Diseño General

El proyecto de imágenes en formato PGM ha seguido la siguiente estructura general:

- Alias de `short int` como `color`.
- Tamaño de filas: 650 pixeles (excepto Rotar).
- Tamaño de columnas: 650 pixeles (excepto Rotar).
- Constantes globales con los valores del color blanco y negro
  - `const int BLANCO = 255;`
  - `const int NEGRO = 0;`
- Tipo de dato Imagen que contiene:
  - 1 string: `cadena_magica.`
  - 3 enteros: filas, columnas, `max_color.`
  - Matriz de tipo `color` de tamaño 650x650 (excepto Rotar).
- Declaración y definición de funciones a usar: entrada, salida y las correspondientes a cada programa individual.
- Programa principal.

También disponemos de una serie de **funciones** que se usan en todos los programas:

- Función de entrada: función que crea la imagen de entrada pasándola por referencia y leyendo sus datos uno a uno, pudiendo usarse la redirección. Esta función está diseñada para leer archivos en formato pgm. Esto es que primero lee una "cadena mágica", luego las dimensiones de la imagen (columnas x filas), el número usado para representar el color máximo y finalmente la matriz que representa los pixeles de la imagen.

```
void Entrada(Imagen &imagen){

      cin >> imagen.cadena_magica;
      cin >> imagen.columnas >> imagen.filas;
      cin >> imagen.max_color;

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

- Función de salida: función que devuelve los datos de una imagen pasada como parámetro. El criterio de espacios en blanco y saltos de línea se basa en el formato pgm, permitiendo obtener un archivo .pgm al usar redirección. Los dos primeros saltos de línea diferencian la "cadena mágica" del tamaño de la imagen y del número máximo usado para representar el color. Tras esto se muestra cada elemento de la matriz con un espacio en blanco entre elementos y una nueva línea entre filas.

```
void Salida(Imagen imagen){
      cout << imagen.cadena_magica << endl;
      cout << imagen.columnas << " " << imagen.filas << endl;
      cout << imagen.max_color;

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

Programa principal *`main`*: en el programa principal de cada programa creado tenemos la declaración de la imagen a modificar, la lectura de sus datos, su modificación con la función correspondiente y la salida de los datos de la nueva imagen.

En el caso del programa Rotar, el programa principal main contiene además la declaración de una imagen de salida en la que se guardarán los datos de la nueva imagen.

Reparto del trabajo

- Blanquear y Rotar: Juntos
- Negativo: Santiago Hernández
- Contraste: Salvador Romero

Programador: Salvador y Santiago

BLANQUEO

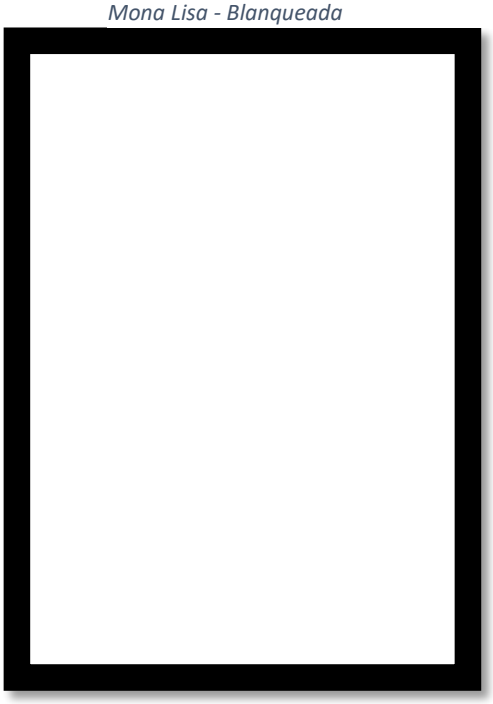
Función de blanqueo: función Imagen toma como parámetro de entrada un tipo de dato `Imagen,` recorre cada componente de la matriz de la imagen de entrada y le asigna el valor 255, correspondiente al blanco.

