MS211 - Cursão - 2007

Primeiro Exercício Programa

Algumas manipulações simples de imagens

Resumo

Este primeiro EP tem como finalidade relembrar alguns conceitos básicos de programação e de linguagem C. É um exercício de complexidade mediana, mas bastante interessante. A intenção é relembrar, principalmente, as operações com matrizes e com arquivos em disco. O problema trata da manipulação de imagens digitais, algo bastante difundido em nosso cotidiano. Serão introduzidos os formatos PBM, PGM e PPM.

Este texto é um arquivo PDF com hipertextos. Você pode imprimí-lo normalmente. Para poder usar os vários links disponíveis ao longo do texto, ele deve ser aberto em um computador com acesso a internet e com algum leitor de PDF atualizado (Acrobat Reader, xpdf, etc..).

1 Arquivos PBM, PGM e PPM

Os formatos PBM (Portable BitMap), PGM (Portable GreyMap) e PPM (Portable PixMap) são muito convenientes (porém não muito eficientes) para o armazenamento e manipulação de imagens digitais. Sua grande conveniência vem do fato que todas as informações são gravadas em código ASCII (American Standard Code for Information Interchange), isto é, como texto, o que permite a visualização e manipulação de seu conteúdo com os editores convencionais de texto. O fato das informações serem gravadas em ASCII é também a grande deficiência desses formatos, fazendo-os ocupar muito mais espaço do que realmente seria necessário. Mais detalhes sobre estes formatos podem ser encontrados aqui.

Comecemos pelo PBM. Trata-se de um formato para figuras em branco e preto, no qual se pode apenas armazenar (e portanto distinguir) dois tons: branco e preto. Os arquivos PBM possuem duas partes: um cabeçalho (header) e os dados da imagem. O header de um arquivo PBM tem sempre a seguinte forma

```
P1
50 40
```

Os caracteres `P1`, obrigatórios e sempre na primeira linha, indicam que se trata de um arquivo PBM. Os dois inteiros seguintes correspondem, respectivamente, ao número de pixels na horizontal e na vertical. Em outras palavras, correspondem à largura e à altura da figura, em número de pontos. Logo após o header, seguem-se os dados da imagem. No caso do PBM, serão uma seqüência de `0` (branco) ou `1` (preto), dispostos como se a imagem fosse registrada em uma matriz lida linha-a-linha. No caso abaixo, por exemplo,

```
P1
8 8
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
```

temos uma imagem PBM, com 8×8 pixels, sendo que a primeira linha tem pixels preto, branco, preto, branco, etc. Não é dificil concluir que esta figura é um tabuleiro de xadrez. Note que a formatação do arquivo em linhas de facil leitura como o exemplo acima é irrelevante. A mesma figura pode ser representada como

```
P1 8     8
1 0 1 0 1  0 1 0 0 1 0 1 0 1 0 1
1 0 1  0 1 0 1 0 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1
1 0   1 0 1 0 1 0 0 1    0 1 0 1 0 1
```

ou mesmo assim

```
P1 8 8 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 ...
```

É possível armazenar em PBM figuras de considerável complexidade. Uma ainda simples, mas bem mais complexa que o tabuleiro de xadres, é o mostrado na Fig. 1. O arquivo correspondente, de nome `teletubby.pbm` e disponível no repositório de arquivos do curso, tem como primeiras linhas:

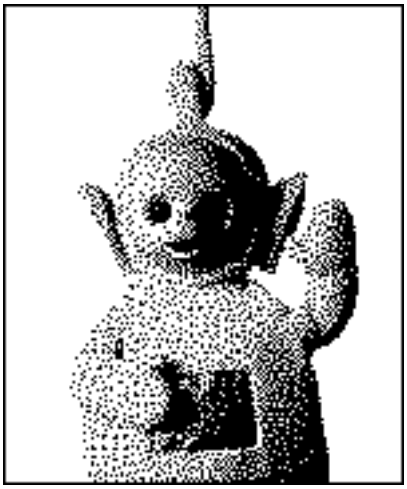

Figura 1: Teletubby! Nosso inseparável companheiro...

```
P1
150 180
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 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 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 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 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 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 0 0 0 0 0
.
.
.
```

Como vemos, trata-se de um arquivo PBM (`P1` na primeira linha), a figura tem 150 pixels de largura e 180 de altura[1].

