Universidad de los Andes
Facultad de Ingeniería

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN
ISIS1304
FUNDAMENTOS DE INFRAESTRUCTURA TECNOLÓGICA

Proyecto 1 - 2015-2

Este proyecto vale 15% de la nota del curso.
Debe ser elaborado en grupo (mínimo 2, máximo 3 integrantes).
No se permite ningún tipo de consulta entre grupos.
Se debe entregar por Sicua a más tardar el 11 de octubre a las 23:55

A. OBJETIVOS

- Practicar el lenguaje C y desarrollar un programa de complejidad pequeña.
- Conocer las operaciones de C para el manejo de bits.
- Manipular pistas de audio en formato wav.

B. DESCRIPCIÓN DEL PROBLEMA

El objetivo del proyecto es tomar dos pistas monofónicas (almacenadas en archivos de sonido con formato wav), y unirlas en un solo archivo con una pista estereofónica; es decir, cada una de las pistas monofónicas se convertirá en uno de los canales de la pista estereofónica.

Estructuras de datos

Un archivo wav está compuesto de múltiples segmentos, entre los cuales se encuentra uno que indica el tamaño de la información del sonido y contiene las muestras de la onda que produce el sonido.

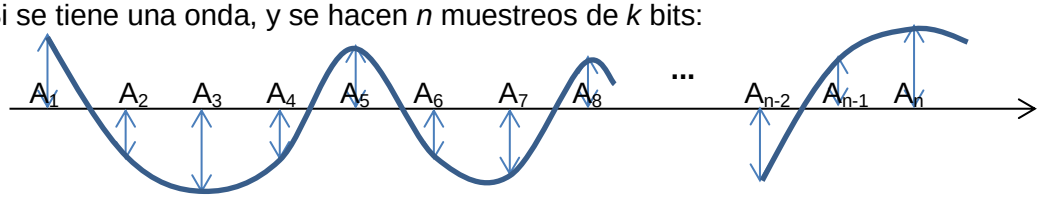
El esqueleto de programa que se entrega para el proyecto, ya incluye procedimientos para leer y escribir archivos wav, así que usted no tiene que ocuparse de eso, sino de manipular los datos en memoria. Para esto, los procedimientos del esqueleto le retornan una estructura simplificada que contiene los datos principales del archivo, tal como se muestra a continuación:

```
struct WaveData {
  unsigned int SoundLength, numSamples, BitsPerSample;
  unsigned char *Sample;
};
```

Dentro de la estructura propuesta en el proyecto (**`WaveData`**) se encuentran los siguientes campos: `SoundLength` es la cantidad de bytes en todo el Sample, `numSamples` es el número de muestreos, `BitsPerSample` es la cantidad de bits que tiene cada muestreo y el último campo de la estructura, `Sample`, es un arreglo de `char` que contiene los muestreos de la onda que produce el sonido. Estos arreglos de los archivos de entrada son los que se manipularán para unificarlos en una pista estereofónica.

Unir pistas monofónicas

Si se tiene una onda, y se hacen *n* muestreos de *k* bits:



La pista almacenada (monofónica) será:

k bit	k bit	k bit	k bit	k bit		k bit	k bit	k bit
A_1	A_2	A_3	A_4	A_5	**...**	A_{n-2}	A_{n-1}	A_n

Note que los muestreos se almacenan uno detrás de otro sin ningún espacio entre ellos.

El objetivo del programa realizado en el proyecto es combinar 2 pistas monofónicas (únicamente la parte del Sample en la estructura) A y B en una estereofónica. Se reciben dos arreglos de entrada y se debe retornar otro con las dos pistas entremezcladas:

k bit	k bit	k bit	k bit		k bit	k bit	k bit	k bit
A_1	B_1	A_2	B_2	**...**	A_{n-1}	B_{n-1}	A_n	B_n

Tenga presente que el tamaño de los muestreos, *k*, es un número entre 1 y 16 y no necesariamente es múltiplo de 8, así que los muestreos no necesariamente están alineados con los bytes:

k bit	k bit	k bit	k bit	k bit	k bit	k bit	k bit
byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	byte 8	byte 9

Si el tamaño total en bits (*k*n*) no es múltiplo de 8, después del último grupo de k bits, habrá unos bits de relleno (en 0) para completar el último byte.

El programa

En el archivo adjunto ("main.c"), encuentra el esqueleto del programa. El programa se invoca por línea de comando, y recibe tres parámetros: los primeros dos parámetros son las pistas de origen (monofónicas), y el ultimo es el archivo de destino.

El programa ya tiene escritos algunos procedimientos para leer los archivos con formato `.wav` y para escribir un archivo con el mismo formato. **Estos métodos no deben ser modificados.**

El procedimiento de lectura (`cargarWAVE`) recibe el nombre de un archivo y se encarga de leerlo en una estructura `HeaderType` y una de tipo `WaveData`.

El procedimiento de escritura (`escribirWAVE`) recibe el nombre de un archivo y se encarga de crearlo con los datos de una estructura `HeaderType` y los de una de tipo `WaveData`.

