UNIVERSIDAD DE GRANADA

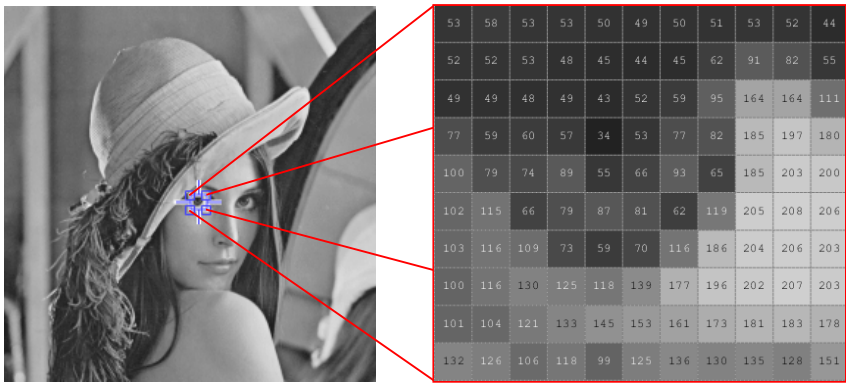
ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

Metodología de la Programación

DGIM

Curso 2021/2022



Guion de prácticas

Memoria dinámica (heap). Zoom In, Zoom out.

Febrero de 2022

Índice

La implementación de la clase Imagen utilizada hasta ahora, almacena los píxeles en un vector cuyo tamaño se fija en tiempo de compilación. En la definición de la clase tenemos lo siguiente:

```
#define IMAGE_MAX_SIZE 200000 ///< Max number of bytes allowd for
class Image {
private:
Byte _data[IMAGE_MAX_SIZE]; ///< Bytes of the image
int _height; ///< number of rows
int _width; ///< number of columms
public:
```

Como consecuencia de esto, cada vez que se instancia un objeto de la clase Imagen, se ”crea” un vector de tamaño `IMAGE_MAX_SiZE` aunque la imagen actual pueda tener unos pocos píxeles ocupados.

Una manera de evitar este problema de desperdicio de memoria es ”solicitar” o ”pedir”, en tiempo de ejecución, solamente la memoria que se va a necesitar (conociendo a priori el tamaño de la imagen), utilizarla y luego ”liberarla”.

Por tanto en esta práctica haremos uso de los conceptos básicos de memoria dinámica a partir de la clase Imagen utilizada en guiones anteriores.

1. Objetivos

El desarrollo de esta práctica pretende servir a los siguientes objetivos:

- Repasar conceptos básicos memoria dinámica.
- Modificar la clase Imagen de prácticas anteriores para incluir gestión de memoria dinámica.

2. Descripción

En esta práctica realizaremos dos tareas:

1. Extender la clase Imagen para que los datos se almacenen en un vector. dinámico.

```
class Image {
private:
Byte * _data; ///< Bytes of the image
int _height; ///< number of rows
int _width; ///< number of columms
```

2. Ampliar la clase imagen con dos funciones nuevas para hacer $ZoomIn$ y $ZoomOut$.

3. Memoria dinámica en la clase Imagen

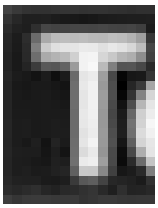
A partir de los ficheros de la práctica anterior, realizar los siguientes campos.

- Reimplementar la clase `Image` utilizando como estructura interna un vector dinámico.
  - Ahora, el constructor de la clase debe reservar memoria (Repase los apuntes de teoría para implementar la reserva de memoria asociada a un vector). Más concretamente, el constructor sin parámetros debe crear una imagen vacía (0 filas, 0 columnas y sin memoria reservada).
  - El constructor con parámetros debe reservar la memoria para la imagen.
  - Por último tenga en cuenta que el método de lectura debe crear la imagen antes de leer los datos.
- Implementar un método `destruir()` que permita liberar la memoria reservada. Note que al destruir la imagen, además de liberar la memoria, también debe poner el número de filas y columnas a cero. Tenga en cuenta que sólo debe liberar la memoria si tuviera reservada, es decir, destruir una imagen ya destruida o vacía no debe producir ningún efecto. Este método debe llamarse explícitamente al final del programa o siempre que sea necesario liberar la memoria. Cuando se imparta en teoría el tema de clases, se verá que la manera correcta de realizar esta tarea es mediante la utilización de un método ”destructor”. Por ahora, aplicaremos esta solución de compromiso.