Muito mais interessante que o formato PBM é o PGM. Nele, as imagens

[1] Os formatos PBM, PGM e PPM podem ser visualizados num ambiente linux sem muitos problemas, basta usar os comandos `display` (pacote ImageMagick), ou o `xview`. Já no Windows, é necessário a instalação de um programa. Eu sugiro o `IrfanView`. É gratís e eficiente. Para instalá-lo, visite esta página. Uma dica: para evitar problemas, associe ao `Irfanview` apenas os arquivos do tipo PBM, PGM e PPM.

são armazenadas em tons de cinza. O tom de cada pixel é representado por um inteiro de 8 bits. O branco corresponde a 255, o preto, a 0. Qualquer outro inteiro intermediário corresponde a uma tonalidade de cinza, sendo os mais claros próximos de 255, os mais escuros próximos a 0. O gradiente de tons tem, portanto, 256 possibilidades diferentes. Os arquivos PGM também tem um header característico, como o abaixo:

```
P2
510 420
255
```

Os caracteres `P2` identificam o formato PGM. Seguem, como no caso anterior, a largura da figura (510 pixels) e sua altura (420 pixels). O terceiro inteiro corresponde ao valor máximo da escala de cinza. Para o nosso caso, sempre será 255 (1 byte). Após o header, seguem os dados da imagem, um conjunto de inteiros positivos menores que 256, correspondendo às tonalidades dos pixels, dispostos linha a linha. Abaixo, temos as primeiras linhas do arquivo de nome `teletubby.pgm`, também disponível no repositório de arquivos do curso, mostrado na Fig. 2, muito mais nítida que a Fig. 1.

Figura 2: Teletubby! Agora em tons de cinza...

```
P2
150 180
```

```
255
 0 6 2 10 0 0 12 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 0
 0 0 0 0 0 0 0 0 0 0 0 0
 .
 .
 .
```

Finalmente, o formato PPM destina-se ao armazenamento e manipulação de imagens coloridas. Possui, também, um header característico:

```
P3
150 180
255
```

Todo arquivo PPM deve ter em sua primeira linha os caracteres `P3`. Seguem, como no PGM, a largura e altura da figura em pixels, e o máximo da escala. A escala de cores, porém, é bem mais complexa que a de tons de cinza. No formato PPM, as cores são representadas segundo o padrão RGB (Red, Green and Blue)[2]. A cor de cada pixel é representada por **três** inteiros positivos menores que 256. Cada inteiro corresponde, respectivamente, à intensidade das componentes Vermelha, Verde e Azul da cor. A terna `255 0 0`, por exemplo, corresponde ao máximo de vermelho, e ausência de Verde e Azul. Trata-se, obviamente, de um vermelho expressivo. O branco corresponde a `255 255 255`, o preto a `0 0 0`. Pode-se representar, desta maneira, $256 \times 256 \times 256 = 2^{32} = 16.777.216$ cores diferentes, representação conhecida como *true colors*. Mais informações sobre o padrão RGB pode ser obtido aqui. Os dados da imagem que seguem o header de um arquivo PPM são, portanto, um conjunto de ternas de inteiros, sendo que cada terna corresponde à cor de um pixel, sempre representado linha-a-linha. A imagem da Fig 3, de nome `teletubby.ppm` e também disponível no repositório de arquivos do curso, tem como linhas iniciais

```
P3
150 180
255
```

[2]A propósito, o funcionamento do padrão RGB esta intimamente ligado à fisiologia do olho humano. Os interessados podem consultar o Vol. 1 das Feynman Lectures on Physics.

Figura 3: Teletubby! Agora em *corolis*...

```
0 0 0 6 6 6 2 2 2 10 10 10
0 0 0 0 0 0 12 12 12 0 0 0
.
.
.
```

A primeira terna `0 0 0` corresponde a um ponto preto. Já a segunda terna, `6 6 6` corresponde a uma cor[3] formada a partir da superposição de um pouco de vermelho, verde e azul, todos em mesma proporção. Trata-se de um tom escuro (próximo ao preto `0 0 0`, longe do branco `255 255 255`). O formato PPM permite a armazenagem e manipulação de imagens muito complexas e ricas.

