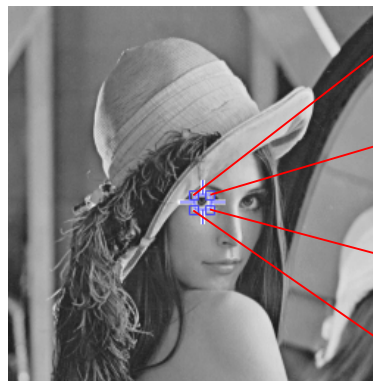




# Metodología de la Programación

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

*Copiar, pegar, binarizar*

*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>7</b>
<b>3. Binarización adaptativa de la imagen</b>	<b>7</b>
<b>4. Copiar un área de una imagen</b>	<b>8</b>
<b>5. Pegado selectivo y gradual en otra imagen</b>	<b>9</b>
5.1. Pegado selectiva . . . . .	9
5.2. Pegado gradual . . . . .	10
<b>6. Image</b>	<b>11</b>
<b>7. Argumentos de main</b>	<b>12</b>
<b>8. Práctica a entregar</b>	<b>14</b>
8.1. Primera parte . . . . .	14
8.1.1. Ejemplo de ejecución . . . . .	15
8.2. Segunda parte . . . . .	16
<b>9. TESTS DOCUMENTATION FOR PROJECT Imaging3b</b>	<b>18</b>
9.1. _01_Basics . . . . .	18
9.1.1. UnitByte_Constructor . . . . .	18
9.1.2. UnitByte_getValue . . . . .	18
9.1.3. UnitByte_setValue . . . . .	18
9.1.4. UnitByte_onBit . . . . .	18
9.1.5. UnitByte_offBit . . . . .	18
9.1.6. UnitByte_getBit . . . . .	18
9.1.7. UnitByte_to_string . . . . .	18
9.1.8. UnitByte_shiftRByte . . . . .	19
9.1.9. UnitByte_shiftLByte . . . . .	19
9.1.10. Image_Constructor . . . . .	19
9.1.11. Image_Width . . . . .	19
9.1.12. Image_Height . . . . .	19
9.1.13. Image_setPixel . . . . .	19
9.1.14. Image_getPixel . . . . .	19
9.1.15. Image_getPos . . . . .	19
9.1.16. Histogram_Constructor . . . . .	19
9.1.17. Histogram_Size . . . . .	20
9.1.18. Histogram_Clear . . . . .	20
9.1.19. Histogram_getLevel . . . . .	20
9.1.20. Histogram_setLevel . . . . .	20
9.1.21. Histogram_getMaxLevel . . . . .	20
9.1.22. Histogram_getAverageLevel . . . . .	20
9.1.23. Histogram_getBalancedLevel . . . . .	20
9.2. _02_Intermediate . . . . .	20



9.2.1. UnitByte_onByte . . . . .	20
9.2.2. UnitByte_offByte . . . . .	20
9.2.3. Image_flatten . . . . .	20
9.2.4. Image_getHistogram . . . . .	21
9.2.5. Image_depictsHistogram . . . . .	21
9.2.6. Image_threshold . . . . .	21
9.3. _03_Advanced . . . . .	21
9.3.1. UnitByte_encodeByte . . . . .	21
9.3.2. UnitByte_decodeByte . . . . .	21
9.3.3. UnitByte_decomposeByte . . . . .	21
9.3.4. Image_readFromFile . . . . .	21
9.3.5. Image_saveToFile . . . . .	22
9.3.6. Image_extractObjects . . . . .	22
9.3.7. Image_copy . . . . .	22
9.3.8. Image_paste . . . . .	22
9.3.9. INTEGRATION_ImageP3b . . . . .	22
9.4. Tests run . . . . .	22





