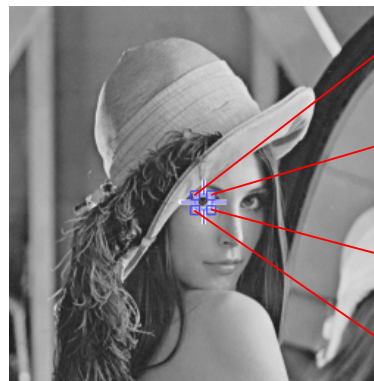




# Metodología de la Programación

DGIM

Curso 2021/2022



53	58	53	53	50	49	50	51	53	52	44
52	52	53	48	45	44	45	62	91	82	55
49	49	48	49	43	52	59	95	164	164	111
77	59	60	57	34	53	77	82	185	197	180
100	79	74	89	55	66	93	65	185	203	200
102	115	66	79	87	81	62	119	205	208	206
103	116	109	73	59	70	116	186	204	206	203
100	116	130	125	118	139	177	196	202	207	203
101	104	121	133	145	153	161	173	181	183	178
132	126	106	118	99	125	136	130	135	128	151

## Guion de prácticas

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

*Febrero de 2022*



# Índice

<b>1. Definición del problema</b>	<b>6</b>
1.1. Arquitectura de la práctica . . . . .	6
<b>2. Objetivos</b>	<b>6</b>
<b>3. Memoria dinámica en la clase Imagen</b>	<b>7</b>
<b>4. Zoom In</b>	<b>8</b>
<b>5. Zoom Out</b>	<b>10</b>
<b>6. Image</b>	<b>12</b>
<b>7. Práctica a entregar</b>	<b>15</b>
<b>8. TESTS DOCUMENTATION FOR PROJECT Imaging4</b>	<b>17</b>
8.1. _01_Basics . . . . .	17
8.1.1. UnitByte_Constructor . . . . .	17
8.1.2. UnitByte_getValue . . . . .	17
8.1.3. UnitByte_setValue . . . . .	17
8.1.4. UnitByte_onBit . . . . .	17
8.1.5. UnitByte_offBit . . . . .	17
8.1.6. UnitByte_getBit . . . . .	17
8.1.7. UnitByte_to_string . . . . .	17
8.1.8. UnitByte_shiftRByte . . . . .	18
8.1.9. UnitByte_shiftLByte . . . . .	18
8.1.10. Image_Constructor . . . . .	18
8.1.11. Image_Width . . . . .	18
8.1.12. Image_Height . . . . .	18
8.1.13. Image_setPixel . . . . .	18
8.1.14. Image_getPixel . . . . .	18
8.1.15. Image_getPos . . . . .	18
8.1.16. Histogram_Constructor . . . . .	18
8.1.17. Histogram_Size . . . . .	19
8.1.18. Histogram_Clear . . . . .	19
8.1.19. Histogram_getLevel . . . . .	19
8.1.20. Histogram_setLevel . . . . .	19
8.1.21. Histogram_getMaxLevel . . . . .	19
8.1.22. Histogram_getAverageLevel . . . . .	19
8.1.23. Histogram_getBalancedLevel . . . . .	19
8.2. _02_Intermediate . . . . .	19
8.2.1. UnitByte_onByte . . . . .	19
8.2.2. UnitByte_offByte . . . . .	19
8.2.3. Image_flatten . . . . .	19
8.2.4. Image_getHistogram . . . . .	20
8.2.5. Image_depictsHistogram . . . . .	20
8.2.6. Image_threshold . . . . .	20
8.3. _03_Advanced . . . . .	20



8.3.1. UnitByte_encodeByte . . . . .	20
8.3.2. UnitByte_decodeByte . . . . .	20
8.3.3. UnitByte_decomposeByte . . . . .	20
8.3.4. Image_readFromFile . . . . .	20
8.3.5. Image_saveToFile . . . . .	21
8.3.6. Image_extractObjects . . . . .	21
8.3.7. Image_copy . . . . .	21
8.3.8. Image_paste . . . . .	21
8.3.9. Image_ZoomIn . . . . .	21
8.3.10. Image_ZoomOut . . . . .	21
8.3.11. INTEGRATION_ImageP3b . . . . .	21
8.4. Tests run . . . . .	22



## 1. Definición del problema

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 allowed for
class Image {
private:
    Byte _data[IMAGE_MAX_SIZE]; ///< Bytes of the image
    int _height; ///< number of rows
    int _width; ///< number of columns
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 cantidad de 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 para la definición e implementación de la clase Image. De ahora en adelante los objetos de esta clase se ubicarán en el heap.