2 Lendo e gravando arquivos PBM, PGM e PPM

O código em linguagem C abaixo, disponível no repositório de arquivos do curso com o nome `le-pgm.c`, abre um arquivo PGM le seu conteúdo e ar-

[3]Seria esta alguma cor diabólica, presente na singela e inocente representação teletúbica?

mazena os dados da imagem numa matriz de inteiros. Podemos, de maneira análoga, escrever um programa que grava um arquivo PGM a partir de uma matriz de inteiros. A única dica é que o comando de leitura, por exemplo

```
fscanf(arq, "%d %d %d", &m, &n, &max)
```

deve ser substituido por

```
fprintf(arq, "%d %d %d", m, n, max)
```

Notem a diferença nas formas de endereçamento das variáveis inteiras (`&m` e `m`, por exemplo). A partir destes programas básicos, você pode construir outras funções e programas que sejam necessários. A leitura e gravação dos outros formatos são feitas de maneira análoga, respeitando-se sempre os caracteres de identificação `P1` para PBM e `P3` para PPM. Para este último, além disso, os dados podem ser armazenados numa matriz multi-dimensional ou, talvez algo mais conveniente, em três matrizes `Red[MAX][MAX]`, `Green[MAX][MAX]` e `Blue[MAX][MAX]`.

```
/************************************/
/*    le-pgm.c    ASaa@20061129     */
/************************************/
#include <stdio.h>
#include <stdlib.h>
#define MAX_NAME 256 /* tamanho maximo para nome de arquivo */
#define MAX 1020 /* tamanho maximo para a matriz
                    e, consequentemente, para a figura, em pixels)*/

int main(){
FILE *arq;
char  fname[MAX_NAME] ;
char key[128] ;
int i,j,n,m,max, M[MAX][MAX];

printf("Digite o nome do arquivo PGM de entrada: ") ;
scanf("%s", fname) ;

arq = fopen(fname , "r") ;
```

```
if(arq == NULL) {
printf("Erro na abertura do arquivo %s\n", fname) ;
return 0 ;
}

/* le dados do cabecalho */
fscanf(arq, "%s", key) ;
if(strcmp(key,"P2") != 0) {
printf("Arquivo nao e um PGM\n") ;
fclose(arq) ;
return 0 ;
}
fscanf(arq, "%d %d %d", &m, &n, &max) ;

/* le os dados da imagem e armazena na matrix M */
for(i=0; i<=n-1; i++)
   for(j=0;j<=m-1;j++)
        fscanf(arq, " %d ", &M[i][j]);

fclose(arq) ;

return 0;
}
```

3 Algumas manipulações

Uma maneira conveniente de se visualizar o padrão das cores RGB é como um vetor, de componentes inteiras e positivas $\vec{c} = r\hat{\imath} + g\hat{\jmath} + b\hat{k}$. Assim, as cores RGB correspondem aos pontos de um cubo, veja Fig 4. A cada ponto (i, j) da figura temos um vetor $\vec{c}_{ij}$. Por convenção, os índices (i, j) varrem a imagem como os índices de uma matriz: da esquerda para direita (j), de cima para baixo (i). Para outras possíveis representações digitais de cores, vejam também os sistemas HSL e HSV.

É fácil, a partir de uma imagem PPM, obter suas componentes em vermelho, verde e azul. O programa abaixo pode fazê-lo. Aplicando-o a Fig. 3, obtemos a decomposição apresentada na Fig. 5.

