
PROYECTO PGM

Juan Manuel Rodríguez Gómez

Doble Grado en Ingeniería Informática y Matemáticas

Fundamentos de la Programación

Curso 2019 – 2020

1. Diseño General

Este proyecto consiste en crear cuatro programas en el lenguaje de programación C++ para trabajar con imágenes. Para almacenar dichas imágenes utilizaremos ficheros de texto con extensión .pgm (PGM es el acrónimo de *Portable Gray Map file format*). Cada programa hace lo siguiente:

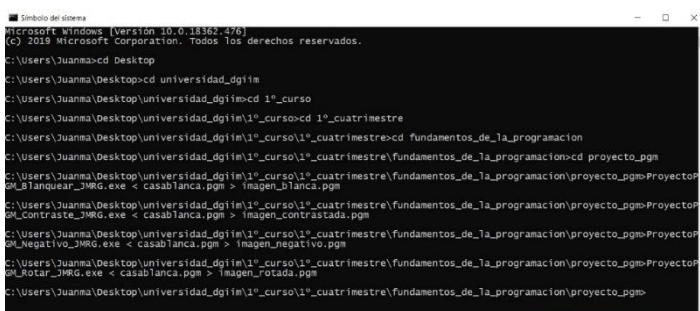
- **ProyectoPGM_Blanquear_JMRG.exe:** Convierte la imagen en otra completamente blanca.
- **ProyectoPGM_Contraste_JMRG.exe:** Contrasta al máximo la imagen.
- **ProyectoPGM_Negativo_JMRG.exe:** Muestra el negativo de la imagen.
- **ProyectoPGM_Rotar_JMRG.exe:** Rota la imagen 90° a la derecha.

Cada programa funcionará mediante cuatro funciones:

- 1) Función principal (que contiene a las tres funciones siguientes).
- 2) Función de lectura de datos de la imagen.
- 3) Función que realiza las acciones necesarias sobre la imagen.
- 4) Función que devuelve los datos de la imagen resultante.

donde las funciones de lectura y devolución de datos serán iguales para los cuatro programas.

Para ejecutar uno de los programas anteriores y que este devuelva la imagen resultante, tenemos que hacerlo desde la terminal de nuestro sistema operativo (en mi caso, Windows 10). Por otro lado, para visualizar la imagen, usaremos una aplicación gratuita para Windows llamada *Brennig's*.



```
Microsoft Windows [Version 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Juanma>cd Desktop
C:\Users\Juanma\Desktop>cd universidad_dgim
C:\Users\Juanma\Desktop\universidad_dgim>cd 1º_curso
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso>cd 1º_cuatrimestre
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre>cd fundamentos_de_la_programacion
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre\fundamentos_de_la_programacion>cd proyecto_pgm
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre\fundamentos_de_la_programacion\proyecto_pgm>ProyectoPGM_Blanquear_JMRG.exe < casablanca.pgm > imagen_blanca.pgm
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre\fundamentos_de_la_programacion\proyecto_pgm>ProyectoPGM_Contraste_JMRG.exe < casablanca.pgm > imagen_contrastada.pgm
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre\fundamentos_de_la_programacion\proyecto_pgm>ProyectoPGM_Negativo_JMRG.exe < casablanca.pgm > imagen_negativo.pgm
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre\fundamentos_de_la_programacion\proyecto_pgm>ProyectoPGM_Rotar_JMRG.exe < casablanca.pgm > imagen_rotada.pgm
C:\Users\Juanma\Desktop\universidad_dgim\1º_curso\1º_cuatrimestre\fundamentos_de_la_programacion\proyecto_pgm>
```

Imagen 1.1: Programas ejecutados desde la terminal de Windows 10.