### 1.1. Arquitectura de la práctica

Como ya se indicó con anterioridad, la práctica Imaging se ha diseñado por etapas, las primeras contienen estructuras más sencillas, sobre las cuales se asientan otras estructuras más complejas y se van completando nuevas funcionalidades.

En Imaging4, se va hacer un cambio mayor en la clase Image, las modificaciones afectan tanto a la componente de datos como a la implementación de los métodos, aunque de puertas para afuera no se perciban los cambios. El interfaz de la clase se mantendrá inmutable salvo por la incorporación de nuevas funcionalidades también objeto de esta práctica. En resumen, se modifica el bloque B de la Figura 1 mientras que los bloques A' y C no sufren ninguna modificación.

## 2. 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.
- Ampliar la clase Image con dos nuevas funcionalidades *ZoomIn* y *ZoomOut*.

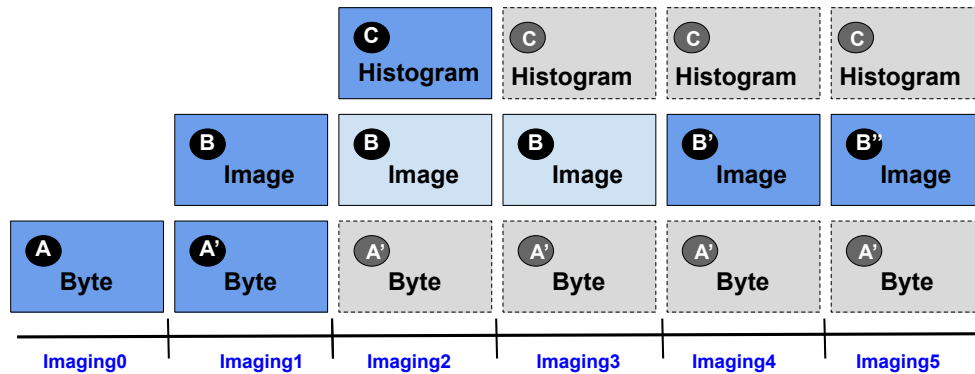


Figura 1: Arquitectura de las prácticas de MP 2022.

### 3. Memoria dinámica en la clase Imagen

Para que el vector se almacene de forma dinámica, la definición de los datos será como sigue:

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

Esto es, la estructura interna, *\_data* es un vector. Lo que implica los siguientes cambios en la componente de métodos.

- 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. Indicar 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.

(LUIS propuesta:) 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.

(SYLV propuesta:) Para terminar, definiremos un método que se conoce como destructor, su identificador comienza con `~` seguido del nombre de la clase. La llamada al destructor es automática, se realiza cada vez que se tiene que destruir un objeto al desaparecer el ámbito en el que se creó. El método incluirá la llamada a nuestro `destruir()` que hemos definido nosotros y que libera específicamente el espacio de un objeto. Este método es indispensable para toda clase que utiliza `new` en alguno sus constructores.

## 4. Zoom In

Para hacer la función de aumentar (Zoom In), que da lugar a una imagen con los detalles de la T del Telediario, es necesario seguir los siguientes pasos.

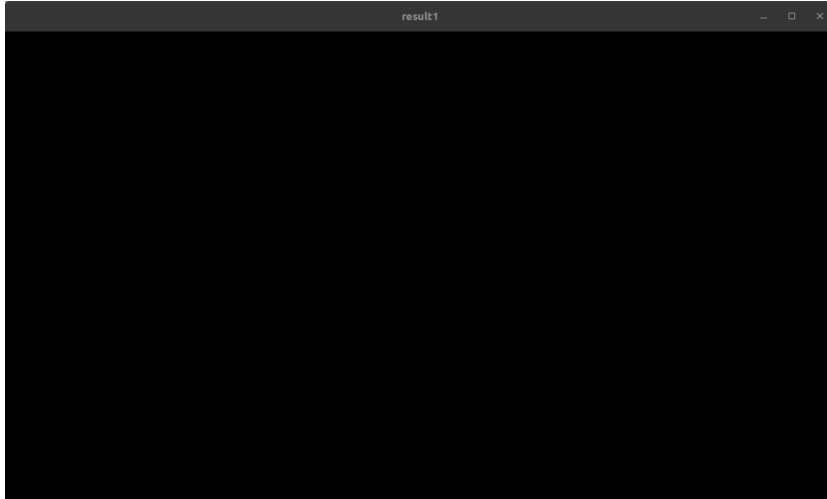
- Cargar la imagen de entrada *input*



- Crear una nueva imagen *zoom* cuyo tamaño sea el doble de columnas y el doble de filas inicialmente a negro