4. Zoom In

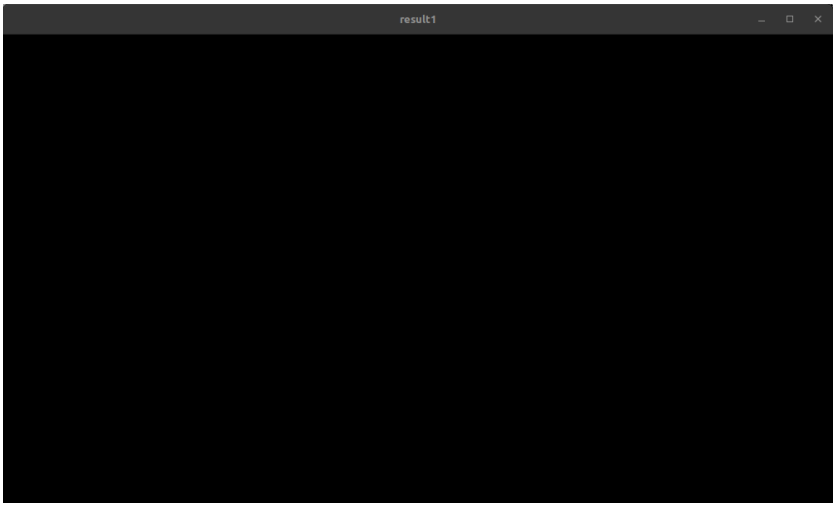
Para hacer la función de aumentar (Zoom In) es necesario seguir los siguientes pasos.

- Cargar la imagen de entrada $input$



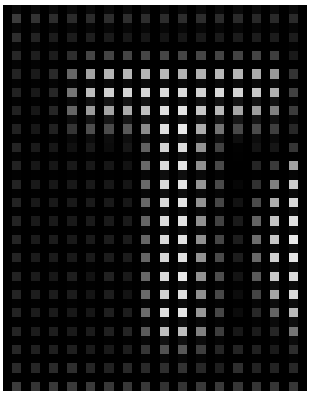
- Crear una nueva imagen $zoom$ cuyo tamaño sea el doble de columnas y el doble de filas

$$\forall i, j \;\; zoom[i][j] = 0$$



- Rellenar solo las posiciones pares en columnas y filas con los píxeles originales de $input$.

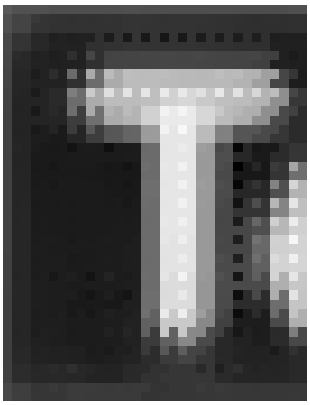
$$\forall i, j, \; i\,\%2 == 0 \&\& j\,\%2 == 0 \;\; zoom[i][j] = input[i/2][j/2]$$



- Rellenar los puntos restantes con una interpolación de los puntos de alrededor. Para ello, se usará el valor balanceado del histograma de la imagen original en un un rectángulo de 3x3 centrado en el píxel en cuestión.

$$\forall i, j,\ i\,\%2! = 0 \&\& j\,\%2! = 0$$

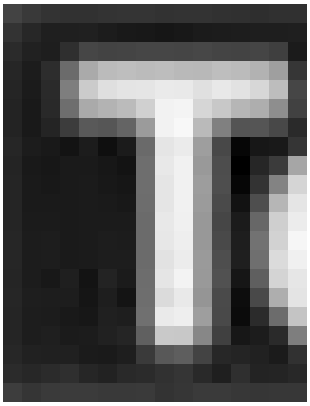
$$zoom[i][j] = input.copyArea(x/2-1, y/2-1, 3, 3).getHistogram().getBalancedLevel()$$



5. Zoom Out

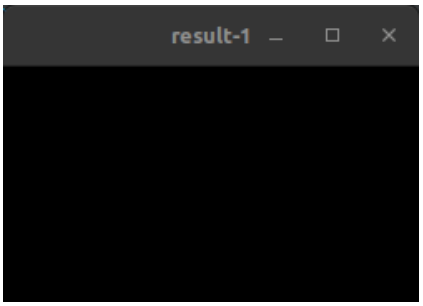
Para hacer la función de reducir (Zoom out) se sigue un procedimiento similar.

