UNIVERSIDAD DE GRANADA

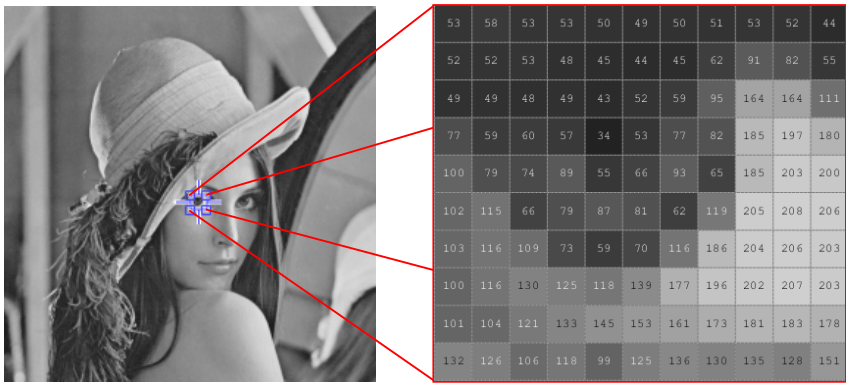
ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

# Metodología de la Programación

DGIM

Curso 2021/2022




# **Guion de prácticas**

*Copiar, pegar, binarizar*

*Febrero de 2022*

# Índice

# 1. Objetivos

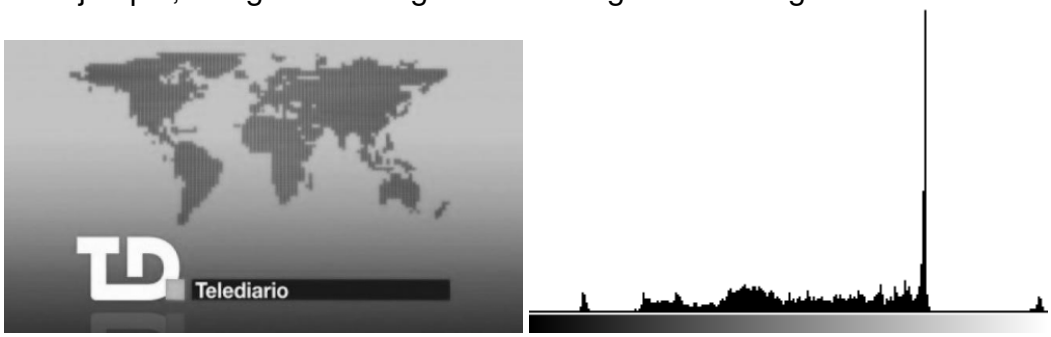
El desarrollo de esta práctica pretende servir a los siguientes objetivos:

- Continuar con otras operaciones basadas en el histograma como la binarización selectiva.
- Explorar la iteración sobre la imagen desde posiciones posiblemente incorrectas sin que esto produzca errores. Para ello se permite copiar un área de la imagen indicando un rectángulo sobre ella. Y pegar esta imagen recortada dentro de otra.

# 2. Binarización adaptativa de la imagen

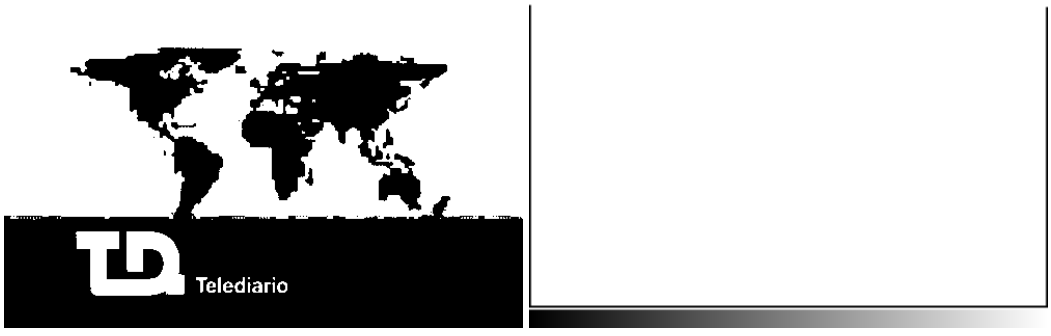
La función de binarizar una imagen a partir de un nivel $t$ permmite transformar en $255$ cualquier nivel estrictamente mayor que $t$ y en 0 los demás, lo que produce una imagen binarizada, solo con dos niveles $0$ y $1$.