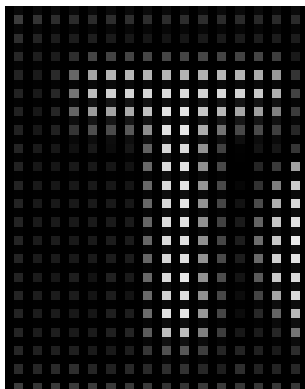
$$\forall i, j \quad zoom[i][j] = 0$$





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

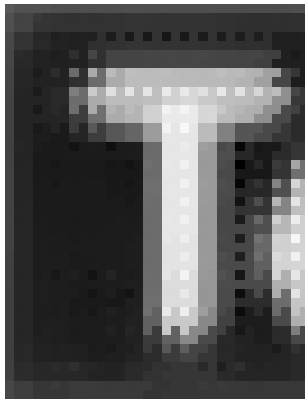
$$\forall i, j, i \% 2 == 0 \ \&\& \ j \% 2 == 0 \quad zoom[i][j] = input[i/2][j/2]$$



- 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 rectángulo de 3x3 centrado en el píxel en cuestión.

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

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

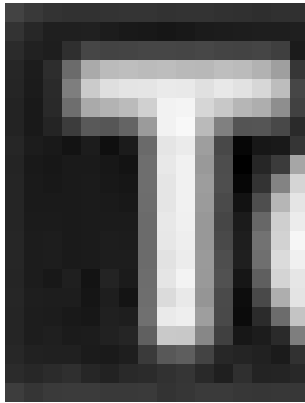


## 5. Zoom Out

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

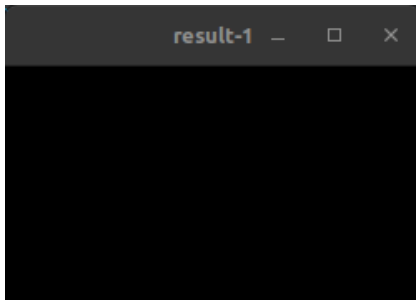
- Cargar la imagen de entrada *input*





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

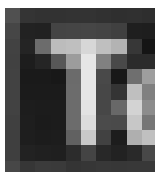
$$\forall i, j \quad zoom[i][j] = 0$$



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

$$\forall i, j$$

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



A continuación se hace una revisión completa de la declaración de la clase *Image*.