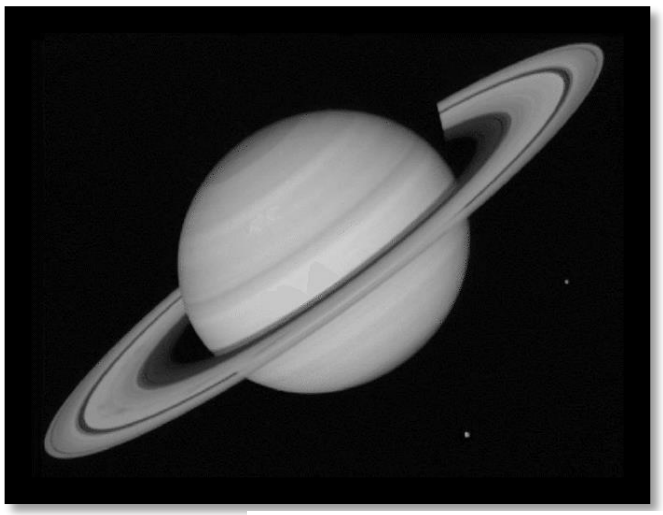
```
void Blanquear (Imagen &imagen){
      for (int i=0; i < imagen.filas; i++)
            for(int j=0; j < imagen.columnas; j++)
                  imagen.matriz[i][j] = BLANCO;
}
```

Ejemplos.

Mona Lisa

Mona Lisa - Blanqueada





Saturno

Saturno - Blanqueado

(Las fotos tienen marcos para poder apreciar la nueva imagen blanqueada).

ROTAR

Función de rotar 90º a la derecha: función que devuelve un dato del tipo `Imagen`, tomando como parámetro un dato del mismo tipo. Siendo las dimensiones de la matriz asociada al primer dato m x n mientras que las de la matriz asociada al segundo n x m.

El algoritmo de giro de imagen consiste en asignar a cada elemento (i,j) de la matriz de salida los valores del elemento (imagen.filas - j - 1, i) de la imagen de entrada, donde imagen.filas = m.

Se toman los elementos de cada columna de la matriz de entrada de "abajo arriba" y se colocan en el orden usual en la matriz de salida.

De esta manera logramos asignar los pixeles para que la imagen resultante esté girada 90º a la derecha.

En este caso la dimensión máxima de la matriz de un dato tipo Imagen es de 400x400.

```
Imagen RotarDerecha(Imagen imagen){
      Imagen girada;
      girada.cadena_magica = imagen.cadena_magica;
      girada.filas = imagen.columnas;
      girada.columnas = imagen.filas;
      girada.max_color = imagen.max_color;

      for (int i=0; i < girada.filas; i++)
            for (int j=0; j < girada.columnas; j++)
                  girada.matriz[i][j] = imagen.matriz[imagen.filas-j -1][i];

      return girada;
}
```

Imagen de Entrada

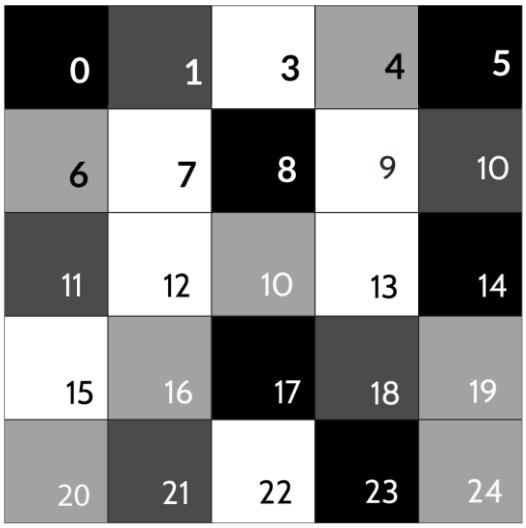
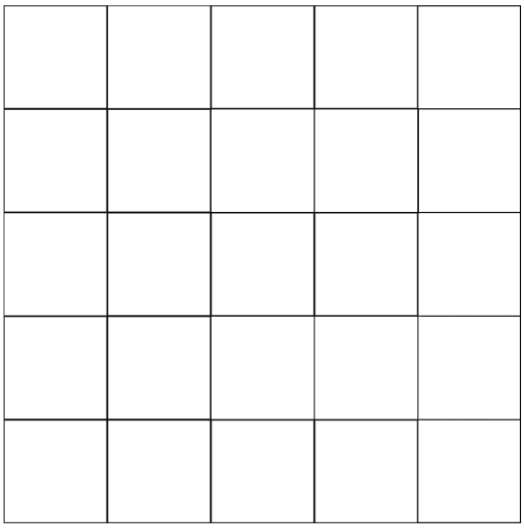
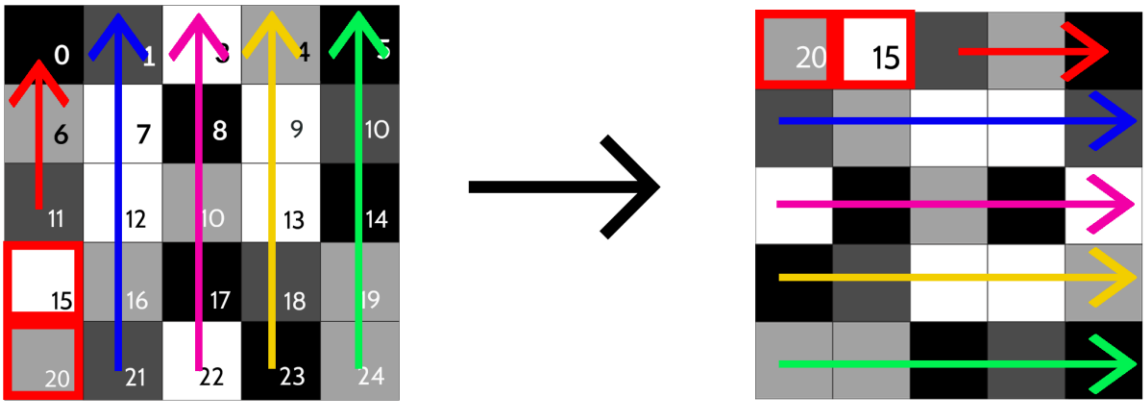