## 2. Objetivos

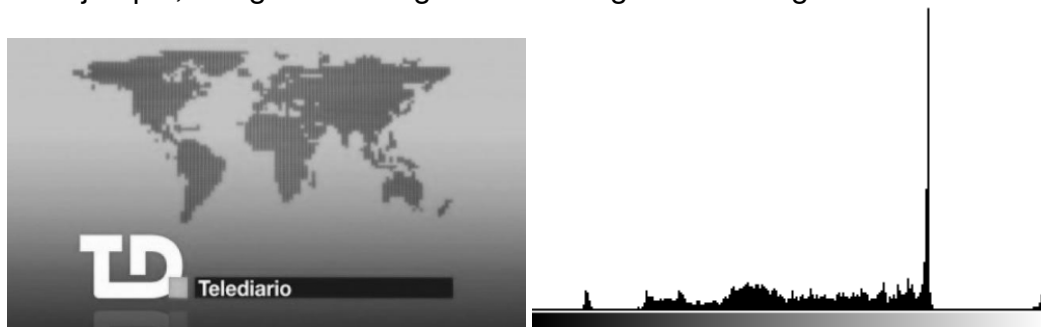
El desarrollo de esta práctica pretende servir a los siguientes objetivos:

- 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.
- Empleo de argumentos de `main` para modificar el funcionamiento de un programa. Se puede consultar más información sobre el uso de parámetros del `main` en la sección 7.

## 3. Binarización adaptativa de la imagen

La función de binarizar una imagen a partir de un nivel  $t$  permite asignar 255 (blanco) cualquier nivel estrictamente mayor que  $t$  y 0 (negro) al resto, lo que produce una imagen binarizada, solo con dos niveles 0 y 255.

Por ejemplo, la siguiente imagen tiene el siguiente histograma



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



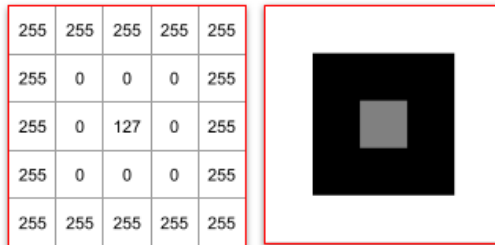
Una imagen también se puede binarizar de forma adaptativa (en función de la propia distribución de tonos de la imagen). Para ello, se utiliza como  $t$  aquél nivel que deja por debajo, o igual a él, al menos a la mitad de la imagen. En este caso, el umbral elegido sería  $t = 145$  y produciría este resultado. Es más elocuente su histograma, respecto del anterior.



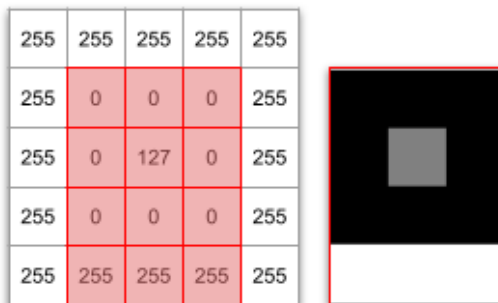
## 4. Copiar un área de una imagen

Dada una imagen que se quiere copiar, se puede seleccionar completa o tan solo una región parcial de la misma. Veamos diferentes situaciones que se pueden producir.