## 6. Image

```
1  /**
2  @file Image.h
3  @brief Second version of class Image for management of black and white images in PGM format
4  @note To be implemented by students. Revise the prototype declaration in order
5  * to respect the condition for a correct communication among modules (input/output parameters as copy/
6  reference parameters...)
7  @author MP-DGIM, MP-IADE, MP-II (grupo B)
8  */
9
10 #ifndef _IMAGE_H_
11 #define _IMAGE_H_
12
13 #include <iostream>
14 #include <fstream>
15 #include "Byte.h"
16 #include "Histogram.h"
17
18 /**
19 @brief A black and white image
20 */
21 class Image {
22 public:
23     static const int IMAGE.MAX.SIZE=200000; ///< Max number of bytes allowed for
24     static const int IMAGE.DISK_OK=0; ///< Image read/write successful
25     static const int IMAGE.ERROR_OPEN=1; ///< Error opening the file
26     static const int IMAGE.ERROR_DATA=2; ///< Missing data in the file
27     static const int IMAGE.ERROR_FORMAT=3; ///< Unknown image format
28     static const int IMAGE.TOO_LARGE=4; ///< The image is too large and does not fit into memory
29
30     /**
31      * @brief It builds an empty, image. Modifier method
32      */
33     Image();
34     /**
35      * @brief It builds a fully black image with @a width columns and @a height rows. Modifier method
36      * @param height number of rows. input param
37      * @param width number of columns. input param
38      */
39     Image(int width, int height);
40     /**
41      * @brief It gives the number of rows of the image. Query method
42      * @return number of rows
43      */
44     int height();
45     /**
46      * @brief It gives the number of columns of the image. Query method
47      * @return The number of rows
48      */
49     int width();
50     /**
51      * @brief It assigns the value @a v to the position(x,y) of the image. It must check that
52      * the values x and y are valid, otherwise, it does not do anything. Modifier method
53      * @param x The column
54      * @param y the row
55      * @param v The new value
56      */
57     void setPixel(int x, int y, Byte v);
58     /**
59      * @brief It returns the value of the requested (x,y) position. It must check that
60      * the values x and y are valid, otherwise, it returns a negative value. Please note that
61      * the value returned is a int. Query method
62      * @param x The column. input param
63      * @param y the row. input param
64      * @return The value of the pixel in [0-256] or -1 if there is an access error
65      */
66     int getPixel(int x, int y);
67     /**
68      * @brief It assigns the value @a v to the linear position i of the image. It must check that
69      * the values i is valid, otherwise, it does not do anything. Modifier method
70      * @param i The linear position. input param
71      * @param v The new value. input param
72      */
73     void setPos(int i, Byte v);
74     /**
75      * @brief It returns the value of the requested linear position. It must check that
76      * the value i is valid, otherwise, it returns a negative value. Please note that
77      * the value returned is a int. Query method
78      * @param i The linear position
79      * @return The value of the pixel in [0-256] or -1 if there is an access error
80      */
81     int getPos(int i);
82     /**
83      * @brief It sets all pixels of the image to the value given. Modifier method
84      * @param b The value
85      */
86     void flatten(Byte b);
87     /**
88      * @brief It shows an image in an external window, ready for inspection. It uses
89      * the program display (ImageMagick) to display every image. For an easier identification
90      * process of all images shown are labeled with a title. Query method
91      * @param title The title on top of the window. input param
92      */
93     void showInWindow(std::string title);
94     /**
95      * @brief It calculates the hash value of the image and returns it inside a string,
```

[illegible]



```
194     * the cut image should not exceed those limits. Query method
195     * @param x x-coordinate of the top left corner. input param
196     * @param y y-coordinate of the topt left corner. input param
197     * @param w width of the subimage. input param
198     * @param h height of the subimage. input param
199     * @return The subimage cut
200     */
201     Image copyArea(int x, int y, int w, int h);
202
203     /**
204     * @brief It pastes a secondary image into the original image, at the given position ,
205     * into the same original image, which results modified.
206     * This operation should maintain the limits of the original image. Modifier method.
207     * @param x x-coordinate of the top left corner. input param
208     * @param y y-coordinate of the topt left corner. input param
209     * @param from The second image. input param
210     * @param toneup value. input param
211     * @param merge value for the combination of the 2 images. input param
212     */
213     void pasteArea(int x, int y, Image from, int toneup=-1, int merge=100);
214
215     /**
216     * @brief Destroys the object and frees the memory. Modifier method.
217     */
218     ~Image();
219     /**
220     * @brief Copy constructor. Creates a new Image, copying or duplicating an existing one. Modifier method
221     * .
222     * @param from the image to duplicate. input param.
223     */
224     Image(Image from);
225
226     /**
227     * @brief Assignment operator. Modifier method
228     * @param rhs The right hand side of the assignment expresion. input param
229     * @return The new instance returned by reference.
230     */
231     Image & operator=(Image rhs);
232     /**
233     * @brief It zooms the image in. Query method
234     * @return The zoomed image
235     */
236     Image zoomIn();
237     /**
238     * @brief It zooms the image out. Query method
239     * @return The zoomed image
240     */
241     Image zoomOut();
242 private:
243     Byte * _data; ///< Bytes of the image
244     int _height; ///< number of rows
245     int _width; ///< number of columns
246
247     /**
248     * @brief It frees the allocated memory. Modifier method.
249     */
250     void clear();
251
252     /**
253     * @brief It copies an existing image into this one, resizing it accordingly. Modifier method.
254     * @param rhs The source Image to copy from. Input parameter
255     */
256     void copy(Image rhs);
257
258     /**
259     * @brief it resizes the Image to a new number of rows and columns, clearing
260     * the previously existing data. Modifier method.
261     * @param newheight value. input param
262     * @param newwidth value. input param
263     */
264     void setSize(int , int );
265 };
266 #endif
```

## 7. Práctica a entregar

El programa principal hace exactamente lo mismo que la práctica anterior, con entradas desde la línea de comandos. Sin embargo se añade un parámetro más para el zoom, el cual puede ser +1 o -1,

```
imaging4 -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 (de tipo int) cuyo valor por defecto es 0. Si aparece se utiliza para ampliar o reducir la imagen *input* antes de hacer la copia desde *copyfrom*. Se realiza idéntico tratamiento `showInWindow()` e `inspect()`.
- Todos los parámetros pueden aparecer en cualquier orden.