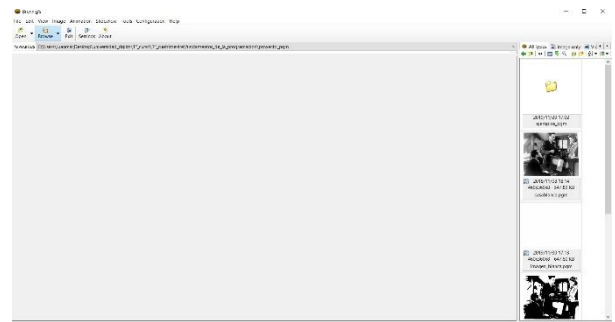


Imagen 1.2: *Brennig's*.

Vamos a usar como ejemplo la imagen *casablanca.pgm*:



Imagen 1.3: *casablanca.pgm*

Tras ejecutar los programas, las imágenes resultantes serían las siguientes (el resultado Del programa 1 no se muestra ya que simplemente es una imagen en blanco):



Imagen 1.4: *imagen_contrastada.pgm*



Imagen 1.5: *imagen_negativo.pgm*



Imagen 1.6: *imagen_rotada.pgm*

2. Programación

A la hora de trabajar con imágenes, hace falta saber que una imagen es en realidad una matriz de n filas por m columnas, donde cada elemento de la matriz es un píxel.

Ahora pasamos a analizar las funciones de los programas:

1) Función principal:

```
int main(){
    Imagen imagen_inicial;
    Imagen imagen_final;
    string cadena_magica;
    int maximo_valor_gris;

    /*
     Algoritmo:
     Se leen los datos de entrada de la imagen inicial.
     La imagen final se iguala a la imagen inicial.
     La imagen final pasa a ser una imagen completamente blanca.
     Se muestra la imagen final.
    */

    LeerImagen(cadena_magica, maximo_valor_gris, imagen_inicial);

    imagen_final = imagen_inicial;
    ImagenBlanca(imagen_final);
    MostrarImagen(cadena_magica, maximo_valor_gris, imagen_final);
}
```

Imagen 2.1: Función principal del programa *ProyectoPGM_Blanquear_JMRG.exe*

Lo único que varía en la función principal de cada programa es la función que realiza las acciones necesarias sobre la imagen (siendo en este ejemplo *ImagenBlanca*).

2) Funciones de lectura y devolución de datos:

Recordemos que son iguales para los cuatro programas.

```
void LeerImagen(string &cadena_magica, int &maximo_valor_gris, Imagen &imagen_inicial){
    /*
     Algoritmo:
     Se lee la cadena magica del archivo de texto .pgm (siempre es P2).
     Se leen el numero de columnas y filas de la imagen inicial.
     Se lee el valor maximo de gris (siempre es 255).
     Se lee cada pixel de la imagen inicial.
    */

    cin >> cadena_magica;

    cin >> imagen_inicial.columnas;
    cin >> imagen_inicial.filas;

    cin >> maximo_valor_gris;

    for(int i = 0; i < imagen_inicial.filas; i++)
        for(int j = 0; j < imagen_inicial.columnas; j++)
            cin >> imagen_inicial.matriz_pixels[i][j];
}
```

Imagen 2.2: Función de lectura de datos

```
void MostrarImagen(string &cadena_magica, int &maximo_valor_gris, Imagen &imagen_final){
    /*
     Algoritmo:
     Se muestra la cadena magica del archivo de texto .pgm (siempre es P2).
     Se muestran el numero de columnas y filas de la imagen final.
     Se muestra el valor maximo de gris (siempre es 255).
     Se muestra cada pixel de la imagen final.
    */

    cout << cadena_magica << endl;

    cout << imagen_final.columnas;
    cout << " " << imagen_final.filas << endl;

    cout << maximo_valor_gris << endl;

    for(int i = 0; i < imagen_final.filas; i++){
        for(int j = 0; j < imagen_final.columnas; j++){
            cout << imagen_final.matriz_pixels[i][j] << " ";
        }
        cout << endl;
    }
}
```

Imagen 2.3: Función de devolución de datos

3) Funciones que realizan las acciones necesarias sobre la imagen:

➤ ProyectoPGM_Blanquear_JMRG.exe:

Para convertir la imagen en otra completamente blanca, el programa asigna a cada píxel el valor 255.

```
void ImagenBlanca(Imagen &imagen_final){  
    /*  
    Algoritmo:  
    Cada pixel de la imagen final pasa a tomar el valor 255 (es decir,  
    pasa a ser de color blanco).  
    */  
    for(int i = 0; i < imagen_final.filas; i++)  
        for(int j = 0; j < imagen_final.columnas; j++)  
            imagen_final.matriz_pixels[i][j] = 255;  
}
```

Imagen 2.4: Función *ImagenBlanca* en C++.