Por ejemplo, la siguiente imagen tiene el siguiente histograma




Si la imagen anterior se binariza a 128 se obtiene lo siguiente

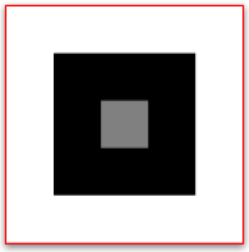

Una imagen también se puede binarizar de forma adaptativa, para ello, se utiliza como $t$ aquél nivel que deja por debajo, o igual a él, al menos a la mitad de la imagen. En este caso, el umbral elegido sería $t = 144$ y produciría este resultado



# 3. Copiar un área de una imagen

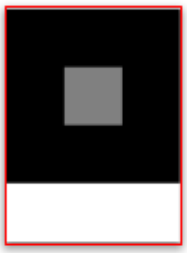
Por ejemplo, dada la siguiente imagen.

| | | | | |
|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 0 | 127 | 0 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 |



se podría generar una copia de la misma en el rectángulo (1,1) con 3 de ancho y 4 de alto, produciendo este resultado.

| | | | | |
|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 0 | 127 | 0 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 |



Pero también se podrían haber copiado los siguientes rectángulos, que involucran a posiciones fuera de la imagen, y el resultado sería el siguiente, sin provocar ningún error

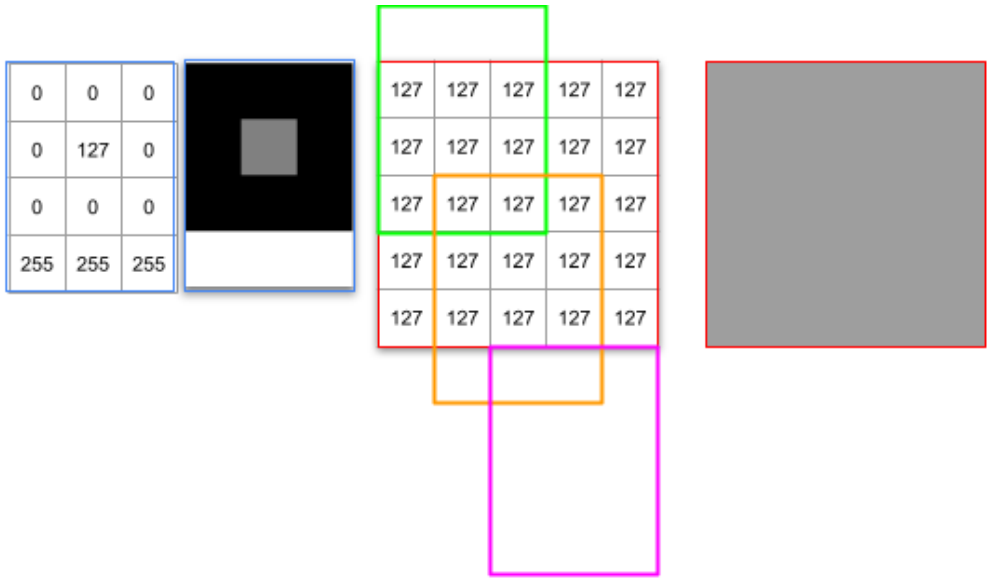
| | | | | |
|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 0 | 127 | 0 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 |

| | | | | |
|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 0 | 127 | 0 | 255 |
| 255 | 0 | 0 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 |

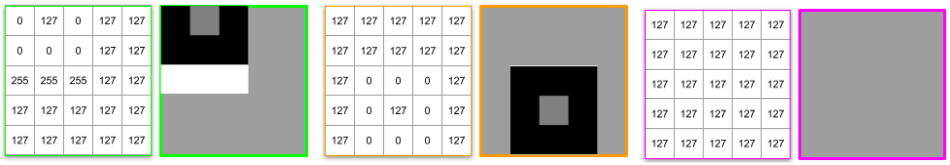
# 4. Pegado selectivo y gradual en otra imagen

Cualquier imagen, independientemente de su procedencia, se puede pegar dentro de otra, de manera que los píxeles de la primera sustituyen a los píxeles de la segunda, teniendo en cuenta lo comentado anteriormente respecto de la posibilidad de indicar posiciones fuera de la imagen.

