SINTAXIS DE ALGUNOS OPCODES DE CSOUND

Es sólo una referencia rápida para los operadores más usados en las prácticas del curso. La referencia completa de la última versión de Csound está en la página `www.csounds.com/manual/html` y en el entorno gráfico del CsoundQt.

La primera letra de las variables indica su tipo. Cuando comienzan con x quiere decir que se pueden utilizar variables de cualquier tipo (i–, k–, a–), aunque ello puede cambiar el comportamiento del operador. Los elementos escritos entre corchetes [...] son opcionales.

Extractores de valores de formato y cabecera de un fichero:

filebit Devuelve el número de bits que codifica cada valor de las muestras del fichero. Su sintaxis es:

```
ivalor  filebit  ifich
```

donde `ivalor` – variable que almacena el valor extraído del bloque de formato del fichero
`ifich` – nombre del fichero

filelen Devuelve la duración en segundos de un fichero. Su sintaxis es:

```
ivalor  filelen  ifich
```

donde `ivalor` – variable que almacena el valor extraído del bloque de formato del fichero
`ifich` – nombre del fichero

filenchnls Devuelve el número de canales de audio de un fichero. Su sintaxis es:

```
ivalor  filenchnls  ifich
```

donde `ivalor` – variable que almacena el valor extraído del bloque de formato del fichero
`ifich` – nombre del fichero

filesr Devuelve la frecuencia de muestreo de un fichero. Su sintaxis es:

```
ivalor  filesr  ifich
```

donde `ivalor` – variable que almacena el valor extraído del bloque de formato del fichero
`ifich` – nombre del fichero

Escritura de valores y resultados:

Csound posee operadores para escribir valores y textos en la consola (como **print**, **printk** o **printks**) o en un fichero (**fprints** o **fprintks**). Los operadores que terminan en 's' pueden utilizar la misma cadena que en el lenguaje C para definir el formato de salida. A continuación se muestran algunos de los formatos posibles:

Formato	Salida	Ejemplo	Valor	Escribe
%d	Entero con signo	%6d	1234	□□1234
%f	Real en coma flotante (float)	%5.2f	12.345678	□12.35
%s	Cadenas de caracteres	%s	"cadena"	cadena

Además, con **\t** escribirá un tabulador y con **\n** escribirá un salto de línea.

print Escribe en la consola de salida los valores de tipo i- que se le suministran. Se hace una sola vez, en el momento de la inicialización del instrumento. **prints** permite usar una cadena de formato de `printf()`. Su sintaxis es:

```
print  ivalor1 [, ivalor2] [,...]
```

printk Escribe en la consola de salida el valor de la variable de tipo k- que se le suministra. Lo hace cada cierto tiempo, especificado por *itime*. Su sintaxis es:

```
printk  itime, kvalor
```

printk2 Escribe en la consola de salida el valor de la variable tipo k- que se le suministra cada vez que cambie de valor. Su sintaxis es:

```
printk  itime, kvalor
```

printks Escribe valores de variables de tipo k– usando los formatos de C. **fprintks** hace lo mismo, pero imprimiendo en un fichero. Su sintaxis es:

```
fprintks  "nom_fichero", "cadena_formato" [, kvalor1] [, kvalor2] [,...]
```

Cualquier manual de C proporcionará información sobre la cadena de formato al describir su **printf**.

Generadores de señales:

lfo Oscilador de baja frecuencia con varias formas de onda. Su sintaxis es:

```
xres  lfo  kamp, kfrec [, itipo]
```

donde `kamp` – amplitud
`kfrec` – frecuencia
`itipo` – (opcional) determina la forma de onda: 0 = seno, 1 = triangular, 2 = cuadrada unipolar, 3 = cuadrada bipolar, 4 = rampa, 5 = diente de sierra. El valor por defecto es 0.