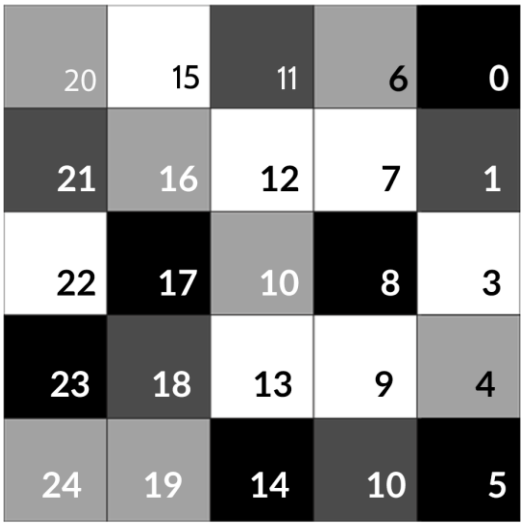
Imagen de Salida



Se va insertando a la imagen de salida los pixeles de la imagen de entrada de abajo arriba y en orden ascedente de columna y fila.



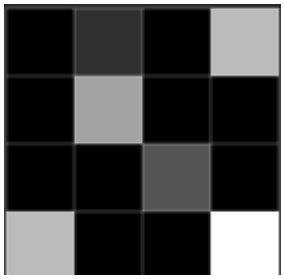
Finalmente queda como



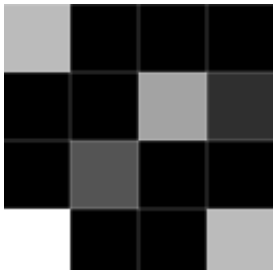
Ejemplos

Mona Lisa

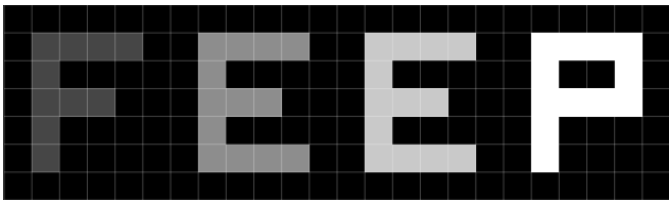
Mona Lisa Rotada



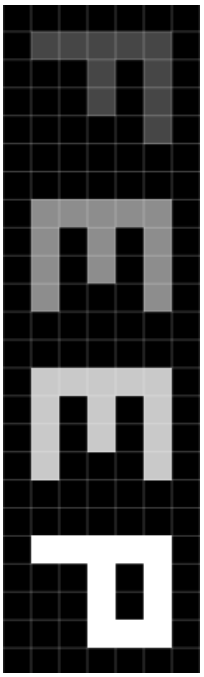
Ejemplo



Ejemplo Rotado



FEEP



FEEP Rotado

Programador: Salvador Romero

CONTRASTE

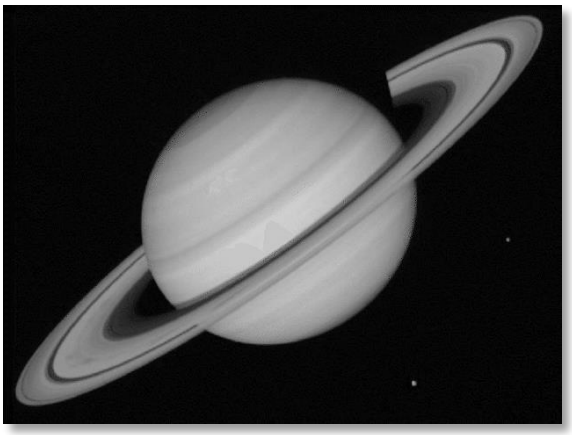
Función que aumenta el contraste de una foto. Es una función `void` que no devuelve nada. En su lugar usa la imagen de entrada con paso por referencia y la modifica.