- Cargar la imagen de entrada $input$



- Crear una nueva imagen $zoom$ cuyo tamaño sea la mitad de columnas y la mitad de filas

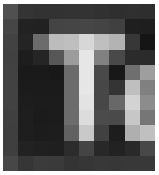
$$\forall i, j \;\; zoom[i][j] = 0$$



- Rellenar cada píxel con una interpolación de los puntos de alrededor en la imagen original.

$$\forall i, j$$

$$zoom[i][j] = input.copyArea(x*2-1, y*2-1, 3, 3).getHistogram().getBalancedLevel()$$



6. Image

```
1  /**
2  @file Image.h
3  @brief Manejo de imÃ¡genes digitales en formato PGM blanco y negro
4  @author MP-DGIM - Grupo A
5   **/
6
7  #ifndef _IMAGE_H_
8  #define _IMAGE_H_
9
10 #include <istream>
11 #include <fstream>
12 #include "Byte.h"
13 #include "Histogram.h"
14
15 #define IMAGE_MAX_SIZE 200000 ///< Max number of bytes allowd for
16 #define IMAGE_DISK_OK 0
17 #define IMAGE_ERROR_OPEN 1
18 #define IMAGE_ERROR_DATA 2
19 #define IMAGE_ERROR_FORMAT 3
20 /**
21 @brief A black and white image
22  */
23 class Image {
24 private:
25     Byte * _data; ///< Bytes of the image
26     int _height; ///< number of rows
27     int _width; ///< number of columms
28
29     /**
30      * @brief It frees the allocated memory
31      */
32     void clear();
33
34     /**
35      * @brief It copies an existing image into this one, resizing it accordingly
36      * @param rhs The source Image to copy from
37      */
38     void copy(const Image &rhs);
39
40     void setSize(int , int );
41
42 public:
43     /**
44      * @brief It builds an empty, image
45      */
46     Image();
47     /**
48      * @brief It builds a fully black image with @a width columns and @a height rows
49      * @param height number of rows
50      * @param width number of columns
51      */
52     Image(int width, int height);
53     /**
54     @brief It gives the number of rows of the image
55     @return number of rows
56      */
57     int height() const;
58     /**
59     @brief It gives the number of columns of the image
60     @return The number of rows
61      */
62     int width() const;
63     /**
64      * @brief It assigns the value @a v to the position(x,y) of the image.It must check that
65      * the values x and y are valid, otherwise, it does not do anythig.
66      * @param x The column
67      * @param y the row
68      * @param v The new value
69      */
70     void setPixel(int x, int y, Byte v);
71     /**
72      * @brief It returns the value of the requested (x,y) position. It must check that
73      * the values x and y are valid, otherwise, it returns a negative value. Please note that
74      * the value returned is a int
75      * @param x The column
76      * @param y the row
77      * @return The value of the pixel in [0-256] or -1 if there is an access error
78      */
79     int getPixel(int x, int y) const;
80     /**
81      * @brief It assigns the value @a v to the linear position i of the image. It must check that
82      * the values i is valid, otherwise, it does not do anythig.
83      * @param i The linear position
84      * @param v The new value
85      */
86     void setPos(int i, Byte v);
87     /**
88      * @brief It returns the value of the requested linear position. It must check that
89      * the value i is valid, otherwise, it returns a negative value. Please note that
90      * the value returned is a int
91      * @param i The linear position
92      * @return The value of the pixel in [0-256] or -1 if there is an access error
93      */
94     int getPos(int i) const;
95     /**
96      * @brief  It sets all pixels of the image to the value given
```

```
97        * @param b The value
98        */
99       void flatten(Byte b);
100      /**
101       * @brief It produces a mesh of vertical and horizontal stripes all along the
102       * image. Every prim pixels it is set to 255 anad every sec pixels
103       * it is set to 127
104       * @param prim Gap between primary mesh
105       * @param sec Gap between secondary mesh, Default value is 0
106       */
107      void mesh(int prim, int sec=0);
108
109      /**
110       * @brief It shows an image in an external window, ready for inspection. It uses
111       * the program display (ImageMagick) to display every image. For an easier identification
112       * process of all images shown are labeled with a title
113       * @param title The title on top of the window
114       */
115      void showInWindow(std::string title);
116      /**
117       * @brief It calculates the hash value of the image and returns it inside a string,
118       * together with its dimension.
119       * @return a string that contains the dimension and the hash value of the image
120       */
121      std::string inspect();
122      /**
123       * @brief It opens a file that contains a PGM Image and reads the data into
124       * a iname in memmory
125       * @param filename Name of the file
126       * @return a code that means the following: 0 - Successful operation.
127       * 1 - Error opening the file
128       * 2 - Error reading the data
129       * 3 - The detected data does not follow the PGM techincal description
130       */
131      int readFromFile(const char filename[]);
132
133      /**
134       * @brief It writes the Image on disk, in PGM ascii format
135       * @param filename The name of the disk file which will contain the image
136       * @return The same code that readFromFile()
137       */
138      int saveToFile(const char filename[]) const;
139      /**
140       * @brief It calculates the histogram of the image, and returns it into an
141       * instance of the class Histogram
142       * @param values
143       */
144      Histogram getHistogram() const;
145     /**
146       * @brief It takes the histogram of the image and depicts a new image with the
147       * visualization of the histogram according to these rules
148       * ~~~~
149       *   +-     +-------------------------------o
150       *   |      |                               | --- 1 pix white line
151       *   |      |   *                           | -+
152       *   |      |   **                          |  |
153       *   |      |   **                          |
154       *   |      |   ***                    *    |   Normalized histogram
155       *   |      |   ***                   **    |     150 pix
156       *   h=160  |   ****    *             **    |  |
157       *          |   *****  ***           ****   |  |
158       *   |      |  ************************** | -+
159       *   |      |-------------------------------| 1 pix  white line
160       *   |      |-------------------------------| -+
161       *   |      |                               |  8 pix Scale of gray levels
162       *   +-     o-------------------------------+ -+
163       *
164       *          |                               |
165       *          +---------- w  = 256 -----------+
166       * ~~~~
167       * @return
168       */
169      Image depictsHistogram() const;
170
171      /**
172       * @brief It segements the histogram by groups whose value is higher than the admitted tolerance and
          returns
173       * an array of images, each of which corresponds to one of these objects
174       * @param set The array of images
175       * @param nimages The number of images found
176       * @param maximages The max number of images to be found
177       * @param tolerance The tolerance addmitted
178       */
179      void extractObjects(Image set [], int &nimages, int maximages, double tolerance=HISTOGRAM_TOLERANCE)
          const;
180
181
182      /**
183       * It returns a binarization of the original image. All pixels  >= the value @a t
184       * are set to 11111111 and the others to 00000000
185       * @param t The threshold
186       * @return A copy of the original image
187       */
188      Image threshold(int t = -1) const;
189      /**
190       * @brief It returns a subimage of the original image given by the parameters.
191       * If the parameters exceed  the dimensions of the original image, then
192       * the cut image should not exceed those limits.
193       * @param x x-coordinate of the top left corner
194       * @param y y-coordinate of the topt left corner
```

```
195         * @param w width of the subimage
196         * @param h height of the subimage
197         * @return The subimage cut
198         */
199        Image copyArea(int x, int y, int w, int h) const;
200
201        /**
202         * @brief It pastes a secondary image into the original image, at the given position,
203         * into the same original image, which results modified.
204         * This operation should maintain the limits of the original image
205         * @param x x-coordinate of the top left corner
206         * @param y y-coordinate of the topt left corner
207         * @param from The second image
208         */
209        void pasteArea(int x, int y, const Image &from, int toneup=-1, int merge=100);
210
211       /**
212        @brief Libera la memoria asociada a un objeto de la clase Imagen
213
214         */
215        ~Image();
216        /**
217         * Copy constructor
218         * @param otra
219         */
220        Image(const Image & otra);
221
222        /**
223         * @brief Assignment operator
224         * @param rhs
225         * @return
226         */
227        Image & operator=(const Image &rhs);
228         /**
229         * @brief It zooms the image in
230         * @return The zoomed image
231         */
232        Image & operator ++();
233        /**
234         * @brief Ti zooms the image out
235         * @return The zoomed image
236         */
237        Image & operator --();
238    };
239    #endif
```

