UNIVERSIDAD DE GRANADA

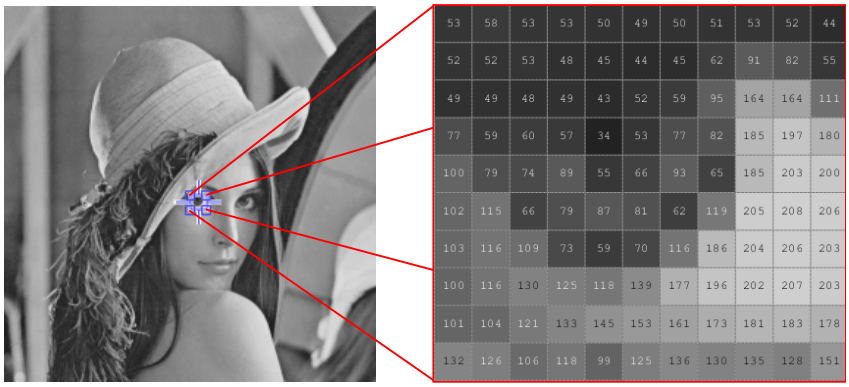
ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

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

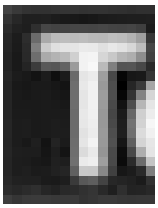
A partir de los ficheros de la práctica anterior, realizar los siguientes campos.

- 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

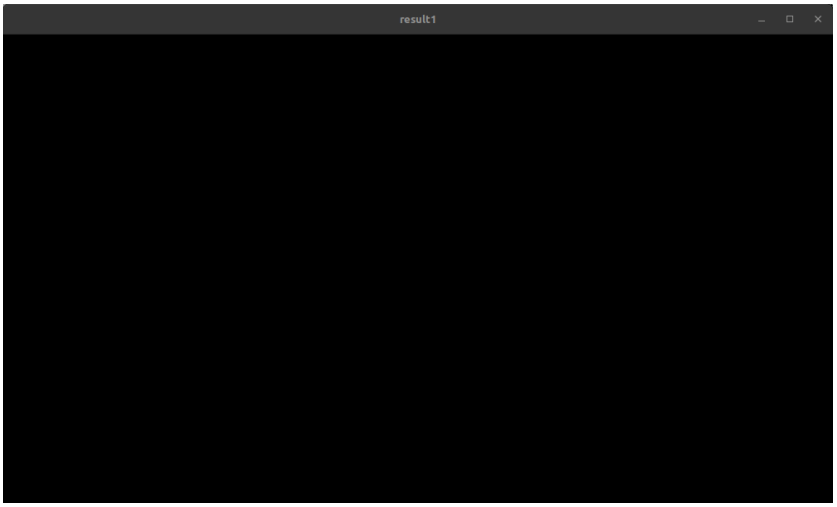
Para hacer la función de aumentar (Zoom In) es necesario seguir los siguientes pasos.

- Cargar la imagen de entrada $input$



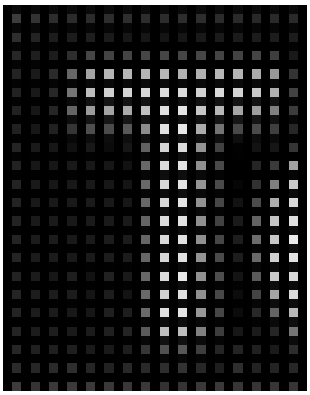
- Crear una nueva imagen $zoom$ cuyo tamaño sea el doble de columnas y el doble de filas

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



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

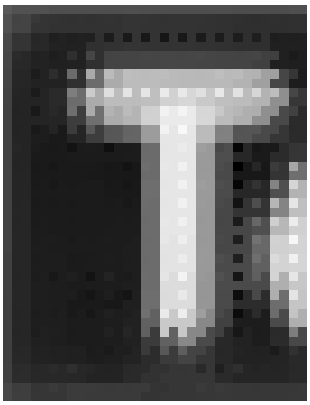
$$\forall i, j, \; i\,\%2 == 0 \&\& j\,\%2 == 0 \;\; zoom[i][j] = input[i/2][j/2]$$



- 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.

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

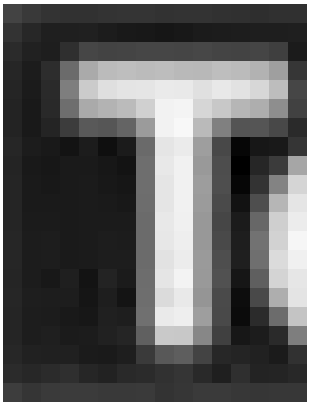
$$zoom[i][j] = input.copyArea(x/2-1, y/2-1, 3, 3).getHistogram().getBalancedLevel()$$



5. Zoom Out

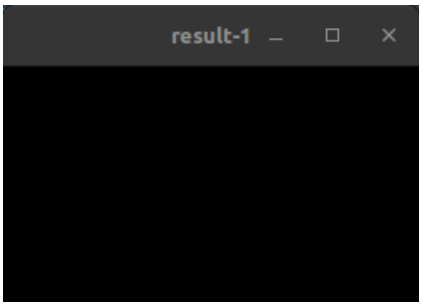
Para hacer la función de reducir (Zoom out) se sigue un procedimiento similar.

