Diseño General

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

* Alias de short intcomo 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 para la entrada de datos de la imagen, por redirección. Modifica la matriz “entrada” con los datos introducidos. (Paso por referencia).

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 muestra la pantalla con “cout”, resultando en un archivo .pgm al usar redirección. El parámetro usado es la imagen de salida. El criterio de espacios en blanco y saltos de línea se basa en el formato pgm. Los dos primeros saltos de línea diferencian la “cadena mágica” del tamaño de la imagen y del número máximo usado para representar el color. A partir de ese momento, se imprime la matriz separado cada elemento con un espacio en blanco y una nueva línea por cada fila.

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***: contiene la declaración de la imagen de entrada y de salida, así como la lectura de los daros de la imagen de entrada. Posteriormente se aplica la función de rotar pasando la imagen de entrada como parámetro y devolviendo la girada, que se asigna como imagen de salida y se muestra mediante la función Salida().

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.

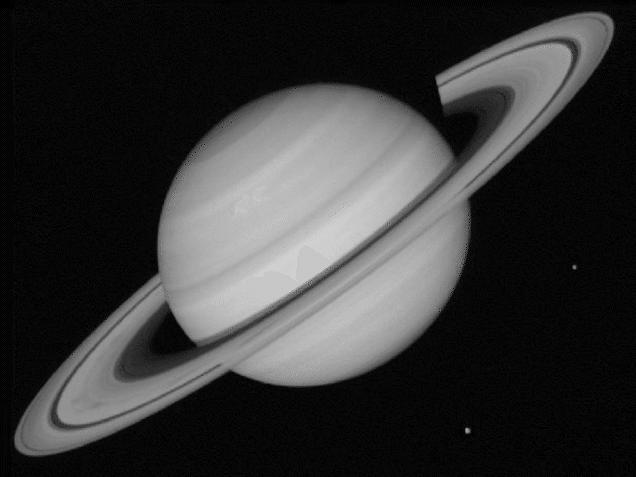
|  |
| --- |
| void Blanquear (Imagen &imagen){  for (int i=0; i < imagen.filas; i++)  for(int j=0; j < imagen.columnas; j++)  imagen.matriz[i][j] = BLANCO;  } |

Ejemplos.

Mona Lisa - Blanqueada

Mona Lisa



****

Saturno - Blanqueado

Saturno

(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.

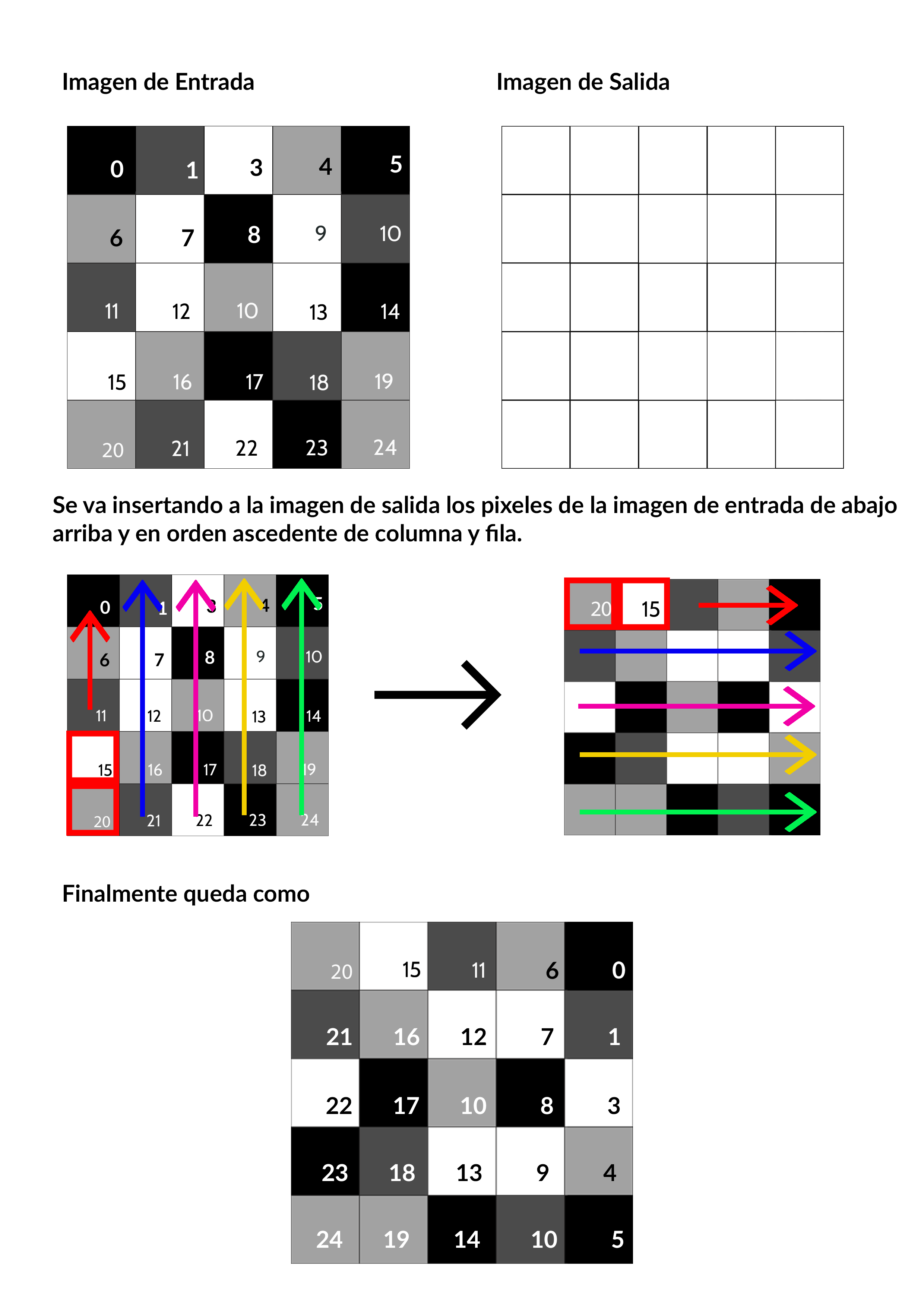
El algoritmo de giro de imagen consiste en asignar a cada elemento (i,j) de la matriz de salida los valores del elemento (imagen.filas - j - 1, i) de la imagen de entrada, donde imagen.filas = m.