```
dist/Debug/GNU-Linux/imaging4 -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
```



Podemos explorar las imágenes disponibles en el directorio data/ con `ls data/*pgm` y visualizarlas con el comando en línea de ImageMagick `display`. Ejemplo: `display data/moon.pgm`.  
Veamos una nueva ejecución de `imaging4`:

```
./dist/Debug/GNU-Linux/imaging4 -i data/moon.pgm -z 1 -o data/lunna.pgm  
  
...Reading image from data/moon.pgm  
200x200  
  
[im_input] 200x200 1587823447  
  
...Zooming in  
  
[im_input] 400x400 604554057  
  
...Saving image into data/lunna.pgm  
  
[im_output] 400x400 604554057
```





## 8. TESTS DOCUMENTATION FOR PROJECT Imaging4

### 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 saveToFile 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. Image\_ZoomIn**

1. A zoomed-in image must have twice ts width
2. A zoomed-in image must have twice ts height
3. A zoomed in flat image continues bein flat

#### **8.3.10. Image\_ZoomOut**

1. A zoomed-out image must have half its width
2. A zoomed-out image must have half its height
3. A zoomed out flat image continues bein flat

#### **8.3.11. 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

En esta práctica, al pasar los tests unitarios, no sólo es necesario comprobar que pasa todos los tests, sino que, además, no hay pérdidas de memoria. Para ello, es necesario ejecutar los tests de una forma especial, que incorporen un rastreador de pérdidas de memoria como `valgrind` (Ver el manual de Valgrind en el material de la asignatura). Y también es necesario tener en cuenta que, cuando se realizan tests dentro de NetBeans, se genera un programa binario especial que sólo se usa para pasar los tests. Este programa suele ser este

```
build/Debug/GNU-Linux/tests/TestFiles/fl
```

Si se ejecuta, se puede ver que se pasan los tests unitarios. Pues bien, es **imprescindible** que, desde la línea de comandos del proyecto ejecutemos este mismo programa, pero desde el rastreador `valgrind`. Para ello, se deben ejecutar los tests con esta llamada

```
valgrind --leak-check=full build/Debug/GNU-Linux/tests/TestFiles/fl
```

```
lcv@numenor:Imaging4: valgrind --leak-check=full build/Debug/GNU-Linux/tests/TestFiles/fl
==448074== Memcheck, a memory error detector
==448074== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==448074== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==448074== Command: build/Debug/GNU-Linux/tests/TestFiles/fl
==448074==
[=====] Running 39 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 (134 ms)
[ RUN      ] _01_Basics.UnitByte_getValue
[ OK      ] _01_Basics.UnitByte_getValue (112 ms)
[ RUN      ] _01_Basics.UnitByte_setValue
[ OK      ] _01_Basics.UnitByte_setValue (109 ms)
[ RUN      ] _01_Basics.UnitByte_onBit
[ OK      ] _01_Basics.UnitByte_onBit (109 ms)
[ RUN      ] _01_Basics.UnitByte_offBit
[ OK      ] _01_Basics.UnitByte_offBit (109 ms)
[ RUN      ] _01_Basics.UnitByte_getBit
[ OK      ] _01_Basics.UnitByte_getBit (100 ms)
[ RUN      ] _01_Basics.UnitByte_to_string
[ OK      ] _01_Basics.UnitByte_to_string (86 ms)
[ RUN      ] _01_Basics.UnitByte_shiftRByte
[ OK      ] _01_Basics.UnitByte_shiftRByte (108 ms)
[ RUN      ] _01_Basics.UnitByte_shiftLByte
[ OK      ] _01_Basics.UnitByte_shiftLByte (111 ms)
[ RUN      ] _01_Basics.Image_Constructor
[ OK      ] _01_Basics.Image_Constructor (124 ms)
[ RUN      ] _01_Basics.Image_Width
[ OK      ] _01_Basics.Image_Width (114 ms)
[ RUN      ] _01_Basics.Image_Height
[ OK      ] _01_Basics.Image_Height (116 ms)
[ RUN      ] _01_Basics.Image_setPixel
[ OK      ] _01_Basics.Image_setPixel (84 ms)
[ RUN      ] _01_Basics.Image_getPixel
[ OK      ] _01_Basics.Image_getPixel (110 ms)
```