Los procedimientos que debe desarrollar son `unirArchivosWAVE`, `escribirMuestreo` y `leerMuestreo`. Las funciones de los procedimientos son las siguientes:

- `unirArchivosWAVE`: se encarga de unir dos pistas monofónicas en una pista estereofónica. Recibe cuatro parámetros: `unsigned short *parte1`, `unsigned short *parte2`, `unsigned short *salida`, `int bitsPorMuestreo`, los cuales corresponden a un vector de short que contiene los muestreos de la primera pista monofónica, un vector de short que contiene los muestreos de la segunda pista monofónica, un vector de short en el que se guardará la fusión de las dos pistas monofónicas y tamaño en bits de las muestras, respectivamente.
- `escribirMuestreo`: se encarga de escribir un muestreo en una pista a partir de una posición específica. Recibe cuatro parámetros: `unsigned short *pista`, `int bitPos`, `unsigned short muestreo`, `int bitsPorMuestreo`, los

cuales corresponden a un vector de short que contiene los muestreos de una pista monofónica, la posición del muestreo en la pista a partir de la cual se desea escribir, short (el muestreo) que se desea escribir en la pista y tamaño del muestreo en bits, respectivamente.

- `leerMuestreo`: se encarga de retornar un muestreo de una pista, ubicado en una posición específica. Recibe tres parámetros: `unsigned short *pista`, `int bitPos`, `int bitsPorMuestreo`, los cuales corresponden a un vector de short que contiene los muestreos de una pista monofónica, posición del muestreo en la pista a partir de la cual se desea leer y tamaño del muestreo en bits, respectivamente. Este procedimiento retorna el muestreo de la pista que se encuentra en la posición indicada.

El generador

Por otro lado, en el archivo adjunto ("`generar.exe`") encuentra un programa que genera dos pistas monofónicas a partir de una pista estereofónica. Este programa se le entrega únicamente con el fin de generar muestras útiles para probar el su código.

Debe ejecutarlo desde consola, siguiendo los pasos que se muestran a continuación:

1. Cambie el directorio actual al directorio en el que se encuentra el ejecutable. Es decir, si el generar.exe se encuentra en la ruta *C:\Users\Documents*, debe ejecutar el comando:
   ```
   $ cd C:\Users\Documents
   ```
   Alternativamente, desde Windows puede hacer `shift + click derecho` sobre el directorio, y, en el menú desplegable que aparece, seleccionar "`Abrir ventana de comandos aquí`"; esto abre la consola con el directorio actual ya seleccionado.

2. Luego, debe ejecutar el programa. Para esto teclee el nombre del comando seguido de los parámetros. El programa recibe tres parámetros: el primero es el archivo con formato `.wav` que contiene la pista estereofónica que se desea descomponer y, los otros dos corresponden a los archivos en los que se guardarán las dos pistas monofónicas resultantes. Estos parámetros debe especificarlos entre comillas, separados por espacios simples. Ejemplo:
   ```
   $ generar.exe "C:\Users\Documents\Muestra\estereo.wav"
   "C:\Users\Documents\Muestra\mono1.wav" "C:\Users\Documents\Muestra\mono2.wav"
   ```
   Donde `C:\Users\Documents\Muestra\estereo.wav` corresponde a la pista estereofónica entrante y `C:\Users\Documents\Muestra\mono1.wav` y `C:\Users\Documents\Muestra\mono2.wav` corresponden a las pistas monofónicas en las que se guardará el resultado.

Una vez en ejecución, el programa solicita especificar la cantidad de bits por muestra en la pista; este valor corresponde a la precisión con la que se reproduce la señal y equivale a la resolución del audio.

Nota: el programa exige que la pista de audio original sea de 16 bits en estéreo.

C. ESPECIFICACIONES

- Los programas se deben escribir en C (en Visual Studio). Nota importante: los programas se calificarán únicamente usando el ambiente de visual; si el programa no compila en este ambiente, se considerará que no corre (así compile en otros ambientes).
- Legibilidad del programa: sangrar el programa; escribir comentarios explicando el código.
- Debe respetar la estructura del código entregado. En particular, debe usar los procedimientos y variables del esqueleto.

D. CONDICIONES DE ENTREGA

- Entregar el código fuente junto con el ejecutable en un archivo *.zip. **Al comienzo del archivo fuente escriba los nombres de los miembros, sus códigos y correos, de lo contrario no será evaluado (ver esqueleto)**. Si su programa no funciona o si su solución tiene particularidades, puede enviar un archivo .doc explicando por qué cree que no funciona o qué fue lo que hizo.
- El trabajo se realiza en grupos de máximo **3** personas. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota de todos los miembros.
- El proyecto debe ser entregado por Sicua por uno solo de los integrantes.
- **Se debe entregar por Sicua a más tardar el 11 de octubre a las 23:55.**

E. CASOS DE PRUEBA

Para hacer pruebas se adjuntan 3 casos de prueba (6 archivos wav, 2 archivos por caso), para que puedan mezclarlos. Además se adjunta el resultado de las mezclas, para que aprecien cómo se deben escuchar al aplicarles los efectos (3 archivos más).

Las pruebas de funcionamiento están basadas en la comparación binaria de los archivos.

F. CRITERIOS DE CALIFICACIÓN PARA LOS PROGRAMAS

La calificación consta de dos partes:

- Ejecución (50%). Para las funciones propuestas se harán 5 pruebas: tres de los casos de prueba entregados y otros dos nuevos.
- Inspección del código (50%). Se consideran tres aspectos:
  - 10% - legibilidad (nombres dicientes para variables, comentarios e indentación)
  - 20% - manejo de bits (uso de los operadores de bits de c: >>, &, etc.)
  - 20% - manejo de la estructura de datos (recorrido, manejo de los elementos).

G. RECOMENDACIONES

- Recuerden que los trabajos hechos en grupo y entregados individualmente (o en grupos diferentes al original) son una forma de fraude académico.