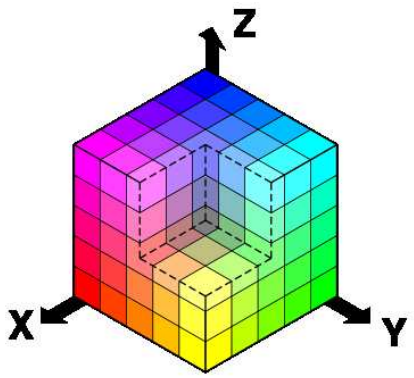

Figura 4: O cubo das cores RGB, com um pequeno cubo contendo o vértice branco 255 255 255 extraído.

```
/*****************************************/
/*    separa-ppm.c    ASaa@20061129      */
/*****************************************/
#include <stdio.h>
#include <stdlib.h>
#define MAX_NAME 256 /* tamanho maximo para nome de arquivo */
#define MAX 1020 /* tamanho maximo para as matrizes
                    e, consequentemente, para a figura, em pixels)*/

int main(){
FILE *arq1 , *arq2 , *arq3 , *arq4;
char  fname1[MAX_NAME] ;
char  fname2[MAX_NAME] ;
char  fname3[MAX_NAME] ;
char  fname4[MAX_NAME] ;
char key[128] ;
int i,j,n,m,max,  r ,g ,b;

printf("Digite o nome do arquivo PPM de entrada: ") ;
scanf("%s", fname1) ;

printf("\nDigite o nome do arquivo PPM de saida (vermelho): ") ;
```

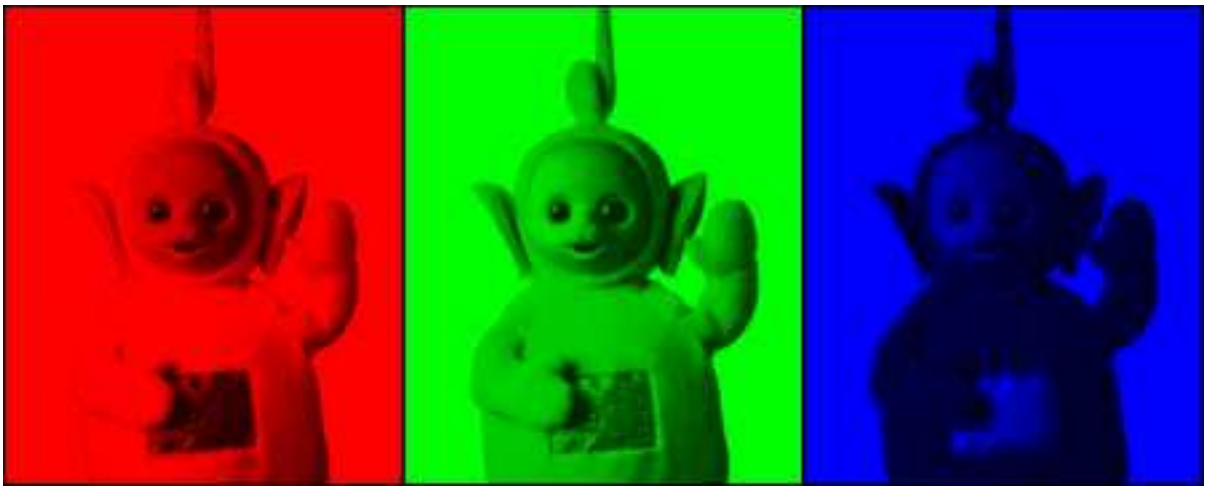

Figura 5: Teletubby! Agora decomposto...

```
scanf("%s", fname2) ;

printf("\nDigite o nome do arquivo PPM de saida (verde): ") ;
scanf("%s", fname3) ;

printf("\nDigite o nome do arquivo PPM de saida (azul): ") ;
scanf("%s", fname4) ;

arq1 = fopen(fname1 , "r") ;
if(arq1 == NULL) {
printf("Erro na abertura do arquivo %s\n", fname1) ;
return 0 ;
}

arq2 = fopen(fname2 , "w") ;
if(arq2 == NULL) {
printf("Erro na abertura do arquivo %s\n", fname2) ;
return 0 ;
}

arq3 = fopen(fname3 , "w") ;
```

```
if(arq3 == NULL) {
printf("Erro na abertura do arquivo %s\n", fname3) ;
return 0 ;
}

arq4 = fopen(fname4 , "w") ;
if(arq4 == NULL) {
printf("Erro na abertura do arquivo %s\n", fname4) ;
return 0 ;
}

fscanf(arq1, "%s", key) ;
if(strcmp(key,"P3") != 0) {
printf("Arquivo nao e um PPM\n") ;
fclose(arq1) ;
return 0 ;
}
fscanf(arq1, "%d %d %d", &m, &n, &max) ;

fprintf(arq2,"P3\n" );
fprintf(arq3,"P3\n" );
fprintf(arq4,"P3\n" );
fprintf(arq2, "%d %d %d\n", m, n, max) ;
fprintf(arq3, "%d %d %d\n", m, n, max) ;
fprintf(arq4, "%d %d %d\n", m, n, max) ;

for(i=0; i<=n-1; i++)
   for(j=0;j<=m-1;j++) {
        fscanf(arq1, " %d %d %d ", &r , &g , &b);
        fprintf(arq2, "  %d %d %d \n " , r , 0 , 0 );
        fprintf(arq3, "  %d %d %d \n " , 0 , g , 0 );
        fprintf(arq4, "  %d %d %d \n " , 0 , 0 , b ); }

fclose(arq1) ;
fclose(arq2) ;
fclose(arq3) ;
fclose(arq4) ;
```

```
return 0;
}
```