El algoritmo de edición consiste en recorrer la matriz y comparar cada elemento de la matriz (cada pixel) con el valor 125. Si es menor lo colorea de negro y en caso contrario de blanco.

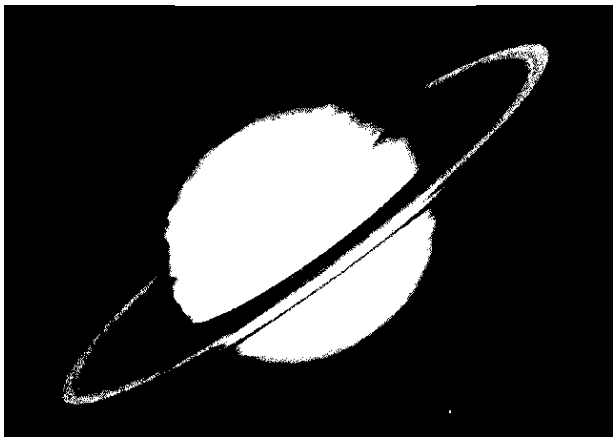
```
void Contraste(Imagen &imagen){
   for (int i=0; i < imagen.filas; i++)
      for (int j=0; j <imagen.columnas; j++){
         if (imagen.matriz[i][j] < 125)
            imagen.matriz[i][j] = NEGRO;
         else
            imagen.matriz[i][j] = BLANCO;
      }
}
```

Ejemplos

Saturno



Saturno con más contraste



Mona Lisa

Mona Lisa con más contraste

Programador: Santiago Hernández

NEGATIVO

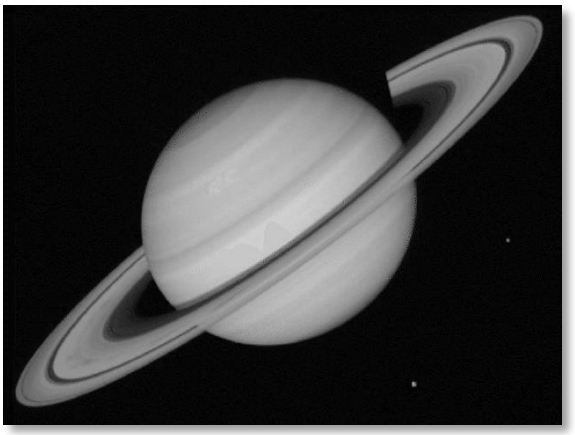
Método que sirve para tornar cada color de una foto en escala de grises por su opuesto, modificando la imagen de forma que quede en negativo.

El algoritmo de edición consiste en recorrer la matriz y asignar a cada elemento su opuesto, de modo que sus valores sumen 255 (BLANCO). Esto es que a cada elemento le corresponde el valor BLANCO(255) – VALOR_ELEMENTO(X)

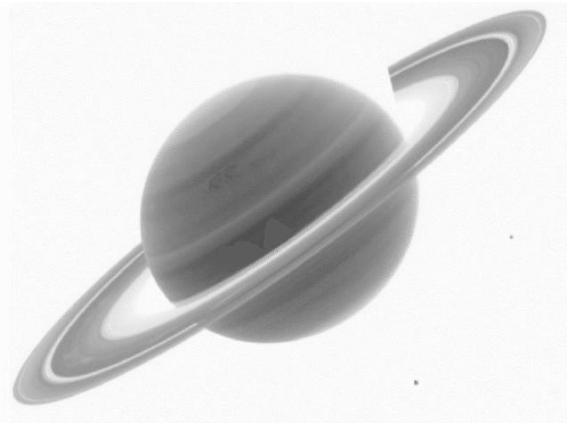
```
void Negativo(Imagen&imagen){
      for(int i = 0; i < imagen.filas; i++)
            for(int j = 0; j < imagen.columnas; j++)
                  imagen.matriz[i][j] = BLANCO - imagen.matriz[i][j];
}
```

Ejemplos

Saturno



Saturno en negativo



Mona Lisa

Mona Lisa en negativo