ESTRUTURAS DE DADOS E ALGORITMOS

Sumário: Ficha 3 | Duração: 2 Aulas

Introdução

Uma imagem gráfica é composta por um array bi-dimensional (matriz) de pontos, designados por pixeis. Conforme o formato de imagem, cada pixel pode ser representado por um único bit (imagem a preto e branco), um valor inteiro entre 0 e 255 (tons de cinzento) ou por um conjunto de três números inteiros representando as componentes em vermelho, verde e azul (RGB em Inglês).

Para este trabalho pretende-se desenvolver um programa capaz de abrir ficheiros de imagem em tons de cinzento, manipular as imagens e voltar a guardar o resultado em novos ficheiros. Devido à sua simplicidade, foi escolhido o formato de ficheiros Portable Gray Map (PGM/Ascii), cuja especificação pode ser consultada em http://netpbm.sourceforge.net/doc/pgm.html .

A listagem seguinte corresponde é imagem de uma bola de futebol apresentada do lado direito, com uma resolução de 32 linhas por 32 colunas de pixéis. A imagem possui 256 tons de cinzento, em que o valor zero corresponde à cor preto e 255 à cor branco.

```
P2
# Comentario
32 32
255
255 255 255 255 255 255 255 255 255 255 255  99  76  59  52  49  41  45  59  76  99 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 110  74  48  79  98 113 115 115 114  97  80  68  53  41  74 110 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255  82  54  95 115 115 115 115 115 110  94 115 115 115 115 115  91  41  82 255 255 255 255 255 255 255 255
255 255 255 255 255 110  57  82 114 115 115 115 113  93  62   7   0  71 115 115 115 115 115 112  81  57 110 255 255 255 255 255 255
255 255 255 255 105  38  89 115 115 115  98  70  21   0   0   0   0   0  64 114 115 115 115 115 115  98  43 105 255 255 255 255 255
255 255 255 110  38  56 115 115 115  71   0   0   0   0   0   0   0   0  58 112 115 115 115 115 115 100  46 110 255 255 255 255 255
255 255  54  35 109 115 115 115  72   0   0   0   0   0   0   0   0   0   0  53 111 115 115 115 115 115  93  54 255 255 255 255 255
255  82  15 103 115 115 115 115  75   0   0   0   0   0   0   0   0   0   0   0  73 115 115 115 115 115  85   0  82 255 255 255 255
110  20  91 115 115 115 115 115  78   0   0   0   0   0   0   0   0   0   0  27 109 115 115 115 115 115  63   0  20 110 255 255 255
 74  76 115 115 115 115 115 115  82   0   0   0   0   0   0   0   0   0   0  94 115 115 115 115 115 115  62   0   0  74 255 255 255
 47 114 115 115 115 115 115 115  83   0   0   0   0   0   0   0   0   0  70 115 115 115 115 115 115 115  92   0   0  23 255  99 255
 76 115 115 115 115 115 115 115  86   0   0   0   0   0   0   0   0   0  38 112 115 115 115 115 115 115 115 112  18   0   0  99  76 255
 98 115 115 115 115 115 115 115 114 104  92  78  58  26   0   0   2 100 115 115 115 115 115 115 115 115 115  68   0   0  76  59 255
111 115 115 115 115 115 115 115 115 115 115 115 115 115 115 108  96  94 115 115 115 115 115 115 115 115 115 115  96   0   0  59  49 255
115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 114  28   0  45  48 255
104 114 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115  75   0  23  48 255
 49  30  75 100 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 113  98  39  46 255
 44   0   0   0  61 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115  49  56 255
 21   0   0   0   2 107 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 109  59  77 255
  0   0   0   0   0  81 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115  98  76  98 255
  0   0   0   0   0  36 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115  92  66  78  87  96 104 111 115 115 115 115  76  99 255 255
 17   0   0   0   0   0  97 115 115 115 115 115 115 115 115 115 115 115 115 115 115  99   9   0   0   0   0   0   0  77 115 115 115 115  43 255 255 255
 76   0   0   0   0   0  66 115 115 115 115 115 115 115 115 115 115 115 115 115 105  22   0   0   0   0   0   0   0   0  90 115 115  96  74 255 255 255
109  13   0   0   0   0  11 111 115 115 115 115 115 115 115 115 115 108  33   0   0   0   0   0   0   0   0   0 103 115 114  48 110 255 255 255
255  82   0   0   0   0  21 104 115 115 115 115 115 115 115 111  45   0   0   0   0   0   0   0   0   0   4 113 115  84  82 255 255 255 255
255 255  49   0   0  36 105 115 115 115 115 115 115 113  55   0   0   0   0   0   0   0   0   0   0   0  45 115  98  56 255 255 255 255 255
255 255 108  35  48 109 115 115 115 115 115 115 115 115 114  64   0   0   0   0   0   0   0   0   0   0  68 105  46 110 255 255 255 255 255
255 255 255 105  44  98 115 115 115 115 115 115 115 115 115 114  61   0   0   0   0   0   0   0  25  77  89  48 105 255 255 255 255 255 255
255 255 255 255 109  53  84 114 115 115 115 115 115 115 115 115 113  57   0   0   0  45  89 111  84  56 110 255 255 255 255 255 255 255
255 255 255 255 255 255  82  46  94 115 115 115 115 115 115 115 112  55  64  99 115  94  48  82 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 108  75  46  76  98 111 115 115 115 115 115 111  98  76  46  74 110 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255  95  77  53  43  46  48  49  59  76  99 255 255 255 255 255 255 255 255 255 255 255 255 255 255
```