➤ ProyectoPGM_Contraste_JMRG.exe:

Para contrastar al máximo la imagen, el programa cambia los valores de los píxeles menores de 125 a 0 y los mayores o iguales los cambia a 255.

```
void ImagenContrastada(Imagen &imagen_final){  
    /*  
    Algoritmo:  
    Si un pixel de la imagen final tiene un valor menor a 125:  
    Pasa a tomar el valor 0 (es decir, pasa a ser de color negro).  
    Si no:  
    Pasa a tomar el valor 255 (es decir, pasa a ser de color blanco).  
    */  
    for(int i = 0; i < imagen_final.filas; i++)  
        for(int j = 0; j < imagen_final.columnas; j++)  
            if(imagen_final.matriz_pixels[i][j] < 125)  
                imagen_final.matriz_pixels[i][j] = 0;  
            else  
                imagen_final.matriz_pixels[i][j] = 255;  
}
```

Imagen 2.5: Función *ImagenContrastada* en C++.

➤ ProyectoPGM_Negativo_JMRG.exe:

Para mostrar el negativo de la imagen, el programa cambia un píxel con el valor x a $255 - x$ (en los casos extremos, $x = 0$ y $x = 255$, el programa cambia, respectivamente, negro por blanco y blanco por negro).

```
void ImagenNegativo(Imagen &imagen_final){  
    /*  
    Algoritmo:  
    Cada pixel de la imagen final que tiene un valor "x" pasa  
    a tener un valor "255 - x" (en los casos extremos,  $x = 0$  y  
     $x = 255$ , se cambia, respectivamente, blanco por negro y negro por blanco).  
    */  
    for(int i = 0; i < imagen_final.filas; i++)  
        for(int j = 0; j < imagen_final.columnas; j++)  
            imagen_final.matriz_pixels[i][j] = 255 - imagen_final.matriz_pixels[i][j];  
}
```

Imagen 2.6: Función *ImagenNegativo* en C++.

➤ ProyectoPGM_Rotar_JMRG.exe:

Para rotar la imagen 90° a la derecha, el programa traspone la primera fila de la imagen y la coloca en la última columna, luego traspone la segunda fila y la coloca en la penúltima columna y así sucesivamente hasta completar todas las filas de la imagen.

```
void ImagenRotada90Derecha(Imagen &imagen_final){  
    Imagen imagen_rotada;  
    /*  
    Algoritmo:  
    La primera fila de pixeles de la imagen se traspone y pasa  
    a ser la ultima columna de la matriz de pixeles de la imagen.  
    La segunda fila de pixeles de la imagen se traspone y pasa  
    a ser la penultima columna de la matriz de pixeles de la imagen.  
    La tercera fila de pixeles de la imagen se traspone y pasa  
    a ser la antepenultima columna de la matriz de pixeles de la imagen.  
    Así sucesivamente hasta la ultima fila de pixeles de la imagen (esta  
    ultima fila tambien esta incluida).  
    */  
    imagen_rotada.filas = imagen_final.columnas;  
    imagen_rotada.columnas = imagen_final.filas;  
    for(int i = 0; i < imagen_rotada.filas; i++)  
        for(int j = 0; j < imagen_rotada.columnas; j++)  
            imagen_rotada.matriz_pixels[i][j] = imagen_final.matriz_pixels[imagen_rotada.columnas - j - 1][i];  
    imagen_final = imagen_rotada;  
}
```

Imagen 2.7: Función *ImagenRotada90Derecha* en C++.