prosseguir, feche a janela do visualizador...

Opcões:
  0 - ler nova imagem                    5 - rotação de 180 graus
  1 - inversão horizontal                6 - filtro da mediana
  2 - inversão vertical                  7 - salvar em arquivo PGM normal
  3 - rotação horária de 90 graus        8 - salvar em arquivo PGM ASCII
  4 - rotação anti-horária de 90 graus   9 - sair

Escolha uma opção: 1
Para prosseguir, feche a janela do visualizador...

Opcões:
  0 - ler nova imagem                    5 - rotação de 180 graus
  1 - inversão horizontal                6 - filtro da mediana
  2 - inversão vertical                  7 - salvar em arquivo PGM normal
  3 - rotação horária de 90 graus        8 - salvar em arquivo PGM ASCII
  4 - rotação anti-horária de 90 graus   9 - sair

Escolha uma opção: 8
Digite o nome do arquivo de saida: exemplo2.pgm

Imagem salva em exemplo2.pgm
Para prosseguir, feche a janela do visualizador...

Opcões:
  0 - ler nova imagem                    5 - rotação de 180 graus
  1 - inversão horizontal                6 - filtro da mediana
  2 - inversão vertical                  7 - salvar em arquivo PGM normal
  3 - rotação horária de 90 graus        8 - salvar em arquivo PGM ASCII
  4 - rotação anti-horária de 90 graus   9 - sair

Escolha uma opção: 9
```

Nesse exemplo, o usuário carregou uma imagem de um arquivo, aplicou a essa imagem uma seqüência de duas transformações (filtro da mediana e inversão horizontal), salvou a imagem transformada num arquivo PGM textual e saiu do programa. Os parâmetros "semi-variação horizontal" (Δ_h) e "semi-variação vertical" (Δ_v) definem respectivamente o número de colunas (q) e o número de linhas (p) das vizinhanças empregadas pelo filtro da mediana: $q = 2\Delta_h + 1$ e $p = 2\Delta_v + 1$.

Após cada escolha de opção (exceto pela opção "sair"), o programa deve mostrar na tela uma

imagem. Como o código C para a apresentação de imagens na tela foge ao escopo desta disciplina, você pode usar um visualizador externo para essa tarefa. A função `system`, definida em `<stdlib.h>`, permite que um programa coloque outro em execução