O conteúdo do ficheiro é composto pelos seguintes elementos:

1. A primeira linha contém sempre o texto “P2”, que identifica o formato da imagem PGM/ASCII
2. A segunda linha começa pelo carácter # e contém um comentário. Todas as restantes linhas que comecem pelo caracter # devem ser ignoradas, porque são comentários.
3. A linha seguinte contém dois números inteiros com a dimensão da imagem: número de colunas e número de linhas
4. A linha seguinte contém a quantidade máxima de cores da imagem, correspondendo ao valor da cor branco. Para os efeitos deste trabalho vamos considerar que é sempre 255.
5. Finalmente temos uma sequência de números inteiros entre 0 e 255, correspondente aos valores da intensidade de cinzento de cada pixel da imagem. Os valores podem aparecer no ficheiro sob a forma de uma matriz como no exemplo apresentado, ou sequencialmente, um valor por linha. Em ambos os casos os valores são guardados sequencialmente por linhas, começando no topo e terminando na parte inferior da imagem. Dentro de cada linha, são guardados da esquerda para a direita.

Existem muitas aplicações com capacidade para abrir e visualizar ficheiros PGM, como por exemplo Draw do Libre-office, Paintshop-pro, Gimp, browsers Web, Irfanview, etc.

Requisitos do programa

Para este trabalho pretende-se que os alunos implementem uma classe em C++ para representar uma imagem e que façam utilização da classe Matrix que desenvolveram na Ficha anterior.

Para isso, primeiramente é necessário alterar a classe Matrix definida na Ficha nº 2 nos seguintes métodos:

Ficheiro Matrix.h

```
#pragma once
// Definição da classe Matriz que contem as operações
// tipicas de matrizes

class Matrix {
private:
    . . .
public:
       . . .
       int GetRows( );
       int GetCols( );

       void SetElement(int i, int j, int value);
       int GetElement(int i, int j);

       bool ReadFromFile(FILE* pFile);
       bool SaveToFile(FILE* pFile);
};
```

Estes métodos devem implementar a seguinte funcionalidade:
P1 – A função **GetRows( )** deve devolver o número de linhas da respetiva matriz.
P2 – A função **GetCols( )** deve devolver o número de colunas da respetiva matriz.
P3 – A função **SetElement(int i, int j, int value)** deve alterar o valor (i, j) da matriz com o novo valor indicado por value.
P4 – A função **GetElement(int i, int j)** deve devolver o valor (i, j) da matriz.
P5 – A função **ReadFromFile(FILE* pFile)** apenas deve ler a partir do ponto indicado pelo apontador passado em pFile e não deve abrir o ficheiro.
P6 – A função **SaveToFile(FILE* pFile)** apenas deve gravar o conteúdo da matriz tal como indicado na ficha anterior e não deve nem abrir nem fechar o ficheiro.

Seguidamente deve-se implementar a classe Image:

Ficheiro: Imagem.h

```
#pragma once
#include “Matrix.h”

class Image {
private:
   Matrix * pImage;
public:
   Image( );
   Image( int nl, int nc, int bg_color );
   Image( const Image& orig );
   ~Image();

   Image& operator = ( Image& orig );
   int GetRows();
   int GetCols();

   void SetPixel(int i, int j, int value);
   int GetPixel(int i, int j);

   bool ReadFromFile ( const char* file_name );
   bool SaveToFile( const char* file_name );

   int GetAverageColor();
   void SetNegative();
   void ChangeBrigthness( int b );
   void ChangeContrast( int c );
   Image Rot90();
   Image Crop( int min_l, int min_c, int max_l, int max_c );
};
```

Os métodos da classe devem implementar a seguinte funcionalidade:

P1 – O construtor por omissão deve inicializar uma imagem vazia de tamanho nulo.
P2 – O segundo construtor «**Image(int nl, int nc, int bg_color)**» deve criar uma imagem, de dimensão **nl** por **nc** pixeis, com a cor de fundo especificada por **bg_color**.