7. Práctica a entregar

Hace exactamente lo mismo que la práctica anterior, con entradas desde la línea de comandos, y añade un parámetro más para el zoom, el cual puede ser $+1$ o -1,

```
practica4 -i <input> [-c <copyfrom>  -z <zoom> -o <output>]
```

- $-i < input >$ Es un parámetro obligatorio y determina qué imagen se considerará como $input$
- $-o < output >$ Es un parámetro opcional. Si no se indica, el resultado sólo aparece en pantalla. Si se indica, además de en pantalla, el resultado se guarda en disco con el nombre indicado
- $-c < copyfrom >$ Es un parámetro opcional. Si aparece, se utiliza la imagen indicada como $copyfrom$, en otro caso, no se hace ningún cambio a la imagen $input$.
- $-z[-1|1]$ Es un parámetro opcional cuyo valor por defecto es 0. Si aparece se utiliza para ampliar o reducir la imagen $input$ antes de hacer la copia desde $copyfrom$
- Todos los parámetros pueden aparecer en cualquier orden.

```
dist/Debug/GNU-Linux/practica4 -i data/telediario.pgm -z 1
-c data/bmw.pgm -o output.pgm

...Reading image from data/telediario.pgm
500x282

[im_input] 500x282 10368250849137031550

...Zooming in

[im_input] 1000x564 14371093910807781907

...Reading image from data/bmw.pgm
135x147

[im_copyfrom] 135x147 4417026241012456264
Thresholding to level 180

[im_bin] 135x147 5849421183935098771
start -1
Found object 0 in [249,255]
start 249
Found object 1 in [174,182]
start 174
Found object 2 in [98,105]
start 98

[im_collection[0]] 135x147 5876811545242803409

...Saving image into output.pgm

[im_output] 1000x564 7954518846368505929
```


output
Telediario

7.1. Tests run

```
---
```