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;  } |



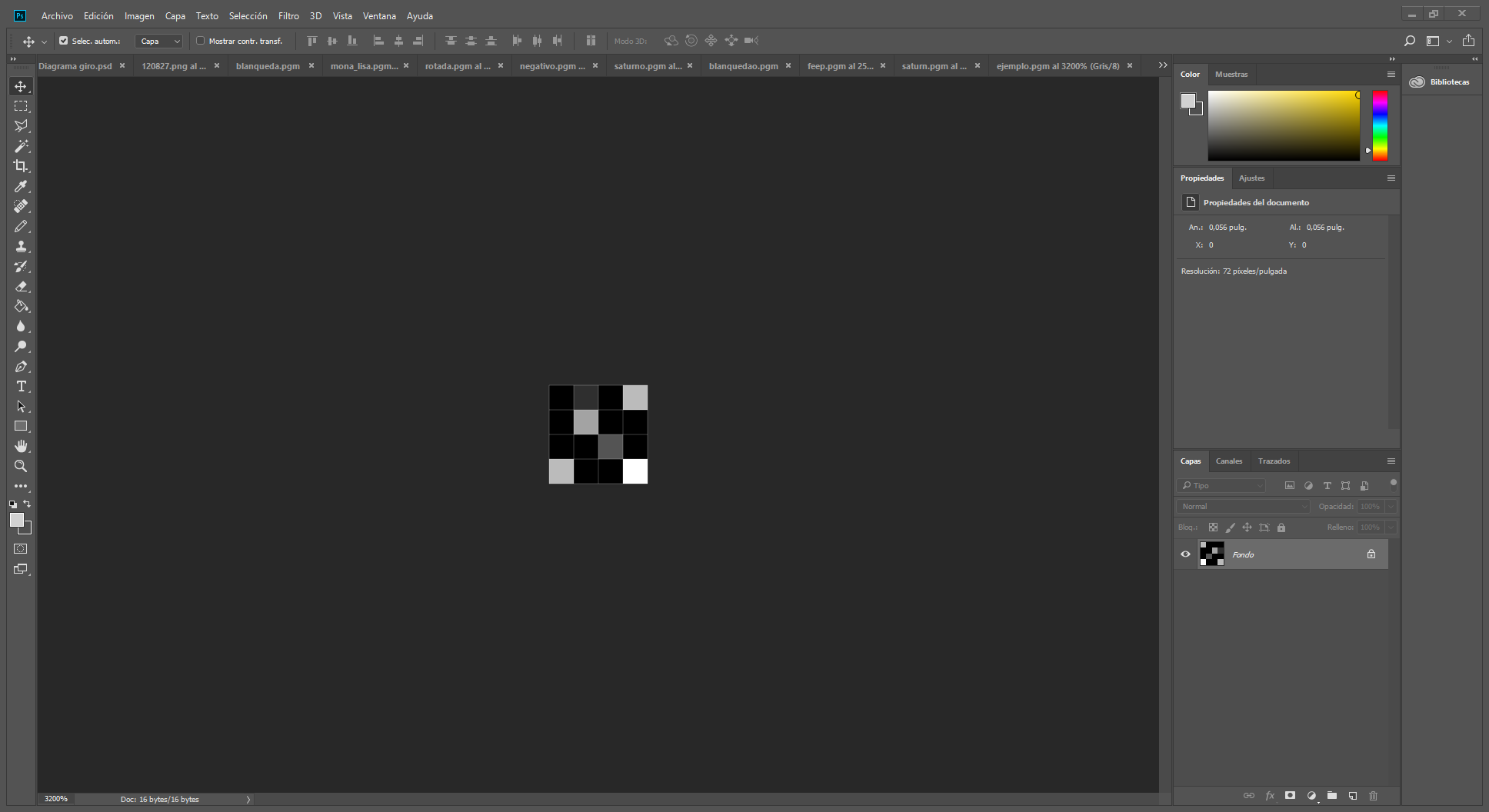
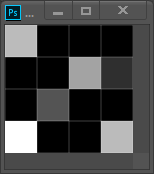
Ejemplos



