

Trabajo Práctico

Manipulación de imágenes sin compresión

Modalidad de entrega

La entrega es grupal, en grupos de 4 personas.

Si algún integrante dejara la materia, las condiciones son las mismas para el resto de los integrantes del TP: el grupo asume el compromiso de hacer el TP completo y en caso de que algún integrante deje la materia o no trabajara para la entrega del TP, los integrantes restantes del grupo deberán hacer el compromiso asumido inicialmente.

Se deberá entregar un archivo con el siguiente formato:

TP_Tópicos_2024_1c_viernes_{NOMBRE_DE_GRUPO}.zip

Por ejemplo: Si el grupo se llamase "Invisible" -y sus integrantes fueran Spinetta, Cerati, García y Napolitano- el archivo debería llamarse

TP_Tópicos_2024_1c_viernes_INVISIBLE.zip

Dentro de ese archivo debe haber solamente dos archivos fuentes (sin ninguna carpeta que los contenga) llamados funciones_estudiante.h y funciones_estudiante.c

El archivo funciones_estudiante.c tiene en un comentario los campos a completar con los datos de sus integrantes, y una sección de comentarios para que el corrector lea. Esta última sección es de uso opcional.

Los grupos deben ser formados y notificados vía mensajería de MIEL hasta las 23:59 hs del viernes 3 de mayo de 2024. El nombre de cada grupo debe ser una palabra, y no se puede repetir con otro grupo.

El trabajo práctico tiene valor de parcial y consta de una entrega (grupal) y una defensa (individual).

Modalidad de defensa

La defensa es individual, en laboratorio. Se solicitará hacer modificaciones en el programa para extender sus funcionalidades.

Requisitos de entrega

La resolución del TP debe ser entregada vía plataforma MIEL. La fecha de vencimiento de la entrega es el día domingo 18 de mayo a las 23:59 hs.

Las entregas que no respeten la fecha indicada no serán válidas.

El trabajo práctico entregado debe funcionar correctamente, sin warnings ni errores de compilación y cumplir con todas las funcionalidades requeridas.

Debe cumplir con las especificaciones de nombre y formato de archivo, de lo contrario la entrega no será válida y el TP será desaprobado.

Los grupos deberán revisar adecuadamente el trabajo práctico previo a la entrega, dado que una vez entregado no se aceptarán reentregas.

Una vez entregado, en caso de cumplir con las condiciones planteadas de fecha, resolver adecuadamente cada uno de los puntos indicados, no tener warnings y tener el formato adecuado de archivo, el TP pasará a estar en estado “entregado”.

En caso de que el TP esté debidamente entregado y no cumpla con alguno de los requisitos, se podrá hacer una nueva entrega, y defensa. La nueva entrega tendrá nuevos requerimientos que se informarán debidamente. Y la defensa tendrá las mismas condiciones que la defensa inicial.

Enunciado

Se solicita realizar un programa en lenguaje C llamado *bmpmanipuleitor*, que a partir de un archivo BMP de 24 bits de profundidad genere otro archivo BMP con las modificaciones solicitadas como argumentos a main.

Por ejemplo:

```
bmpmanipuleitor --negativo imagenOriginal.bmp
```

Debe generar un archivo llamado *estudiante_negativo.bmp* que contenga la misma imagen, pero con los colores invertidos.

Del mismo modo, deberá soportar las siguientes funcionalidades:

- escala-de-grises // deberá promediar los valores de cada color RGB y transformarlo a gris
- aumentar-contraste // aumenta el contraste en un 25%
- reducir-contraste // reduce el contraste en un 25%
- tonalidad-azul // aumenta en un 50% la intensidad del color azul
- tonalidad-verde // aumenta en un 50% la intensidad del color verde
- tonalidad-roja // aumenta en un 50% la intensidad del color rojo

- recortar // reduce el tamaño de la imagen al 50%, sin cambiar sus proporciones, solamente descarta lo que exceda ese tamaño. Por ejemplo: una imagen de 1000px x 500px, deberá mantener todos los píxeles que estén entre 0 y 499 en el eje X y entre 0 y 249 el eje Y.