Por ejemplo si quisiésemos pegar la imagen pequeña (enmarcada en color azul) en la segunda imagen (rojo) en tres posiciones distintas (verde, naranja y rosa)




Los resultados de cada una de ellas, serían estos




## 4.1. Pegado selectivo

Cuando se pega una imagen en otra, se pueden elegir qué niveles se consideran transparentes, es decir, no se van a copiar en la imagen de destino. Para ello lo más sencillo es considerar un nivel de gris de la imagen, $k$, y pegar solo aquellos píxeles cuyo nivel estén por encima de $k$.

## 4.2. Pegado gradual

Normalmente, en una operación de pegado, los píxeles de la primera imagenn sustituyen a los de la segunda. En este caso, se puede indicar que se fusionen a nivel $\alpha \in [0, 100]$ de forma que si $b_1$ es el byte de la primera imagen y $b_2$ es el byte de la segunda imagen, entonces el byte que realmente se pega en la imagen de destino es

$$b_d = \frac{\alpha * b_1 + (100 - \alpha) * b_2}{100}$$

# 5. Histogram

```
1  /**
2   * @file Histogram.h
3   * @author MP
4   */
5  #include <istream>
6  #include <fstream>
7  #include "Byte.h"
8  #include "Image.h"
9
10 #ifndef _HISTOGRAM_H_
11 #define _HISTOGRAM_H_
12
13
14
15
16 /**
17 @brief A black and white histogram
18  */
19 class Histogram {
20 public:
21     static const int HISTOGRAM_LEVELS=256; ///< Max number of bytes allowd for
22     static const double HISTOGRAM_TOLERANCE; ///< Default tolerance
23
24
25     /**
26      * @brief It builds an empty
27      */
28     Histogram();
29     /**
30      * @brief It returns the number of levels in the histogram
31      * @return The number of levels
32      */
33     int size() const;
34     /**
35      * @brief Sets the whole histogram to 0
36      */
37     void clear();
38     /**
39      * @brief It returns the value associated to the level indicated
40      * @param level The level indicated
41      * @return The value associated to the level
42      */
43     int getLevel(Byte level) const;
44     /**
45      * @brief It sets the value associated to the level
46      * @param level The level
47      * @param npixeles The new value
48      */
49     void setLevel(Byte level, int npixeles);
50     /**
51      * @brief It returns the maximum value stored
52      * @return The max of the levels
53      */
54     int getMaxLevel() const;
55     /**
56      * @brief it returns the average value stored
57      * @return The average level
58      */
59     int getAverageLevel() const;
60     /**
61      * It returns a balance level, that is, the level that leaves half of the points
62      * underneath or equal to it.
63      * @return The point of balance of the histogram
64      */
65     int getBalancedLevel() const;
66     /**
67      * @brief It returns a unique hash code for every object so that they might be compared
68      * @return The hash code as an string
69      */
70     std::string inspect() const;
71
72 private:
73     int _data[HISTOGRAM_LEVELS]; ///< datos de la imagen
74
75 };
76 #endif
```