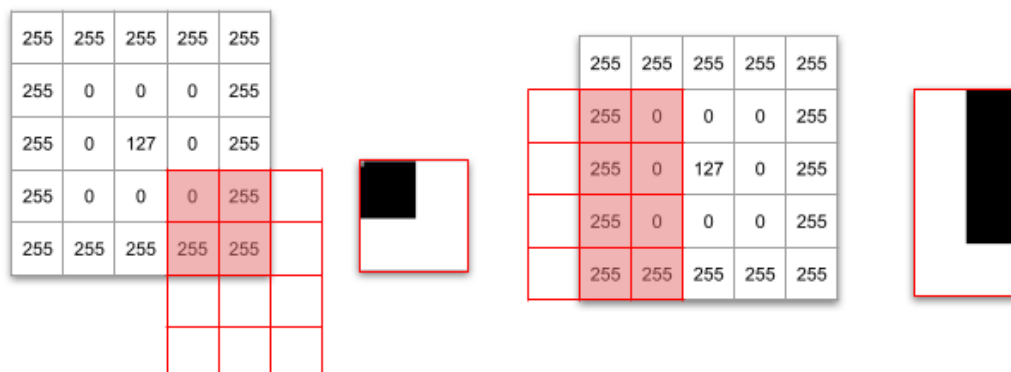
Por ejemplo, dada la siguiente imagen



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



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





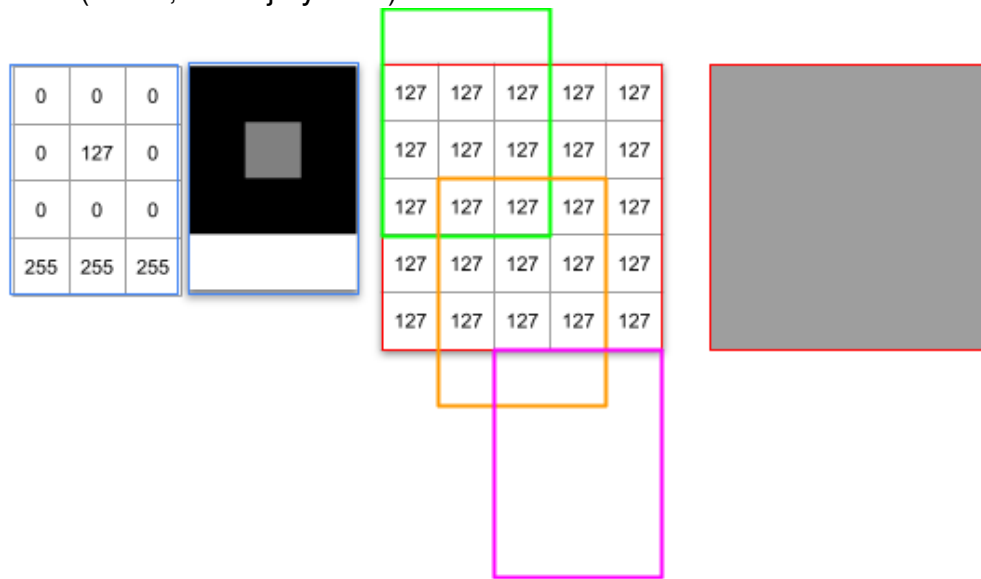
## 5. Pegado selectivo y gradual en otra imagen

Para cualquier operación de pegar vamos a manejar tres imágenes: una imagen de entrada (*input*) sobre la que se va a poder copiar otra (*copyfrom*<sup>1</sup>), el resultado se lleva a cabo sobre una nueva imagen (*output*) dejando las dos primeras sin modificar.

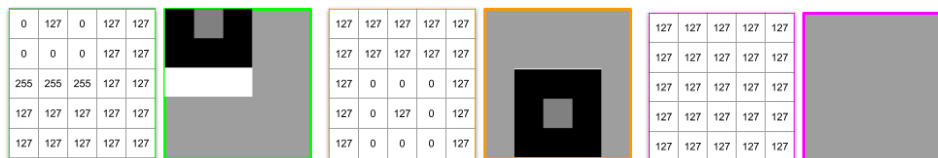
A la hora de pegar, de forma similar a como se hacía para la copia, es importante tener en cuenta las coordenadas en *input* donde se va a pegar la *copyfrom*. Como ya sabemos, la coordenada superior izquierda es (0,0). Dependiendo de las coordenadas (x,y) la copia puede ser total (la imagen *copyfrom* entra completamente) o parcial (solo se trasladan los píxeles de *copyfrom* que entran dentro de la imagen *input*). Esto es, las dimensiones de salida, coinciden con las dimensiones de la imagen *input*.