```
[ RUN      ] _01_Basics.Image_getPos
[ OK       ] _01_Basics.Image_getPos (106 ms)
[ RUN      ] _01_Basics.Histogram_Constructor
[ OK       ] _01_Basics.Histogram_Constructor (102 ms)
[ RUN      ] _01_Basics.Histogram_Size
[ OK       ] _01_Basics.Histogram_Size (51 ms)
[ RUN      ] _01_Basics.Histogram_Clear
[ OK       ] _01_Basics.Histogram_Clear (110 ms)
[ RUN      ] _01_Basics.Histogram_getLevel
[ OK       ] _01_Basics.Histogram_getLevel (87 ms)
[ RUN      ] _01_Basics.Histogram_setLevel
[ OK       ] _01_Basics.Histogram_setLevel (50 ms)
[ RUN      ] _01_Basics.Histogram_getMaxLevel
[ OK       ] _01_Basics.Histogram_getMaxLevel (86 ms)
[ RUN      ] _01_Basics.Histogram_getAverageLevel
[ OK       ] _01_Basics.Histogram_getAverageLevel (84 ms)
[ RUN      ] _01_Basics.Histogram_getBalancedLevel
[ OK       ] _01_Basics.Histogram_getBalancedLevel (95 ms)
[-----] 23 tests from _01_Basics (2309 ms total)

[-----] 6 tests from _02_Intermediate
[ RUN      ] _02_Intermediate.UnitByte_onByte
[ OK       ] _02_Intermediate.UnitByte_onByte (59 ms)
[ RUN      ] _02_Intermediate.UnitByte_offByte
[ OK       ] _02_Intermediate.UnitByte_offByte (48 ms)
[ RUN      ] _02_Intermediate.Image_flatten
[ OK       ] _02_Intermediate.Image_flatten (85 ms)
[ RUN      ] _02_Intermediate.Image_getHistogram
[ OK       ] _02_Intermediate.Image_getHistogram (199 ms)
[ RUN      ] _02_Intermediate.Image_depictsHistogram
[ OK       ] _02_Intermediate.Image_depictsHistogram (124 ms)
[ RUN      ] _02_Intermediate.Image_threshold
[ OK       ] _02_Intermediate.Image_threshold (190 ms)
[-----] 6 tests from _02_Intermediate (708 ms total)

[-----] 10 tests from _03_Advanced
[ RUN      ] _03_Advanced.UnitByte_encodeByte
[ OK       ] _03_Advanced.UnitByte_encodeByte (63 ms)
[ RUN      ] _03_Advanced.UnitByte_decodeByte
[ OK       ] _03_Advanced.UnitByte_decodeByte (222 ms)
[ RUN      ] _03_Advanced.UnitByte_decomposeByte
[ OK       ] _03_Advanced.UnitByte_decomposeByte (144 ms)
[ RUN      ] _03_Advanced.Image_readFromFile
[ OK       ] _03_Advanced.Image_readFromFile (389 ms)
[ RUN      ] _03_Advanced.Image_saveToFile
[ OK       ] _03_Advanced.Image_saveToFile (165 ms)
[ RUN      ] _03_Advanced.Image_extractObjects
[ OK       ] _03_Advanced.Image_extractObjects (242 ms)
[ RUN      ] _03_Advanced.Image_copy
[ OK       ] _03_Advanced.Image_copy (271 ms)
[ RUN      ] _03_Advanced.Image_paste
[ OK       ] _03_Advanced.Image_paste (87 ms)
[ RUN      ] _03_Advanced.Image_ZoomIn
[ OK       ] _03_Advanced.Image_ZoomIn (567 ms)
[ RUN      ] _03_Advanced.Image_ZoomOut
[ OK       ] _03_Advanced.Image_ZoomOut (267 ms)
[-----] 10 tests from _03_Advanced (2420 ms total)

[-----] Global test environment tear-down
[=====] 39 tests from 3 test suites ran. (5481 ms total)
[ PASSED ] 39 tests.

==448074==
==448074== HEAP SUMMARY:
==448074==    in use at exit: 0 bytes in 0 blocks
==448074==   total heap usage: 15,991 allocs, 15,991 frees, 18,935,640 bytes allocated
==448074==
==448074== All heap blocks were freed -- no leaks are possible
==448074==
==448074== For lists of detected and suppressed errors, rerun with: -s
==448074== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```