# 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  /**
16  @brief A black and white image
17   */
18  class Image {
19  public:
20      static const int IMAGE_MAX_SIZE=200000; ///< Max number of bytes allowd for
21      static const int IMAGE_DISK_OK=0; ///< Image read/write successful
22      static const int IMAGE_ERROR_OPEN=1; ///< Error opening the file
23      static const int IMAGE_ERROR_DATA=2; ///< Missing data in the file
24      static const int IMAGE_ERROR_FORMAT=3; ///< Unknown image format
25      static const int IMAGE_TOO_LARGE=4; ///< The image is too large and does not fit into memory
26
27
28      /**
29       * @brief It builds an empty, image
30       */
31      Image();
32      /**
33       * @brief It builds a fully black image with @a width columns and @a height rows
34       * @param height number of rows
35       * @param width number of columns
36       */
37      Image(int width, int height);
38      /**
39      @brief It gives the number of rows of the image
40      @return number of rows
41       */
42      int height() const;
43      /**
44      @brief It gives the number of columns of the image
45      @return The number of rows
46       */
47      int width() const;
48      /**
49       * @brief It assigns the value @a v to the position(x,y) of the image.It must check that
50       * the values x and y are valid, otherwise, it does not do anythig.
51       * @param x The column
52       * @param y the row
53       * @param v The new value
54       */
55      void setPixel(int x, int y, Byte v);
56      /**
57       * @brief It returns the value of the requested (x,y) position. It must check that
58       * the values x and y are valid, otherwise, it returns a negative value. Please note that
59       * the value returned is a int
60       * @param x The column
61       * @param y the row
62       * @return The value of the pixel in [0-256] or -1 if there is an access error
63       */
64      int getPixel(int x, int y) const;
65      /**
66       * @brief It assigns the value @a v to the linear position i of the image. It must check that
67       * the values i is valid, otherwise, it does not do anythig.
68       * @param i The linear position
69       * @param v The new value
70       */
71      void setPos(int i, Byte v);
72      /**
73       * @brief It returns the value of the requested linear position. It must check that
74       * the value i is valid, otherwise, it returns a negative value. Please note that
75       * the value returned is a int
76       * @param i The linear position
77       * @return The value of the pixel in [0-256] or -1 if there is an access error
78       */
79      int getPos(int i) const;
80      /**
81       * @brief  It sets all pixels of the image to the value given
82       * @param b The value
83       */
84      void flatten(Byte b);
85      /**
86       * @brief It produces a mesh of vertical and horizontal stripes all along the
87       * image. Every prim pixels it is set to 255 anad every sec pixels
88       * it is set to 127
89       * @param prim Gap between primary mesh
90       * @param sec Gap between secondary mesh, Default value is 0
91       */
92      void mesh(int prim, int sec=0);
93
94      /**
95       * @brief It shows an image in an external window, ready for inspection. It uses
96       * the program display (ImageMagick) to display every image. For an easier identification
```

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

```
195         * @param y y-coordinate of the topt left corner
196         * @param from The second image
197         */
198        void pasteArea(int x, int y, const Image &from, int toneup=-1, int merge=100);
199    private:
200        Byte _data[IMAGE_MAX_SIZE]; ///< Bytes of the image
201        int _height; ///< number of rows
202        int _width; ///< number of columms
203
204    };
205    #endif
```

# 7. Práctica a entregar

## 7.1. Primera parte

Esta parte se entrega junto a la segunda parte en una única entrega. Se ha divido en dos partes para que su implementación sea más gradual.

- Se deben implementar las funciones incluidas en el fichero `Image.h` y en `Histogram.h`
- Leer una imagen de disco, que llamaremos $input$ y otra imagen a copiar, que llamaremos $copyfrom$. Registraremos el ancho $w$ y alto $h$ de $copyfrom$.
- Segmentar la imagen $copyfrom$ en objetos y elegir el primero de la colección de imágenes que resulta, es decir, el de la posición 0, que llamaremos $coleccion$.
- Binarizar $copyfrom$ de forma selectiva, lo que dará otra imagen que llamaremos $bin$
- Pegar $copyfrom$ en la posición $(0, 0)$ de input. Pegar $coleccion$ en la posición $(w + 5, 0)$ y $bin$ en $(w + 5, h + 5)$.
- Pegar $coleccion$, desde el nivel $64$ en adelante, en la posición $(2 * w + 10, 0)$ y $bin$, desde el nivel $64$ en adelante, en $(2w + 10, h + 5)$.
- Pegar $coleccion$, desde el nivel $64$ en adelante y $alpha = 50$, en la posición $(3 * w + 15, 0)$ y $bin$, desde el nivel $64$ y $alpha = 50$, en $(3w + 15, h + 5)$.
- Guarda el resultado en una imagen dentro de la carpeta `data/` llamada `new.pgm`.

### 7.1.1. Ejemplo de ejecución

Si se coge como imagen a copiar la imagen `kfc.pgm`


y como *input* la imagen `telediario.pgm`,

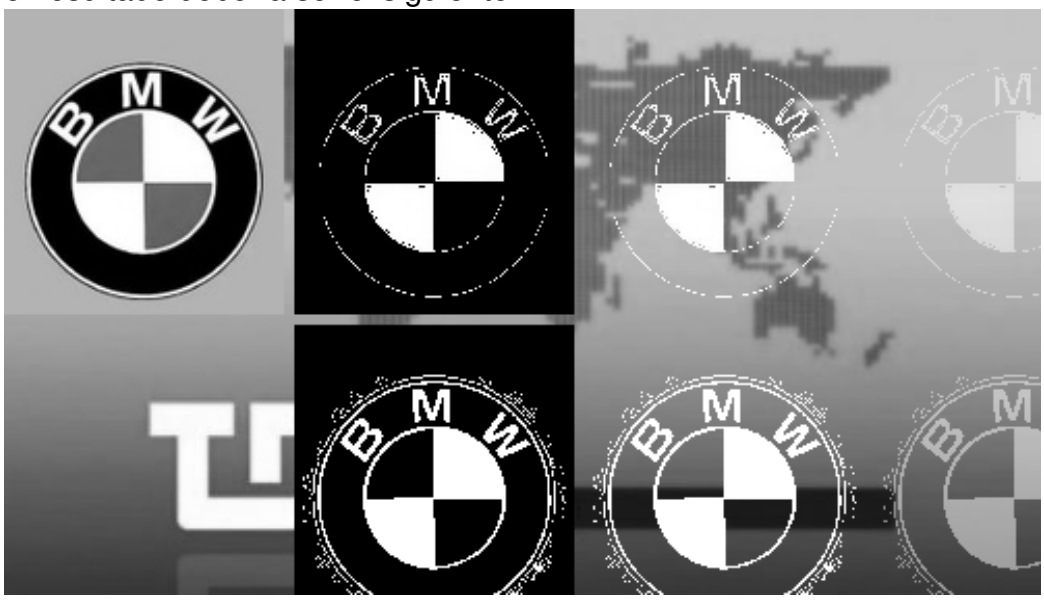

el resultado debería ser el siguiente