4 Atividades do Exercício Programa

Este EP possui três ítens:

Item a: Faça um programa capaz de abrir um arquivo PPM, ler seu conteúdo e armazenar os dados da imagem em três matrizes (uma para cada cor). O programa deve oferecer ao usuário, a partir de menus, as seguintes opções (Dica: use a instrução `switch` do C):

1. girar a imagem de 90 graus, para cima ou para baixo, e gravar o resultado em um arquivo;
2. obter reflexões horizontais e verticais, e gravar o resultado em um arquivo;
3. gerar uma imagem em tons de cinza e gravá-la num arquivo PGM. Defina o tom de cinza t associado a uma cor (r, g, b) como

$$t = \text{int}\left(\frac{r+g+b}{3}\right),$$

   sendo $\text{int}(x)$ a parte inteira de x.
4. usando a definição de tom de cinza acima, gerar uma imagem PGM contendo o negativo da imagem original e gravá-la num arquivo.
5. gerar e gravar uma imagem PBM. Defina o valor p do pixel da imagem PBM a partir do correspondente tom de cinza t como

$$p(t) = \begin{cases} 1, \text{ se } t \leq t_c, \\ 0, \text{ se } t > t_c. \end{cases}$$

   sendo t o tom de cinza correspondente ao pixel no formato PGM e t_c um tom de corte. Para que valores de t_c são obtidos os melhores resultados?
6. decompor uma imagem colorida PPM em três cores (direções) linearmente indepententes arbitrárias (generalizando a decomposição do programa `separa-pgm.c`) e gravá-las em três arquivos diferentes.

Item b: Na figura 6, temos um expressivo personagem da política atual. Para ajudá-lo em sua carreira, escreva um programa que faça as seguintes modificações na foto:

1. altere o fundo. Primeiro, mude apenas a cor. Em seguida, use como fundo a figura 7.
2. É possível consertar o sorriso da foto? (Para discussão nas aulas exploratórias, não é necesário incluir no EP.)

Figura 6: Famosio introdutorio di nóvias teorilicas di linguágia, Seu Creysson foi u homio maizi importantio di todia a histiória du mundiu i du univérsio. Eli foliu presedentio du Brasílico.

Item c: Colorização.

Figura 7: Ambiente predileto do Seu Creysson. (A imagem está reduzida).

1. Inverta o procedimento do item a6, *i.e.*, a partir da decomposição em três cores linearmente independentes, obtenha a imagem colorida original. (Para discussão: este procedimento é semelhante ao usado pelo telescópio Hubble para obter fotos coloridas a partir de suas cameras em branco e preto).
2. Desafio (para discussão nas aulas exploratórias): Como colorizar uma imagem originalmente em tons de cinza?

No repositório de arquivos do curso há algumas imagens em formato PPM que podem ser usadas para testar os programas. Para converter os formatos mais comuns de imagens (jpg, gif, png, etc...) em ppm, pbm, ou pgm, use o comando convert do linux como abaixo:

```
convert -compress none a.jpg a.ppm
```

Bom trabalho!

Agradecimentos

Valéria, Olívia, Ian, Isabel e Cindy Lauper.