El copia más básica consiste en que los píxeles de *copyfrom* sustituyen a los píxeles de la *input*, teniendo en cuenta lo comentado anteriormente respecto de la posibilidad de indicar posiciones fuera de la imagen.

Por ejemplo si quisiésemos copiar la imagen pequeña (*copyfrom*, enmarcada en color azul) en la *input* imagen (en rojo) en tres posiciones distintas (verde, naranja y rosa)



Las imágenes de salida resultante de cada una de las copias ellas, serían:



### 5.1. Pegado selectiva

En el pegado selectivo interviene un umbral  $k$ , un valor de tono respecto a la imagen *copyfrom*. Los píxeles de *copyfrom* cuyo valor está por encima de  $k$  se pegan en la salida en la región correspondiente de la imagen *input*, mientras que los que quedan por debajo,  $[0, k]$  más oscuros de

<sup>1</sup>Esta imagen puede ser la selección parcial de una imagen original.

*copyfrom*, se ignoran. El efecto es que solo se trasladan píxeles de colores brillantes desapareciendo los oscuros (en ocasiones el fondo) através de los cuales se pueden ver los píxeles de la imagen *input* original, dando un efecto de transparencia.

## 5.2. Pegado gradual

En esta modalidad, interviene un nuevo parámetro, el porcentaje  $\alpha \in [0, 100]$  que va a indicar el grado de visibilidad de la imagen *copyfrom* en la imagen de salida. Para montar la imagen de salida, los píxeles de la imagen *copyfrom* se mezclan con la región de copia de la *input* en un porcentaje de mezcla. De manera que si  $b_1$  es el byte de la imagen *input* y  $b_2$  es el byte de la imagen *copyfrom*, entonces el byte que realmente se pega en la imagen de salida es

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

En los pegados anteriores, el valor de porcentaje utilizado es 100, cuando el valor es más reducido el efecto es que la imagen *copyfrom* se vuelve translúcida en la salida.



## 6. Image

```
1  /**
2  @file Image.h
3  @brief First class 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 /// Additional methods
11
12 /**
13  * It returns a binarization of the original image. All pixels strictly greater than the value @a t
14  * are set to 11111111 and the others to 00000000. Query method.
15  * @param t The threshold. This value must be within [0,255]. In the case
16  * that the threshold is not within these bounds, an automatic threshold is chosen,
17  * that is, the first level that leaves, at least the half of points less than
18  * or equal to it. Input parameter, default value is set to -1.
19  * @return A copy of the original image
20  */
21 Image threshold(int t = -1);
22
23 /**
24  * @brief It returns a subimage of the original image given by the parameters.
25  * If the parameters exceed the dimensions of the original image, then
26  * the cut image should not exceed those limits. Query method.
27  * @param x x-coordinate of the top left corner.
28  * @param y y-coordinate of the top left corner
29  * @param w width of the subimage
30  * @param h height of the subimage. x,y,w, are input parameters.
31  * @return The subimage cut
32  */
33 Image copyArea(int x, int y, int w, int h);
34
35 /**
36  * @brief It pastes a secondary image into the original image, at the given position,
37  * into the same original image, which results modified.
38  * This operation should maintain the limits of the original image
39  * @param x x-coordinate of the top left corner. input param
40  * @param y y-coordinate of the top left corner. input param
41  * @param from The second image. input param
42  * @param toneup value. input param
43  * @param merge value for the combination of the 2 images. input param
44  */
45 void pasteArea(int x, int y, Image from, int toneup=-1, int merge=100);
```