```
lcv@numenor:Imaging3: dist/Debug/GNU-Linux/imaging3

[im_input] 500x282 4105296496

...Reading image from data/kfc.pgm
89x84

[im_copyfrom] 89x84 1714063812
Thresholding to level 117

[im_bin] 89x84 3205833621
```

```
Found object 0 in [249,251]
Found object 1 in [229,230]
Found object 2 in [227,228]

[im_collection[0]] 89x84 2366925379

...Saving image into data/new.pgm

[im_output] 500x282 805563168
```

Si, por el contrario, se coge como imagen a copiar la imagen `bmw.pgm`

el resultado debería ser el siguiente



## 7.2. Segunda parte

Esta es la verdadera práctica a entregar. Hace exactamente lo mismo que la parte anterior, pero recibe los datos de entrada desde la línea de comandos

```
p3b -i <input> [-c <copyfrom> -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 $input$.
- Todos los parámetros pueden aparecer en cualquier orden.

```
lcv@numenor:Imaging3b: dist/Debug/GNU-Linux/imaging3b

Error in call:  Missing input file
Please use: -i <input> [-c <copyfrom> -o <output>]

-i <input>
   Input image from <input>
-c <copyfrom>
   Copy clip from <copyfrom>
-o <output>
   Output image to <output>
```

```
lcv@numenor:Imaging3b: dist/Debug/GNU-Linux/imaging3b -i data/telediario.pgm

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

[im_input] 500x282 4105296496

[im_output] 500x282 4105296496
```

```
lcv@numenor:Imaging3b: dist/Debug/GNU-Linux/imaging3b -o data/telebmw.pgm
-i data/telediario.pgm -c data/bmw.pgm

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

[im_input] 500x282 4105296496

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

[im_copyfrom] 135x147 567635164
Thresholding to level 179

[im_bin] 135x147 614964599
Found object 0 in [249,255]
Found object 1 in [174,182]
Found object 2 in [98,105]

[im_collection[0]] 135x147 2959173412

...Saving image into data/telebmw.pgm

[im_output] 500x282 1648917767
```