- rotar-derecha // gira la imagen 90 grados a la derecha
- rotar-izquierda // gira la imagen 90 grados a la izquierda

Nota: los argumentos pueden enviarse de a uno, o de a varios, de modo que en un solo llamado al programa, ejecute todas las modificaciones y genere todos los archivos necesarios. Ejemplo de un llamado al programa, y de los archivos generados:

```
ezequiel@thompson:~/Progra/BMP/bmpmanipuleitor/bin/Debug$ ./bmpmanipuleitor --negativo --escala-de-grises --aumentar-contraste --reducir-contraste unlan.bmp
ezequiel@thompson:~/Progra/BMP/bmpmanipuleitor/bin/Debug$ ls -l
total 1308
-rwxrwxr-x 1 ezequiel ezequiel 28336 abr  8 08:27 bmpmanipuleitor
-rw-rw-r-- 1 ezequiel ezequiel 259339 abr  8 13:03 docente_aumentar-contraste.bmp
-rw-rw-r-- 1 ezequiel ezequiel 259339 abr  8 13:03 docente_escala-de-grises.bmp
-rw-rw-r-- 1 ezequiel ezequiel 259339 abr  8 13:03 docente_negativo.bmp
-rw-rw-r-- 1 ezequiel ezequiel 259339 abr  8 13:03 docente_reducir-contraste.bmp
-rw-rw-r-- 1 ezequiel ezequiel 259338 abr  7 17:18 unlan.bmp
ezequiel@thompson:~/Progra/BMP/bmpmanipuleitor/bin/Debug$
```

--achicar // reduce la imagen al 50%. Por ejemplo: una imagen de 1000px x 500px, deberá reescribir todos los píxeles para que se ubiquen entre 0 y 499 en el eje X y entre 0 y 249 el eje Y, formando la misma imagen inicial (con peor calidad, claro).

--monocromo // modifica la imagen para que en lugar de ser de 24 bits de profundidad, sea de 1 bit de profundidad. Este tipo de imágenes almacena en un byte 8 píxeles distintos, que pueden obviamente tomar valores 0 ó 1. El criterio para saber si se guarda un 0 ó un 1, es que el promedio de los valores RGB (ver más adelante) dé entre 0 y 127 ó entre 128 y 244.

Estos últimos tres puntos requieren modificaciones en el encabezado del archivo BMP, cuya descripción es la siguiente:

Bytes	Información
0, 1	Tipo de fichero "BM"
2, 3, 4, 5	Tamaño del archivo
6, 7	Reservado
8, 9	Reservado
10, 11, 12, 13	Inicio de los datos de la imagen
14, 15, 16, 17	Tamaño de la cabecera del bitmap
18, 19, 20, 21	Anchura (píxeles)
22, 23, 24, 25	Altura (píxeles)
26, 27	Número de planos
28, 29	Tamaño de cada punto
30, 31, 32, 33	Compresión (0=no comprimido)
34, 35, 36, 37	Tamaño de la imagen
38, 39, 40, 41	Resolución horizontal
42, 43, 44, 45	Resolución vertical
46, 47, 48, 49	Tamaño de la tabla de color
50, 51, 52, 53	Contador de colores importantes

Muchos campos de este encabezado no son necesarios para la realización del TP. Es necesario contemplar que hay campos de 16 y de 32 bits y que deben ser almacenados

en el tipo de dato apropiado, para su correcta interpretación.

Alto, ancho, tamaño de imagen, tamaño de cada punto/píxel, inicio de los datos de la imagen son datos de suma importancia para la realización del TP.

Todas las modificaciones deben hacerse sobre los archivos funciones_estudiante.h y funciones_estudiante.c

Detalles a tener en cuenta

Imagenes como las que vamos a utilizar en este TP están formadas por un encabezado y un array de bits que representan la imagen. Este array de bits, al haber definido que cada pixel tiene 24 bits de profundidad, se debe interpretar como un pixel al contenido de 3 bytes consecutivos.