```
#include <stdlib.h>

int system(const char *command);
```

O parâmetro `command` é uma cadeia de caracteres com o comando de acionamento do programa que se deseja executar. Essa cadeia contém os caracteres que você digitaria se fosse acionar o programa pelo teclado. A função `system` só volta para o chamador quando o programa acionado encerrar sua execução.

O arquivo `esqueleto.c`, disponível juntamente com este enunciado, contém um "esqueleto de EP" que mostra como a função `system` pode ser utilizada para acionar um visualizador externo, como o programa `eog` ("Eye of Gnome") ou o `kview` ("KDE Image Viewer") . Como esses são visualizadores de arquivos contendo imagens, é necessário salvar a imagem transformada num arquivo temporário e acionar o visualizador passando o nome desse arquivo como argumento. O arquivo `esqueleto.c` contém, ainda, uma implementação incompleta[2] da leitura de uma imagem de um arquivo e uma implementação da inversão horizontal.

Como a função `system` só volta para o chamador (o seu programa) depois o que programa acionado (o visualizador) encerrar sua execução, é necessário fechar o visualizador para que o controle volte para o seu programa. Esta é a razão das mensagens "`Para prosseguir, feche a janela do visualizador...`". Para o usuário, é inconveniente ter de fechar o visualizador! O desejável seria que o menu de opções fosse apresentado e que uma opção pudesse ser escolhida enquanto a imagem estivesse sendo visualizada. A escolha da opção faria com que uma nova imagem fosse visualizada. O arquivo `esqueleto2.c` implementa isso. Embora seja bem curto o código que faz a visualização de uma imagem e a leitura da próxima opção acontecerem em paralelo, esse código envolve conceitos que fogem ao escopo desta disciplina. Mesmo assim, você pode usá-lo se quiser. Essa segunda versão do esqueleto é específica para Linux, pois ela efetua chamadas diretamente ao sistema operacional. (Mais precisamente, ela é para sistemas tipo Unix, como o próprio Unix, o Linux, o FreeBSD e o OS X.) Já a primeira versão do esqueleto chama apenas a biblioteca padrão da linguagem C e, portanto, deve rodar também no Windows[3], desde que ela seja modificada para, em vez do `eog` (que é para Linux), usar algum visualizador de imagens para Windows[4].

4 Sobre a organização do seu programa

(1) Seu programa deve ser dividido em funções razoavelmente curtas, cada uma responsável por uma tarefa não muito grande.

(2) Deve haver uma função para cada uma das transformações sobre imagens implementadas pelo programa.

(3) Para o filtro da mediana você deverá ter também uma função auxiliar, que calcula a mediana dos valores numa vizinhança especificada de uma matriz. A função que implementa o filtro da mediana deve fazer chamadas a essa função auxiliar.

[2] Essa implementação lida apenas com arquivos PGM ASCII e não reconhece comentários.

[3] *Disclaimer*: Eu não testei o `esqueleto.c` no Windows.

[4] O IrfanView é um visualizador de imagens para Windows que lê arquivos PGM e pode ser obtido gratuitamente no sítio `http://www.irfanview.com/`.

(4) Embora os arquivos `esqueleto.c` e `esqueleto2.c` estejam razoavelmente bem organizados, eles quase não contém comentários. Use-os como exemplo no que diz respeito à identação e ao particionamento das tarefas entre funções curtas, mas não no que diz respeito aos comentários! Seu programa deve conter mais comentários! Antes de cada definição de função, coloque um comentário explicando precisa e sucintamente o que a função faz.

(5) Não use variáveis globais em nenhuma das funções do programa.

5 Outras informações

Em `http://www.ime.usp.br/~\/reverbel/mac110-BCC-07/#EPs` você pode encontrar alguns arquivos de entrada contendo imagens com "sujeira pontual", para teste do seu filtro da mediana.

O programa `gimp` (para Linux) é um manipulador de imagens bastante completo, que implementa as transformações pedidas neste EP e inúmeras outras. Com ele você pode, entre muitas outras coisas, converter para o formato PGM imagens em outros formatos.

Observações Importantes

1) Todos os exercícios-programa devem ter um cabeçalho como o seguinte:

```
/*************************************************************/
/* Aluno: Fulano de Tal                                      */
/* Número USP: 12345678                                      */
/* Exercicio-Programa 4 -- Tranformações Sobre Imagens       */
/* MAC110 (BCC) -- 2007 -- Professor: Reverbel               */
/* Compilador: ... (DevC++ ou gcc) versão ...                */
/*************************************************************/
```

2) O exercício-programa é estritamente individual. Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO.

3) Exercícios atrasados não serão aceitos.

4) Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO.

5) É muito importante que seu programa tenha comentários e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A avaliação dos exercícios-programa levará isto em conta.

6) Cada programa deve ter sido executado tantas vezes quantas forem necessárias para testar os vários casos possíveis para as entradas.

7) Você entregará seu exercício-programa através do sistema Paca/Moodle (`http://paca.ime.usp.br`). Para isto você precisa estar cadastrado nesse sistema (use o seu número USP para se cadastrar) e registrado no Paca como aluno da disciplina `MAC110-BCC` (uma vez cadastrado no Paca, basta entrar na área da disciplina `MAC110-BCC` que o sistema perguntará a você se deseja se registrar como aluno da disciplina).

8) Entregue apenas o programa fonte em C, num arquivo com nome `ep4-`<*seu-número-USP*>`.c`. (Exemplo: se seu número USP for 12345678, você deverá entregar um arquivo `ep4-12345678.c`.)

9) Enquanto o prazo de entrega não expirar, você poderá entregar várias versões do mesmo exercício-programa. Apenas a última versão entregue será guardada pelo sistema. Encerrado o prazo, o sistema não aceitará mais a entrega de exercícios-programa. Não deixe para entregar seu exercício na última hora!

10) Guarde uma cópia do seu exercício-programa pelo menos até o final do semestre.

Bom trabalho!