# 8. TESTS DOCUMENTATION FOR PROJECT Imaging3b

## 8.1. _01_Basics

### 8.1.1. UnitByte_Constructor

1. Declaring a Byte gives 0 by default
2. Declaring a Byte(1) gives 1
3. Declaring a Byte(128) gives 128

### 8.1.2. UnitByte_getValue

1. Declaring a Byte gives 0 by default
2. Declaring a Byte(1) gives 1
3. Declaring a Byte(128) gives 128

### 8.1.3. UnitByte_setValue

1. Declaring a Byte and setting its value to 0 gives 0 by default
2. Declaring a Byte and setting its value to 1 gives 1
3. Declaring a Byte and setting its value to 128 gives 128

### 8.1.4. UnitByte_onBit

1. Given a byte 00000000, activating the 0-bit gives 1
2. Given a byte 00000000, activating the 1-bit gives 2
3. Given a byte 00000000, activating the 7-bit gives 128

### 8.1.5. UnitByte_offBit

1. Given a byte 11111111, deactivating the 0-bit gives 254
2. Given a byte 11111111, deactivating the 1-bit gives 253
3. Given a byte 11111111, deactivating the 7-bit gives 127

### 8.1.6. UnitByte_getBit

1. Given a byte 11111111, querying any bit always give true
2. Given a byte 00000000, querying any bit gives false

### 8.1.7. UnitByte_to_string

1. A byte 11111111 prints as it is
2. A byte 00000000 prints as it is

#### 8.1.8. UnitByte_shiftRByte

1. A byte 11111111 shifted to the right gives 127
2. A byte 11111111 shifted twice to the right gives 63
3. A byte 00000001 shifted to the right gives 0

#### 8.1.9. UnitByte_shiftLByte

1. A byte 11111111 shifted to the left gives 254
2. A byte 11111111 shifted twice to the right gives 252
3. A byte 00000001 shifted to the right gives 2

#### 8.1.10. Image_Constructor

1. and empty data
2. and empty data
3. and empty data

#### 8.1.11. Image_Width

1. gives width
2. gives width
3. gives width

#### 8.1.12. Image_Height

1. gives height
2. gives height
3. gives height

#### 8.1.13. Image_setPixel

1. but should have been
2. but should have been

#### 8.1.14. Image_getPixel

1. but should have been
2. but should have been

#### 8.1.15. Image_getPos

1. but should have been
2. but should have been

#### 8.1.16. Histogram_Constructor

1. A newly created instance of an histogram must be empty
2. A newly created instance of an histogram must be empty hash

#### 8.1.17. Histogram_Size

1. Any histogram must have a capacity for 256 values

#### 8.1.18. Histogram_Clear

1. Any modified histogram must not be empty
2. A crescent triangular histogram is wrong
3. Once filled up, and cleared, an histogram must be empty again

#### 8.1.19. Histogram_getLevel

1. A crescent triangular histogram has wrong values
2. A crescent triangular histogram has wrong values

#### 8.1.20. Histogram_setLevel

1. A crescent triangular histogram is wrong

#### 8.1.21. Histogram_getMaxLevel

1. A crescent triangular histogram has wrong values
2. A crescent triangular histogram has wrong values

#### 8.1.22. Histogram_getAverageLevel

1. A crescent triangular histogram has wrong values
2. A crescent triangular histogram has wrong values

#### 8.1.23. Histogram_getBalancedLevel

1. A crescent triangular histogram has wrong values
2. A crescent triangular histogram has wrong values

### 8.2. _02_Intermediate

#### 8.2.1. UnitByte_onByte

1. Activating a Byte gives 255

#### 8.2.2. UnitByte_offByte

1. Deactivating a Byte gives 0

#### 8.2.3. Image_flatten

1. is wrong
2. is wrong

#### 8.2.4. Image_getHistogram

1. The single pixel image must have one pixel per each 256 gray level
2. The single pixel image must have a maximum histogram of 1
3. The single pixel image must have a balanced level of 128
4. The checkers image must have only 4 levels
5. The checkers image must have a maximum histogram of 64
6. The checkers image must have a balanced level of 86