Podría ser de utilidad utilizar un programa que haga un dump hexadecimal, si no conocen ninguno... podrían hasta resolverlo como punto extra.

```
ezequiel@thompson: ~/Progra/BMP/bmpmanipuleitor/bin/Debug$ hexdump -x unlam.bmp | head -n 20
00000000  4d42  f50a  0003  0000  0000  008a  0000  007c
00000010  0000  0168  0000  00f0  0000  0001  0018  0000
00000020  0000  f480  0003  2e23  0000  2e23  0000  0000
00000030  0000  0000  0000  00ff  ff00  0000  00ff
00000040  0000  0000  ff00  4742  7352  0000  0000  0000
00000050  0000  0000  0000  0000  0000  0000  0000  0000
*
00000070  0000  0000  0000  0000  0000  0004  0000  0000
00000080  0000  0000  0000  0000  0000  392d  2e2d  2b36
00000090  3237  4029  323b  3733  292b  2633  362c  2b29
000000a0  2734  2826  211c  1925  2d20  0f1f  1220  231a
000000b0  2216  1c29  1715  140b  0a16  1315  1308  0713
000000c0  1213  1408  0d18  1f1c  1e16  1c26  271c  171f
000000d0  1e26  2c17  2824  3a43  3213  182b  353d  4015
000000e0  1035  3440  4515  0b39  2f3b  3305  1427  343f
000000f0  3305  0926  2a37  4c1b  123c  3343  4c18  133b
0000100  3849  430c  0432  2e40  5416  0e42  3d50  5817
0000110  2448  596a  5e18  214d  5768  540d  1240  4357
0000120  6220  164f  4254  5016  113e  3748  3500  1024
0000130  3647  4f18  183e  4052  4004  102e  3a4c  5c1e
ezequiel@thompson: ~/Progra/BMP/bmpmanipuleitor/bin/Debug$
```

Los tres bytes consecutivos son valores que van entre 0 y 255 (unsigned char), o más claramente en hexadecimal: 0x00 hasta 0xff, que indican qué cantidad de color (rojo, verde o azul) va a tener la imagen.

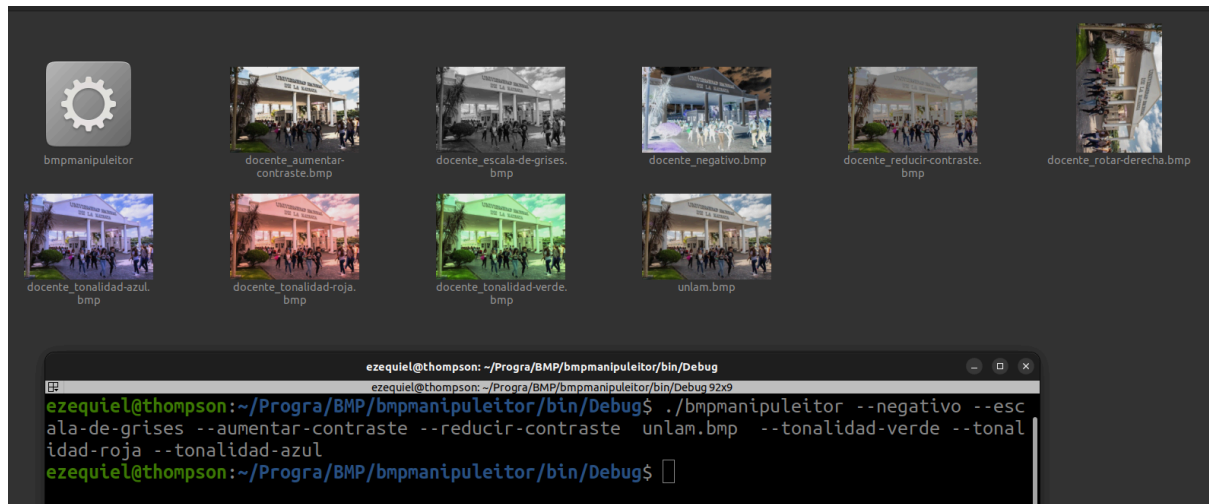
Si las tres variables RGB (Red, Green, Blue; o en español Rojo, Verde y Azul) están igualadas, el color será gris, y dependiendo de su intensidad, se acercará más al blanco o al negro.

El color 0xfffff equivale al blanco, mientras que el 0x000000 equivale al negro.

Si quisiéramos escribir un color azul puro, debería ser 0x0000ff (porque tiene cero en la componente roja, cero en la componente verde y 255 en la componente azul).

De ese modo se pueden armar 2^{24} colores distintos.

A continuación un ejemplo de un comando ejecutado, y los resultados (en vista previa) obtenidos:



Nota: el ejemplo fue ejecutado utilizando un sistema GNU/Linux, y como en cualquier sistema tipo Unix, los ejecutables utilizan *./nombreDelPrograma*, en lugar del formato que utiliza Windows de ser nombreDelPrograma.exe