PROYECTO PGM

INFORMACIÓN GENERAL

1. AUTORES DEL PROYECTO

- Marcos Rico Guerra
- Pablo Olivares Martínez

2. ARCHIVOS REALIZADOS:

Hemos realizado 5 ficheros, siendo 4 los que se pedían en el proyecto, y uno extra para almacenar los módulos para leer y devolver imágenes pgm:

- ***Entrada_y_Salida.cpp*** → Fichero que contiene las funciones y estructuras para leer y devolver una imagen pgm. Este fichero lo hemos incluido en el resto de ficheros cpp.
- ***Contraste.cpp*** → Fichero que maximiza el contraste de una imagen pgm.
- ***Girar.cpp*** → Fichero que rota 90º a la derecha una imagen pgm.
- ***Blanquear.cpp*** → Fichero que blanquea al máximo una imagen pgm.
- ***Negativo.cpp*** → Fichero con el que se obtiene el negativo de una imagen.

3. REPARTO DEL TRABAJO

El trabajo ha sido repartido entre los dos programadores, de forma que cada fichero fue realizado por un programador distinto, aunque durante la realización de ellos, nos ayudábamos mutuamente para lograr una mayor eficiencia de los códigos.

- **Marcos Rico Guerra:** realizó los ficheros de Entrada y Salida, además de realizar los ficheros Contraste.cpp y Rotar.cpp.

- **Pablo Olivares Martínez:** se encargó de construir los ficheros Blanquear.cpp y Negativo.cpp, además de comentar los ficheros y módulos.

TRABAJO REALIZADO POR CADA PROGRAMADOR

MARCOS RICO GUERRA:

La parte del trabajo que yo he realizado ha consistido en crear los ficheros Entrada_Y_Salida.cpp, Girar.cpp y Contraste.cpp.

Para el primero, he realizado un *struct* llamado Imagen que incluye los datos de una imagen PGM (una cadena mágica, el número de filas y columnas, el brillo máximo permitido (en nuestro caso 255) y la matriz que incluía los datos del brillo de cada pixel en la imagen, tomando valores de 0 a 255).

Además, un módulo para leer todos estos datos y declararlos dentro de una variable de tipo Imagen, al que he llamado ***EntradaImagen***, y otro módulo que devolvía todos los datos de una variable tipo Imagen, llamado ***SalidaImagen***.

A continuación, para el fichero Girar.cpp, he construido un código en el que además de incluir el fichero Entrada_Y_Salida.cpp, he declarado una nueva función que rota 90 grados a la derecha todos los elementos de la matriz de una variable tipo Imagen.
A este módulo lo he llamado ***Girar***.

Por último, para el fichero Contraste.cpp, además de volver a incluir el fichero Entrada_Y_Salida.cpp, también he realizado una función que varía los elementos de una matriz de una variable tipo Imagen, de forma que todos los elementos menores que 125 pasan a ser 0, y todos los elementos mayores o iguales a 125 pasan a ser 255.
A este módulo lo he llamado ***Contraste***.

Cabe destacar que todos los módulos realizados han sido de tipo **void** e incluían siempre como parámetro una variable de tipo Imagen.

PABLO OLIVARES MARTÍNEZ:

La parte del trabajo que yo he realizado ha consistido en crear los ficheros, Blanquear.cpp y Negativo.cpp.

Para ambos ficheros se ha empleado una estructura muy similar. Primero, declaro la estructura la cual servirá para almacenar los datos de la imagen introducida con el módulo ***EntradaImagen*** y almacena también la imagen final resultante de la modificación producida por el programa, la cual es devuelta mediante el módulo ***SalidaImagen***.

Para el fichero Blanquear.cpp se ha creado un módulo el cual vuelve todos los elementos de la matriz, es decir, todos los píxeles de la imagen, en blanco. Para ellos cambia los valores de la matriz por 255, el cual es el valor del blanco. Este módulo ha sido denominado ***Blanquear***.

Por otro lado, el fichero Negativo.cpp se encarga de hallar el negativo de la imagen, es decir, cada pixel toma el valor de su contrario en el rango de 0 – 255. Para ello se le resta a 255 el valor del pixel de la imagen, siendo el resultado de la operación el negativo. El módulo empleado ha sido llamado ***Negativo***.

Finalmente he incluido los comentarios del programa y normalizado el formato de los cuatro programas con el fin de tener una estructura lo más similar posible en todos ellos, facilitando la legibilidad y la reutilización del código.