#### 8.2.5. Image_depictsHistogram

1. The histogram of singlepix Image is wrong
2. The histogram of a flat-128 Image is wrong

#### 8.2.6. Image_threshold

1. of checkers is wrong
2. of singlepix is wrong
3. The balanced threshold of checkers is wrong
4. The balanced threshold of singlepix is wrong

### 8.3. _03_Advanced

#### 8.3.1. UnitByte_encodeByte

1. Activating bits 0,1 and 7 gives 131

#### 8.3.2. UnitByte_decodeByte

1. A byte 131 gives true only in bits 0,1 and 7
2. A byte 131 gives true only in bits 0,1 and 7
3. A byte 131 gives true only in bits 0,1 and 7
4. A byte 131 gives true only in bits 0,1 and 7
5. A byte 131 gives true only in bits 0,1 and 7

#### 8.3.3. UnitByte_decomposeByte

1. Decomposing byte 131 gives 3 active bits
2. Decomposing byte 131 gives 3 active bits
3. Decomposing byte 131 gives 3 active bits
4. Decomposing byte 131 gives 3 active bits

#### 8.3.4. Image_readFromFile

1. Method readFromFIle must warn if a file could not be open
2. Method readFromFIle must warn if a file has a data error
3. Method readFromFIle must warn if a file does not follow the ASCII PGM format
4. Method readFromFIle must warn if a file is too large

5. Method readFromFIle must read valid files with ASCII PGM format
6. Method readFromFIle does not read well valid files with ASCII PGM format

#### 8.3.5. Image_saveToFile

1. Method saveToFilIe must warn if a file could not be open
2. Method saveToFile must save to disk valid ASCII PGM images
3. Method saveToFile must save to disk valid ASCII PGM images

#### 8.3.6. Image_extractObjects

1. The checkers image should decompose into 4 objects
2. of the objects found in checkers image is wrong
3. of the objects found in checkers image is wrong
4. of the objects found in checkers image is wrong
5. of the objects found in checkers image is wrong
6. The flat image should decompose into 1 object

#### 8.3.7. Image_copy

1. Copying the top left corner of chekers must have half width
2. Copying the top left corner of chekers must have half height
3. The top left quarter of checkers is a flat image
4. Copying the bottom right corner of chekers must have half width
5. Copying the bottom right corner of chekers must have half height
6. The bottom right quarter of checkers is a flat image

#### 8.3.8. Image_paste

1. Checkers cannot be built by pastin each quadrant

#### 8.3.9. INTEGRATION_ImageP3b

1. The execution of the program does not produce the expected output
2. Command line arguments may appear in any order
3. The execution of the program does not produce the expected output
4. The output image is wrong
5. The option -i must be mandatory
6. The option -p is wrong