## 7. Argumentos de `main`

Hasta el momento, los programas que hemos implementado tenían siempre la misma funcionalidad, y para poder cambiar su comportamiento teníamos que reescribir el código y volver a compilar, o interactuar con él usando el teclado durante su ejecución. No obstante, la interacción con el programa a través del teclado puede no ser la forma más adecuada de cambiar su comportamiento en determinados contextos. Por ejemplo, el comando `ls`, de la terminal de Linux que ya debes conocer (el cual es un programa escrito en C), nos permite mostrar el contenido del directorio que se pasa como argumento. Si no se pasa ningún argumento, se muestra el contenido del directorio actual:

```
fluque1995@fluque1995-Aspire-VX5-591G:~/Netbeans$ ls
DataTable      Gtest          Scripts        Shopping2_S
DataTable.zip  MPTTest        Shopping1_S    Shopping2_S_files
googletest-master MPTTest_V2.zip Shopping2       Shopping2_S.zip
fluque1995@fluque1995-Aspire-VX5-591G:~/Netbeans$ ls Shopping1_S
build      dist      Makefile  __nTest    src
data       doc       nbproject __nTestname tests
dirInfo.md include  __nReport scripts    zip
```

Figura 2: Ejemplo de ejecución del program `ls` con y sin parámetros

Aquí tenemos un ejemplo claro de un programa cuyo comportamiento depende de los parámetros que recibe. Como podrás imaginar, preguntar al usuario en el momento de la ejecución por el directorio del que se quiere mostrar el contenido puede ser bastante incómodo. Además, el hecho de poder especificarlo de antemano permite que el programa se pueda ejecutar sin necesidad de que una persona interactúe con el programa en el momento de la ejecución.

Los parámetros o argumentos de la función `main` son, precisamente, las herramientas de las que disponemos para cambiar el comportamiento de nuestro programa. Funcionan de la misma forma que los argumentos de cualquier otra función. C++ acepta dos tipos de declaraciones de la función `main`, con y sin argumentos:

```
int main(){
    // Función main sin argumentos
}

int main(int argc, char **argv){
    // Función main con argumentos
}
```

Como se puede observar, cuando trabajamos con la declaración del `main` con argumento, se reciben dos argumentos distintos:

- `int argc`: Entero que indica el número de argumentos que se han recibido
- `char **argv`: Array de cadenas de caracteres con cada uno de los argumentos recibidos

A continuación se muestra un ejemplo de programa que recorre todos los argumentos que recibe y los imprime por pantalla:

```
#include <iostream>

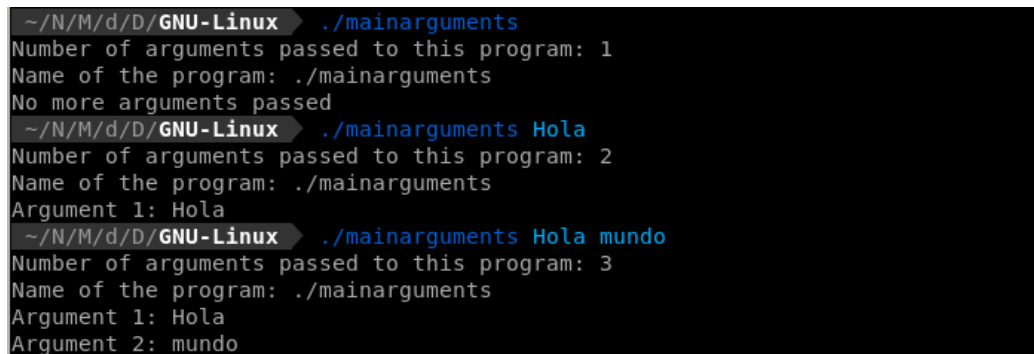
using namespace std;
int main(int argc, char** argv) {

    /* Usage of main arguments. Main receives a series of arguments when called from the terminal:
     * - argc: Number of arguments passed
     * - argv: Array of strings representing the arguments
     */
    cout << "Number_of_arguments_passed_to_this_program:_\n" << argc << endl;

    /* argv[0] is always the name of the program
     */
    cout << "Name_of_the_program:_\n" << argv[0] << endl;

    /* The rest of the arguments (if passed), come ordered. We use argc as stopping point (the size of argv)
     * We start from 1 since argv[0] is the name of the program
     */
    if (argc == 1){
        cout << "No_more_arguments_passed" << endl;
    }
    for (int i = 1; i < argc; i++){
        cout << "Argument_\n" << i << ":\n" << argv[i] << endl;
    }
    return 0;
}
```

