PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
2° semestre 2015

Actividad 22

Manejo de Bytes

Instrucciones

Sus conocimientos en programación se han hecho muy famosos y han llegado a oidos de un gran fotógrafo. Este le pide por favor que programe algunas funciones que modifiquen imágenes en formato **PNG**. Estas funciones son `girar` y `grises`.

Formato PNG

Cada imagen **PNG** (con la extensión `.png`), en su estructura de bytes, comienza siempre con los bytes `137 80 78 71 13 10 26 10`. Luego de esto, los bytes se ordenan en bloques o *chunks*.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	...	n	n+1	n+2	n+3	n+4	n+5	n+6	...
89	80	78	71	13	10	26	10	chunk						...	chunk							...

Los bloques tienen una estructura establecida, que es la siguiente:

Estructura de un chunk o bloque			
Largo de información del bloque	**Tipo de bloque**	**Información**	**CRC**
4 bytes	4 bytes	Largo	4 bytes

Donde los primeros 4 bytes indican el largo que posee la sección de información del chunk. Los siguientes 4 bytes pueden ser decodificados para obtener un string de 4 caracteres que indican qué tipo de información posee el bloque. Luego viene la sección de la información, que posee la cantidad de bytes indicadas en un principio, esta información varía dependiendo del tipo de bloque en el que se esté trabajando. Por último, los siguientes 4 bytes es un código para la información del chunk.

Tipos de bloque

- `IHDR`
  En este bloque se tiene la metadata de la imagen.

IHDR	
Cantidad de bytes	Descripción
4	Ancho en pixeles
4	Alto en pixeles
1	Profundidad de bits
1	Tipo de colores
1	Tipo de compresión
1	Tipo de filtro
1	Tipo de entrelazado

- `IDAT`
  Este bloque posee los bytes de la imagen comprimidos. En una imagen pueden existir varios bloques `IDAT`. Para descomprimir la información de la imagen y poder trabajar con los datos es necesario concatenar todas las secciones de información de la imagen.

- `IEND`
  Este bloque indica el término del archivo y en la sección información no posee datos (el largo de la sección es 0).

Como se indica en `IDAT` la información contenida dentro del chunk está comprimida. Se le entrega la función `decompress`, `compress` y `crc32` del módulo `zlib` que le permitirán descomprimir los datos para poder trabajar con los pixeles, comprimir para reescribir el archivo y generar el código `CRC` final de cada chunk, respectivamente.

Requerimientos

- Obtener la información de la imagen a partir de la lectura de bytes.
- Generar una matriz con los pixeles.
- Cambiar la imagen a escala de grises.
- Girar la imagen [Bonus].
- Escribir un archivo **PNG** con la información obtenida.

Notas

- A continuación se muestra cómo está la información originalmente. Cuando usted construya su matriz, ignore la primera columna.

$$\begin{bmatrix} 8 & (64,123,55) & (67,144,65) & \cdots & (68,53,60) \\ 0 & (157,87,202) & (17,64,253) & \cdots & (181,39,131) \\ \vdots & \vdots & & \ddots & \vdots \\ 4 & (11,52,198) & (3,53,139) & \cdots & (1,74,167) \end{bmatrix}$$

- La imagen que te entrega el fotógrafo tiene Tipo de colores (o *colortype*) igual a 2, que indica que la imagen está en formato **RGB**, esto quiere decir que cada pixel tiene 3 bytes.
- Una imagen en blanco y negro tiene un Tipo de colores igual a 0, en este formato cada pixel tiene 1 byte.
- El `CRC` de cada chunk corresponde a 4 bytes equivalentes al número obtenido de aplicar la función `crc32` a los bytes del tipo de chunk concatenado a la información del bloque.

To - DO

- (1.50 pts) Método `getData`.
- (1.50 pts) Método `bytes2matrix`.
- (1.50 pts) Método `grey`.
- (1.50 pts) Método `writeImage`
- **[BONUS]** (1.00 pts) Método `rotate`.

Tips

- `archivo.read(n)`: lee `n` bytes del archivo.
- `int.from_bytes(x, byteorder='big')`: para transformar `x` desde `bytes` a `int`.
- `x.to_bytes(n, byteorder='big')`: para transformar `x` desde `int` a `n` cantidad de `bytes`.
- Transformar de formato **RGB** a escala de grises: el byte de cada pixel en escala de grises debe ser el promedio de los 3 bytes de los pixeles en **RGB**.

Figura 1: Sin el bonus

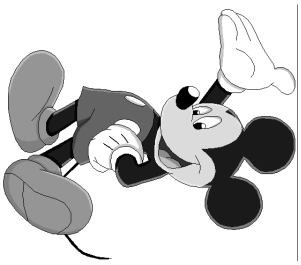

Figura 2: Con el bonus