### 8.4. Tests run

```
[==========] Running 38 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 23 tests from _01_Basics
[ RUN      ] _01_Basics.UnitByte_Constructor
[       OK ] _01_Basics.UnitByte_Constructor (1 ms)
[ RUN      ] _01_Basics.UnitByte_getValue
[       OK ] _01_Basics.UnitByte_getValue (0 ms)
[ RUN      ] _01_Basics.UnitByte_setValue
```

```
[       OK ] _01_Basics.UnitByte_setValue (1 ms)
[ RUN      ] _01_Basics.UnitByte_onBit
[       OK ] _01_Basics.UnitByte_onBit (1 ms)
[ RUN      ] _01_Basics.UnitByte_offBit
[       OK ] _01_Basics.UnitByte_offBit (1 ms)
[ RUN      ] _01_Basics.UnitByte_getBit
[       OK ] _01_Basics.UnitByte_getBit (2 ms)
[ RUN      ] _01_Basics.UnitByte_to_string
[       OK ] _01_Basics.UnitByte_to_string (0 ms)
[ RUN      ] _01_Basics.UnitByte_shiftRByte
[       OK ] _01_Basics.UnitByte_shiftRByte (1 ms)
[ RUN      ] _01_Basics.UnitByte_shiftLByte
[       OK ] _01_Basics.UnitByte_shiftLByte (1 ms)
[ RUN      ] _01_Basics.Image_Constructor
[       OK ] _01_Basics.Image_Constructor (2 ms)
[ RUN      ] _01_Basics.Image_Width
[       OK ] _01_Basics.Image_Width (2 ms)
[ RUN      ] _01_Basics.Image_Height
[       OK ] _01_Basics.Image_Height (2 ms)
[ RUN      ] _01_Basics.Image_setPixel
[       OK ] _01_Basics.Image_setPixel (1 ms)
[ RUN      ] _01_Basics.Image_getPixel
[       OK ] _01_Basics.Image_getPixel (1 ms)
[ RUN      ] _01_Basics.Image_getPos
[       OK ] _01_Basics.Image_getPos (1 ms)
[ RUN      ] _01_Basics.Histogram_Constructor
[       OK ] _01_Basics.Histogram_Constructor (0 ms)
[ RUN      ] _01_Basics.Histogram_Size
[       OK ] _01_Basics.Histogram_Size (1 ms)
[ RUN      ] _01_Basics.Histogram_Clear
[       OK ] _01_Basics.Histogram_Clear (1 ms)
[ RUN      ] _01_Basics.Histogram_getLevel
[       OK ] _01_Basics.Histogram_getLevel (0 ms)
[ RUN      ] _01_Basics.Histogram_setLevel
[       OK ] _01_Basics.Histogram_setLevel (1 ms)
[ RUN      ] _01_Basics.Histogram_getMaxLevel
[       OK ] _01_Basics.Histogram_getMaxLevel (0 ms)
[ RUN      ] _01_Basics.Histogram_getAverageLevel
[       OK ] _01_Basics.Histogram_getAverageLevel (1 ms)
[ RUN      ] _01_Basics.Histogram_getBalancedLevel
[       OK ] _01_Basics.Histogram_getBalancedLevel (0 ms)
[----------] 23 tests from _01_Basics (22 ms total)

[----------] 6 tests from _02_Intermediate
[ RUN      ] _02_Intermediate.UnitByte_onByte
[       OK ] _02_Intermediate.UnitByte_onByte (0 ms)
[ RUN      ] _02_Intermediate.UnitByte_offByte
[       OK ] _02_Intermediate.UnitByte_offByte (0 ms)
[ RUN      ] _02_Intermediate.Image_flatten
[       OK ] _02_Intermediate.Image_flatten (1 ms)
[ RUN      ] _02_Intermediate.Image_getHistogram
[       OK ] _02_Intermediate.Image_getHistogram (3 ms)
[ RUN      ] _02_Intermediate.Image_depictsHistogram
[       OK ] _02_Intermediate.Image_depictsHistogram (3 ms)
[ RUN      ] _02_Intermediate.Image_threshold
[       OK ] _02_Intermediate.Image_threshold (8 ms)
[----------] 6 tests from _02_Intermediate (15 ms total)

[----------] 9 tests from _03_Advanced
[ RUN      ] _03_Advanced.UnitByte_encodeByte
[       OK ] _03_Advanced.UnitByte_encodeByte (0 ms)
[ RUN      ] _03_Advanced.UnitByte_decodeByte
[       OK ] _03_Advanced.UnitByte_decodeByte (1 ms)
[ RUN      ] _03_Advanced.UnitByte_decomposeByte
[       OK ] _03_Advanced.UnitByte_decomposeByte (1 ms)
[ RUN      ] _03_Advanced.Image_readFromFile
[       OK ] _03_Advanced.Image_readFromFile (6 ms)
[ RUN      ] _03_Advanced.Image_saveToFile
[       OK ] _03_Advanced.Image_saveToFile (2 ms)
[ RUN      ] _03_Advanced.Image_extractObjects
[       OK ] _03_Advanced.Image_extractObjects (7 ms)
[ RUN      ] _03_Advanced.Image_copy
[       OK ] _03_Advanced.Image_copy (4 ms)
[ RUN      ] _03_Advanced.Image_paste
```

```
[       OK ] _03_Advanced.Image_paste (3 ms)
[ RUN      ] _03_Advanced.INTEGRATION_ImageP3b
[       OK ] _03_Advanced.INTEGRATION_ImageP3b (225 ms)
[----------] 9 tests from _03_Advanced (249 ms total)

[----------] Global test environment tear-down
[==========] 38 tests from 3 test suites ran. (286 ms total)
[  PASSED  ] 38 tests.
```