Y la ejecución del programa depende ahora de la llamada que hagamos desde la terminal:



```
~/N/M/d/D/GNU-Linux ➤ ./mainarguments
Number of arguments passed to this program: 1
Name of the program: ./mainarguments
No more arguments passed
~/N/M/d/D/GNU-Linux ➤ ./mainarguments Hola
Number of arguments passed to this program: 2
Name of the program: ./mainarguments
Argument 1: Hola
~/N/M/d/D/GNU-Linux ➤ ./mainarguments Hola mundo
Number of arguments passed to this program: 3
Name of the program: ./mainarguments
Argument 1: Hola
Argument 2: mundo
```

Figura 3: Ejemplos de ejecución del programa anterior con distintos argumentos

De esta manera, podremos modificar el comportamiento de nuestro programa sin necesidad de interactuar con él en el momento de la ejecución, o sin tener que recompilarlo.

En nuestro proyecto, será de gran utilidad poder emplear los argumentos de main. Por ejemplo, en la práctica Imaging2 utilizamos una imagen de entrada a partir de la que obteníamos, varias imágenes derivadas como: su histograma y varios segmentos de imagen hallados. Para realizar las mismas operaciones partiendo de otra imagen era necesario modificar el código del programa y volver a compilarlo. Ahora, utilizando los parámetros de main, podremos especificar el nombre de la imagen de partida sin tener que cambiar nada. En el archivo `main.cpp` de la práctica aparece información sobre los distintos argumentos que habrá que procesar, y que condicionarán el comportamiento de nuestro programa (el archivo de entrada, el archivo que copiar, el archivo de salida) con una sintaxis determinada.

Los detalles sobre cómo procesar argumentos dados, en cualquier orden, de los que parte son obligatorios y algunos optativos puede encontrarse en un Guión especialmente elaborado sobre argumentos de main. (Ver web de la asignatura.)



## 8. Práctica a entregar

### 8.1. Primera parte

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

- Se deben implementar los métodos indicados en el fichero `Image.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 *salida* dentro de la carpeta `data/` llamada `new.pgm`.

### 8.1.1. 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

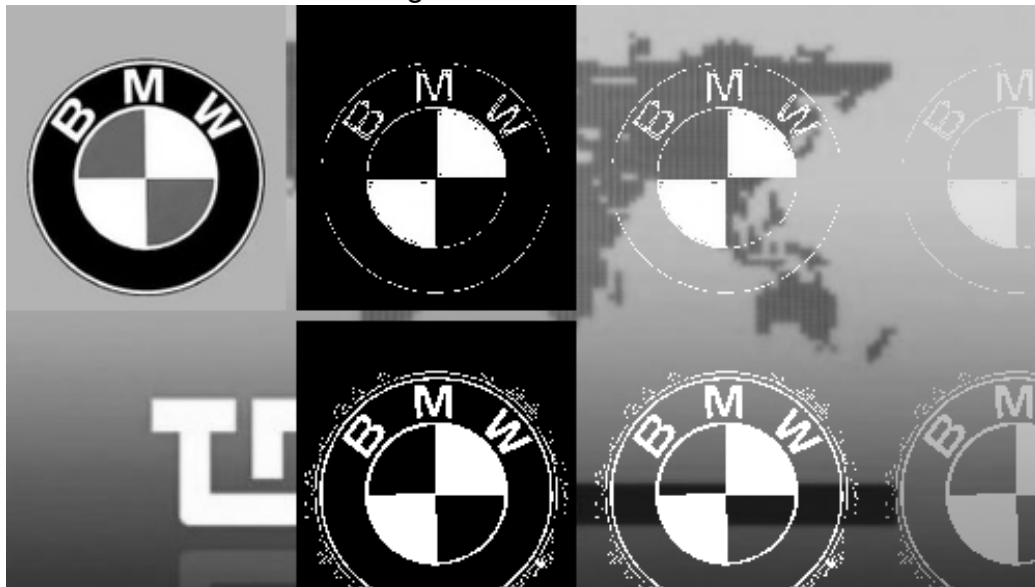
[im_output] 500x282 805563168

```

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



el resultado debería ser el siguiente



## 8.2. Segunda parte

Se van a utilizar los argumentos del main para modificar el comportamiento de nuestro programa. En los comentarios del fichero `main.cpp` se describen con detalle los distintos argumentos que puede recibir nuestro programa. Los argumentos que aparecen entre corchetes son argumentos opcionales:

```
imaging3 -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.

Para poder ejecutar el programa con una de las órdenes anteriores es necesario utilizar la terminal o modificar la orden de ejecución de las propiedades del proyecto (**Run - Run Command**). Por ejemplo, Si queremos utilizar la imagen de entrada `telediario.pgm`, copiar la imagen `kfc.pgm` (*copyfrom*) y dejar la salida en `p3b.pgm`, tendríamos que especificar lo siguiente:

```
"${OUTPUT_PATH}" -i ./data/telediario.pgm -c ./data/kfc.pgm -o ./data/p3b.pgm
```

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

```
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:Imaging3: dist/Debug/GNU-Linux/imaging3 -i data/telediario.pgm
```

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

```
[im_input] 500x282 4105296496
```

```
[im_output] 500x282 4105296496
```

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



## 9. TESTS DOCUMENTATION FOR PROJECT Imaging3b

### 9.1. \_01\_Basics

#### 9.1.1. UnitByte\_Constructor

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

#### 9.1.2. UnitByte\_getValue

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

#### 9.1.3. 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

#### 9.1.4. 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

#### 9.1.5. 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

#### 9.1.6. UnitByte\_getBit

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

#### 9.1.7. UnitByte\_to\_string

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



#### 9.1.8. **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

#### 9.1.9. **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

#### 9.1.10. **Image\_Constructor**

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

#### 9.1.11. **Image\_Width**

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

#### 9.1.12. **Image\_Height**

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

#### 9.1.13. **Image\_setPixel**

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

#### 9.1.14. **Image\_getPixel**

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

#### 9.1.15. **Image\_getPos**

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

#### 9.1.16. **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



#### **9.1.17. Histogram\_Size**

1. Any histogram must have a capacity for 256 values

#### **9.1.18. 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

#### **9.1.19. Histogram\_getLevel**

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

#### **9.1.20. Histogram\_setLevel**

1. A crescent triangular histogram is wrong

#### **9.1.21. Histogram\_getMaxLevel**

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

#### **9.1.22. Histogram\_getAverageLevel**

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

#### **9.1.23. Histogram\_getBalancedLevel**

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

### **9.2. \_02\_Intermediate**

#### **9.2.1. UnitByte\_onByte**

1. Activating a Byte gives 255

#### **9.2.2. UnitByte\_offByte**

1. Deactivating a Byte gives 0

#### **9.2.3. Image\_flatten**

1. is wrong
2. is wrong



#### 9.2.4. `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

#### 9.2.5. `Image_depictsHistogram`

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

#### 9.2.6. `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

### 9.3. `_03_Advanced`

#### 9.3.1. `UnitByte_encodeByte`

1. Activating bits 0,1 and 7 gives 131

#### 9.3.2. `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

#### 9.3.3. `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

#### 9.3.4. `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

#### 9.3.5. `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

#### 9.3.6. `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

#### 9.3.7. `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

#### 9.3.8. `Image_paste`

1. Checkers cannot be built by pastin each quadrant

#### 9.3.9. `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

### 9.4. 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.
```