P3 – O construtor de cópia «**Image(const Image& orig)»** deve criar uma cópia de uma imagem já existente.

P4 – O destrutor **«~Image( )»** deve apagar a matriz criada.

P5 – O **operador = (Image& orig)** deve criar uma cópia da imagem já existente.

P6 – A função **SetPixel (int i, int j, int value)** deve alterar o valor do pixel (i, j) com o novo valor indicado por value.

P7 – A função **GetPixel (int i, int j)** deve devolver o valor do pixel (i, j).

P8 – O método «**ReadFromFile(const char* file_name)»** deve abrir um ficheiro contendo uma imagem em formato PGM. Esta função recebe como parâmetro de entrada o nome de um ficheiro PGM e procede da seguinte forma:

a) Abre o ficheiro
b) Verifica se a primeira linha do ficheiro contém o texto «P2». Em caso negativo deve terminar e devolver o resultado falso.
c) Se a segunda linha começar pelo caracter **#** deve ser ignorada, saltando para a linha seguinte.
d) Ler as dimensões da imagem, que devem ser atribuídas às dimensões da matriz que constitui a imagem. A linha seguinte, que deve conter sempre o valor 255, é ignorada.
e) Se alguma das dimensões for inferior a 1 pixel, a função deve terminar e devolver o resultado falso.
f) Deve alterar a função **Matrix::ReadFromFile** para ler o conteúdo da matriz a partir do ficheiro diretamente a partir da linha dos dados.

    Matrix::ReadFromFile(FILE* pFile)

g) Fechar o ficheiro

P9 – O método «**SaveToFile(const char* file_name)»** deve guardar o conteúdo da imagem para um ficheiro PGM:

a) Este método recebe como parâmetro de entrada o nome/caminho do ficheiro a escrever.
b) A função deve criar um ficheiro PGM de acordo com as regras acima descritas. Na segunda linha deve ser adicionado um comentário com o texto "#Criado por " e o nome dos alunos.
c) A função produz um resultado booleano verdadeiro quando o ficheiro foi corretamente escrito, ou falso quando houve erro ao abrir/criar o ficheiro.
d) Deve alterar o conteúdo da função **Matrix::SaveToFile** para escrever o conteúdo da matriz diretamente para o ficheiro a partir da linha dos dados.

    Matrix::SaveToFile(FILE* pFile)

h) Fechar o ficheiro

P10 – O método **GetAverageColor()** calcula um tom de cinza com a média de todos os pixéis da imagem.

P11 – O método **SetNegative()** inverte o conteúdo da imagem**.** Para o efeito, deverá percorrer todo o conteúdo da matriz e aplicar a formula Y = 255-X a todos os pixeis da imagem.

P12 – O método **ChangeBrightness(int b)** altera o brilho da imagem de acordo com o parâmetro **b**.
O método deve verificar se n está compreendido dentro do intervalo [-50 a +50] e adicionar o valor do parâmetro **b** a todos os pixéis da imagem, tendo o cuidado de nunca deixar o valor de cada pixel sair fora do intervalo [0 – 255].
P13 – O método **ChangeContrast(int c)** altera o brilho da imagem de acordo com o parâmetro **c**. O método deve verificar se n está compreendido dentro do intervalo [-0.5 a +0.5] e aplicar a fórmula «y = 128 + (c+1.0) * (x – 128)» a todos os pixéis da imagem, limitando os valores resultantes ao intervalo [0 – 255].
P14 – O método **Rot90()** cria uma cópia da imagem, rodada de 90 graus. Para o efeito, deve criar uma nova matriz em que o número de linhas e colunas é trocado. De seguida deverá copiar o conteúdo da matriz original para a nova matriz, trocando linhas por colunas.
P15 – O método **Crop(int min_l, int min_c, int max_l, int max_c)** cria uma nova imagem com a cópia de uma parte da imagem orignal, limitada ao rectângulo definido pelos números de linha **min_l** e **max_l** e coluna **min_c** e **max_c**.

P16 – Crie um programa principal (função main) que:

1. Pede o nome do ficheiro de imagem PGM e abre-o.
2. Inverte o conteúdo da imagem (negativo)
3. Cria uma nova matriz com uma cópia rodada da imagem
4. Grava o conteúdo das duas imagens para 2 ficheiros PGM (a original invertida e a rodada 90º)
5. Pede ao utilizador dois valores, de brilho e contraste e aplica-os à imagem original.
6. Guarda o conteúdo da imagem corrigida (brilho/contraste) para outro ficheiro PGM.

Por cada operação realizada deve haver uma mensagem no ecrã que confirme a sua realização.