Algoritmos y Programación II

Trabajo Práctico Individual N 1:

“De las matemáticas a la música”

Objetivo

Desarrollar un programa que se ejecute por línea de comando para generar un archivo *wav* a partir de un archivo de texto, en donde se indicarán las notas según la notación anglo-sajona y su duración, expresada en milisegundos.

Datos de entrada

La melodía estará descrita en un archivo de texto. El nombre de este archivo será el primer parámetro del programa.

La primera línea contendrá un número que representará la duración total de la melodía expresado en milisegundos. Las líneas siguientes hasta el final del archivo tendrán dos datos cada una, los cuales estarán separados por un espacio. El primer dato corresponderá a la nota en notación anglo-sajona (ver tabla). El segundo dato será la duración de esa nota en milisegundos.

Nota	Notación Anglo-sajona	Frecuencia (Hz)
Do central	C4	261,63
Do# / Reb	C#4 / Db4	277,18
Re	D4	293,66
Re# / Mib	D#4 / Eb4	311,13
Mi	E4	329,63
Fa	F4	349,23
Fa# / Solb	F#4 / Gb4	370
Sol	G4	392
Sol# / Lab	G#4 / Ab4	415,30
La central	A4	440
La# / Sib	A#4 / Bb4	466,16
Si	B4	493,88

Pueden ver la tabla completa en la siguiente dirección:
http://es.wikipedia.org/wiki/Frecuencias_de_afinaci%C3%B3n_del_piano

Por ejemplo, un archivo de texto podría contener los siguientes datos:

```
1100
A3 300
D#5 400
H 150
Bb4 250
```

La H representará un silencio.

Una posible llamada al programa sería:

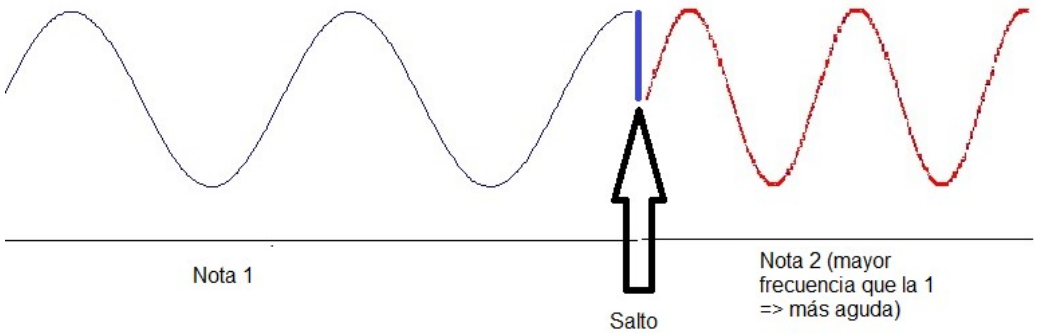
```
musica minue.txt minue.wav 8 44100
```

Procesamiento

El primer parámetro será el nombre del archivo de texto donde estarán las notas y su duración. El segundo parámetro será el nombre del archivo *wav* que deberán generar. El archivo *wav* a generar será mono (1 canal). Para representar cada muestra, se podrá utilizar 8, 16 o 32 bits. El tercer parámetro es el que indicará esto último. Finalmente, el cuarto parámetro, indicará la cantidad de muestras que habrá en un segundo. Este número deberá ser mayor o igual a 8000.

La melodía se creará generando muestras utilizando la función seno. La amplitud tendrá relación con el volumen y el valor que multiplica a la *x* con la frecuencia. Cuando la nota cambie, deberán tener en cuenta un factor de ajuste para que la próxima muestra no empiece desde cero produciendo un salto que se traduciría en un ruido en la melodía. Lo mismo deberán contemplar en los silencios, que no estarán representados por el valor cero, sino por el último valor de la nota anterior.

Gráficamente:



Frecuencias de las notas musicales

La escala musical que utilizamos actualmente en occidente está compuesta por 12 notas, que se van repitiendo en sucesivas escalas. Cada nota tiene una frecuencia diferente, que se va duplicando en cada octava. Por ejemplo, A4 (el la central) tiene una frecuencia de 440Hz, es decir, 440 ciclos por cada segundo. La siguiente octava, A5, tiene una frecuencia de 880Hz. Y la anterior, A3, 220Hz.

La separación entre cada nota es proporcional, como hay 12 notas en cada octava y, en las sucesivas octavas se duplica la frecuencia, la constante de proporcionalidad es

$$k = \sqrt[12]{2}$$

Es decir, sabiendo una sola frecuencia, por ejemplo, el la central (A4), que tiene 440Hz, se pueden sacar el resto de las frecuencias mediante el siguiente cálculo:

$$f = f_n k^s = f_n \left(\sqrt[12]{2}\right)^s$$

donde f_n representa la frecuencia de la nota base, por ejemplo A4, y *s* representa el salto o cantidad de notas que hay desde la nota base a la que se desea calcular.
Por ejemplo, si la nota base es A4 con frecuencia 440 Hz, y la que se quiere calcular es B5, *s* será 14, dado que entre A4 y A5 hay 12 notas, y entre el A5 y B5 (la y si) hay dos. Entonces la frecuencia de B5 será:

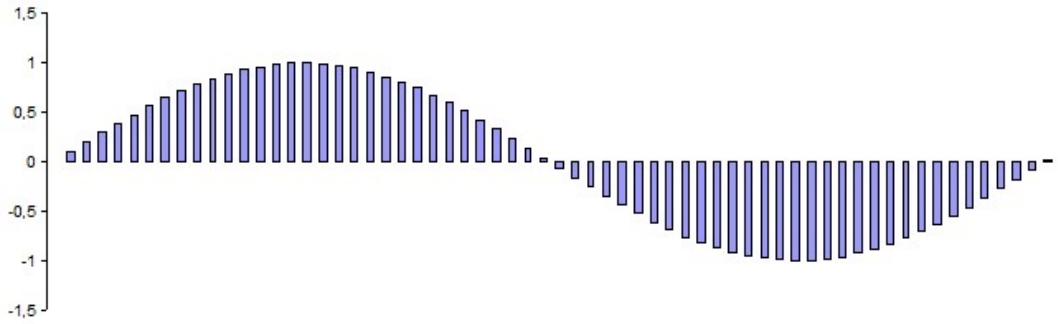
$$f = 440.\left(\sqrt[12]{2}\right)^{14} \approx 987.77$$

En cambio, F3 tendrá un salto de 16 notas pero en negativo, por lo tanto

$$f = 440.\left(\sqrt[12]{2}\right)^{-16} \approx 174.61$$

Digitalización de los sonidos

Dado que el formato de sonido en una computadora es digital, no podremos representar una onda continua, sin embargo, se pueden representar muchas muestras como se ve a continuación, lo cual da un efecto ondulatorio.



¿Cuántas muestras se deben representar en determinada cantidad de tiempo?

Esto depende de la velocidad de muestreo (el cuarto parámetro que se ingresa). Por ejemplo, la velocidad de muestreo estándar, de un disco de audio es de 44100Hz. Esto significa que se utilizan 44100 muestras por segundo. Por lo tanto, si un sonido tiene una duración de 500 milisegundos, se utilizarán 22050 muestras para su representación.

Si cada muestra demanda, por ejemplo, 8 bits para representarla, se necesitarán 22050 bytes para esa nota.

Formato de un archivo wav

Los archivos wav tienen un formato determinado. Se dividen en *chunks* (pedazos). Usaremos tres chunks: el primero será el *header* (chunk de cabecera), el segundo el del formato y el tercero el de los datos propiamente dichos (las muestras).

Chunk	Cantidad de bytes	desde-hasta	Dato
Header	4	0 - 3	"RIFF"
	4	4 - 7	Tamaño del archivo a partir del byte 8
	4	8 - 11	"WAVE"
Formato	4	12 - 15	"fmt "
	4	16 - 19	Tamaño del chunk de formato (16 bytes)
	2	20 - 21	Formato de audio: 1 = sin compresión
	2	22 - 23	cc = cantidad de canales: 1 = Mono
	4	24 - 27	mps = muestras por segundo (4to parámetro)
	4	28 - 31	br = byte rate = mps * cc * bpm / 8
	2	32 - 33	ba = block align = cc * bpm / 8
	2	34 - 35	bpm = bits por muestra: 8, 16 o 32 (3er param)
Datos	4	36 - 39	"data"
	4	40 - 43	Tamaño del chunk de datos = n
	n	44 -	Muestras

Nota: se debe tener en cuenta que el formato numérico es *little-endian*, por lo tanto, si queremos representar el número 6000425, que en hexadecimal es 5B8F29, en 4 bytes, la representación natural de dicho número será:

00 5B 8F 29

Sin embargo, en formato little-endian, los bytes menos significativos irán antes que los más significativos, es decir, el número se representará como:

29 8F 5B 00

Este formato es el que utilizan los procesadores como *Intel*.

Nota 2: si las muestras son de 8 bits, la escala irá de 0 a 255. En cambio con 16 o 32 bits se tomará el dato con signo.

Nota 3: no se utilizarán doble sostenidos o doble bemoles. Tampoco se utilizarán sostenidos con notas que no tienen semi tonos, por ejemplo: E#4 es F4.

Sitios de utilidad

Conversor decimal / hexadecimal:
https://www.onlinehexeditor.com/

Cálculo de frecuencia de las notas:
http://elclubdelautodidacta.es/wp/2012/08/calculo-de-la-frecuencia-de-nuestras-notas-musicales/

Tabla con frecuencia de las notas:
http://es.wikipedia.org/wiki/Frecuencias_de_afinaci%C3%B3n_del_piano

Formato wav:
http://soundfile.sapp.org/doc/WaveFormat/

También:
http://www.topherlee.com/software/pcm-tut-wavformat.html

Formatos big-endian y little-endian:
http://nachocabanes.blogspot.com.ar/2006/12/el-formato-big-endian-y-el-little.html

Conversor:
https://www.rapidtables.com/prog/endianess.html

Referencia de las librerías de C++ con sus funciones:
http://www.cplusplus.com/reference/

De la librería cmath necesitan las funciones *sin* y *exp*:
http://www.cplusplus.com/reference/cmath/?kw=cmath

Normas de entrega

Se deberá subir un archivo comprimido al campus, en un link que se habilitará para esta entrega. Este archivo deberá tener un nombre formado de la siguiente manera:

ApellidoNombre-TP1

Deberá contener solo los archivos fuente.

La fecha de entrega vence el día lunes 1/4/19 a las 23.55hs.

Se evaluará: funcionalidad, eficiencia, algoritmo utilizado, buenas prácticas de programación (como nombres, sangrías, etc.), modularización, comentarios.