Mona Lisa

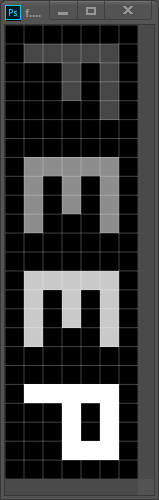


Mona Lisa Rotada

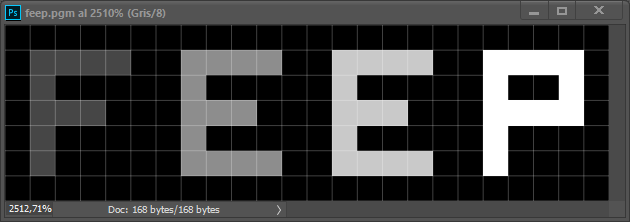


Ejemplo

Ejemplo Rotado

****

FEEP Rotado

****

FEEP

**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.

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;

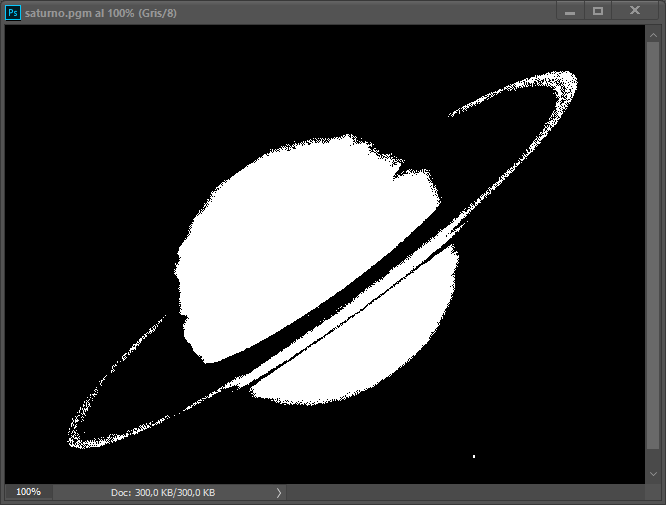
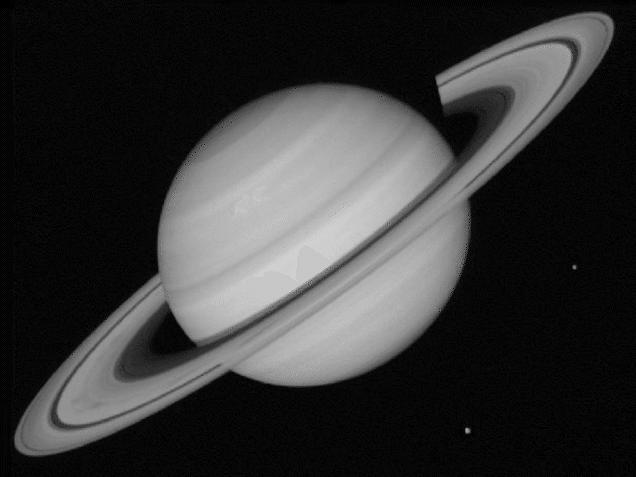
else

imagen.matriz[i][j] = BLANCO;

}

}

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.

Ejemplos****

Saturno con más contraste

Saturno



Mona Lisa

con más contraste

Mona Lisa