- Cargar la imagen de entrada $input$



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

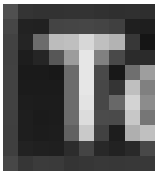
$$\forall i, j \;\; zoom[i][j] = 0$$



- 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

```
/**
@file Image.h
@brief Manejo de imÃ¡genes digitales en formato PGM blanco y negro
@author MP-DGIM - Grupo A
 **/

#ifndef _IMAGE_H_
#define _IMAGE_H_

#include <istream>
#include <fstream>
#include "Byte.h"
#include "Histogram.h"

/**
@brief A black and white image
 */
class Image {
public:
    static const int IMAGE_MAX_SIZE=200000; ///< Max number of bytes allowd for
    static const int IMAGE_DISK_OK=0; ///< Image read/write successful
    static const int IMAGE_ERROR_OPEN=1; ///< Error opening the file
    static const int IMAGE_ERROR_DATA=2; ///< Missing data in the file
    static const int IMAGE_ERROR_FORMAT=3; ///< Unknown image format
    static const int IMAGE_TOO_LARGE=4; ///< The image is too large and does not fit into memory


    /**
     * @brief It builds an empty, image
     */
    Image();
    /**
     * @brief It builds a fully black image with @a width columns and @a height rows
     * @param height number of rows
     * @param width number of columns
     */
    Image(int width, int height);
    /**
    @brief It gives the number of rows of the image
    @return number of rows
     */
    int height() const;
    /**
    @brief It gives the number of columns of the image
    @return The number of rows
     */
    int width() const;
    /**
     * @brief It assigns the value @a v to the position(x,y) of the image.It must check that
     * the values x and y are valid, otherwise, it does not do anythig.
     * @param x The column
     * @param y the row
     * @param v The new value
     */
    void setPixel(int x, int y, Byte v);
    /**
     * @brief It returns the value of the requested (x,y) position. It must check that
     * the values x and y are valid, otherwise, it returns a negative value. Please note that
     * the value returned is a int
     * @param x The column
     * @param y the row
     * @return The value of the pixel in [0-256] or -1 if there is an access error
     */
    int getPixel(int x, int y) const;
    /**
     * @brief It assigns the value @a v to the linear position i of the image. It must check that
     * the values i is valid, otherwise, it does not do anythig.
     * @param i The linear position
     * @param v The new value
     */
    void setPos(int i, Byte v);
    /**
     * @brief It returns the value of the requested linear position. It must check that
     * the value i is valid, otherwise, it returns a negative value. Please note that
     * the value returned is a int
     * @param i The linear position
     * @return The value of the pixel in [0-256] or -1 if there is an access error
     */
    int getPos(int i) const;
    /**
     * @brief  It sets all pixels of the image to the value given
     * @param b The value
     */
    void flatten(Byte b);
    /**
     * @brief It produces a mesh of vertical and horizontal stripes all along the
     * image. Every prim pixels it is set to 255 anad every sec pixels
     * it is set to 127
     * @param prim Gap between primary mesh
     * @param sec Gap between secondary mesh, Default value is 0
     */
    void mesh(int prim, int sec=0);

    /**
     * @brief It shows an image in an external window, ready for inspection. It uses
     * 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
200       /**
201        @brief Destroye the objetc and frees the memory
202         */
203        ~Image();
204        /**
205         * @brief Copy constructor. Creates a new Image, copying or duplicating an existing one
206         * @param from the image to duplicate
207         */
208        Image(const Image & from);
209
210        /**
211         * @brief Assignment operator
212         * @param rhs The right hand side of the assignment expresion
213         * @return The new instance
214         */
215        Image & operator=(const Image &rhs);
216         /**
217         * @brief It zooms the image in
218         * @return The zoomed image
219         */
220        Image zoomIn() const;
221        /**
222         * @brief Ti zooms the image out
223         * @return The zoomed image
224         */
225        Image zoomOut() const;
226    private:
227        Byte * _data; ///< Bytes of the image
228        int _height; ///< number of rows
229        int _width; ///< number of columms
230
231        /**
232         * @brief It frees the allocated memory
233         */
234        void clear();
235
236        /**
237         * @brief It copies an existing image into this one, resizing it accordingly
238         * @param rhs The source Image to copy from
239         */
240        void copy(const Image &rhs);
241
242        /**
243         * @brief it resizes the Image to a new number of rows and columns, clearing
244         * the previously existing data
245         * @param
246         * @param
247         */
248        void setSize(int , int );
249    };
250    #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

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 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. 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/f1
```

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/f1
```

```
lcv@numenor:Imaging4: valgrind --leak-check=full build/Debug/GNU-Linux/tests/TestFiles/f1
==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/f1
==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)
```