loscil Lee un sonido muestreado almacenado en una tabla, con bucles opcionales en el sostenimiento.

```
ares  loscil  xamp, kcps, ifn [, itrans]
```

donde `xamp` – valor de amplitud
`kcps` – determina la transposición de la frecuencia fundamental de la señal de la tabla
`ifn` – número de la tabla de onda que contiene una nota digitalizada
`itrans` – relación de transposición de la nota. Si se desconoce úsese 1 aquí y para `kcps`

mp3in Lee audio estéreo de un fichero MP3. Su sintaxis es:

```
as1,as2  mp3in  ifich
```

donde `as1,as2` – variables de audio en las que se almacenan los canales leídos del fichero MP3.
`ifich` – nombre del fichero

oscil / oscili / oscil3 Produce señales periódicas repitiendo el ciclo de una forma de onda de una tabla. **oscili** y **oscil3** devuelven valores mediante interpolación lineal o cúbica, respectivamente, con la misma sintaxis:

```
ares  oscil  xamp, xcps, ifn
```

donde `xamp` – amplitud
`xcps` – frecuencia fundamental de la señal generada
`ifn` – número de la tabla que contiene un ciclo de la onda a generar

oscils Genera ondas sinusoidales. Su sintaxis es:

```
ares  oscil  iamp, icps, ifase
```

donde `iamp` – amplitud
`icps` – frecuencia
`ifase` – fase inicial de la onda, entre 0 y 1 (= 2π)

phasor Genera una fase normalizada ($0 \leq kfase < 1$) según la frecuencia *kfrec* que se le indica. Su sintaxis es:

```
kfase  phasor  kfrec
```

soundin Lee un sonido almacenado en un fichero. Su sintaxis básica es:

```
ares  soundin  ifilcod
```

donde `ifilcod` – entero o cadena de caracteres con el nombre del fichero del que leer la muestra. Si es un entero, se leerá el fichero "`soundin.<ifilcod>`". Si es cadena, se pondrá entre dobles comillas.

vco Genera ondas mediante un simulador de un oscilador controlado por tensión:

```
ares  vco  kamp, kfrec, itipo, ipw, itabla, imaxd
```

donde `kamp` – amplitud
`kfrec` – frecuencia fundamental
`itipo` – tipo de onda: = 1: diente de sierra, = 2: cuadrada o modulada en anchura de pulso (PWM), = 3: diente de sierra / triangular / en rampa
`ipw` – entre 0 y 1. Sin efecto si *itipo* = 1; cuando *itipo* = 2 determina la anchura de pulso (se modula con LFO para conseguir PWM, *ipw* = 0.5 será cuadrada); si *itipo* = 3 controla si es una sierra (*ipw* = 0), triangular (*ipw* = 0.5) o rampa (*ipw* = 1).
`itabla` – tabla que contendrá un ciclo de onda sinusoidal.
`imaxd` – tiempo máximo de retraso. = 1/*kfrec* es necesario para PWM y triangular.

vco2 Versión simplificada del anterior, que usa tablas precalculadas:

```
ares   vco2   kamp, kfrec [, imodo]
```

donde `kamp` – amplitud
`kfrec` – frecuencia fundamental
`imodo` – tipo de onda: = 0 (o no se pone): diente de sierra, = 6: pulso, = 10: cuadrada, 12 = triangular.

table / tablei / table3 Devuelve un valor almacenado en una tabla, según un índice dado como entrada. **tablei** es como **table** pero si el índice no es entero interpola el valor entre sus vecinos (**table3** con cúbica). Su sintaxis es:

```
xres  table  xindice, itabla, imodo, ioffset
```

donde `xindice` – entrada con el valor del índice a buscar en la tabla (tipo i-, k- o a-).
`itabla` – número de la tabla a leer.
`imodo` – 0 significa un índice sin modificar y 1 un índice normalizado (entre 0 y 1).
`ioffset` – (opcional) elemento de la tabla en el que se sitúa el cero del índice de lectura. Para una tabla con el cero en el centro, debe valer tamaño/2 (para índices sin modificar) o 0.5 (para índices normalizados). El valor por defecto es 0.

Análisis de Fourier y procesamiento espectral:

pvsanal Calcula el espectrograma *fspec* de una señal *asig* mono. Su sintaxis es:

```
fspec  pvsanal  asig, ifftsize, isolapa, ivensize, iventipo
```

donde `ifftsize` – Tamaño de la transformada de Fourier en muestras. Mejor que sea potencia de 2.
`isolapa` – Salto de la ventana en muestras para el cálculo del espectrograma, p.ej. *ifftsize*/4.
`ivensize` – Tamaño de la ventana en muestras. Normalmente = *ifftsize*.
`iventipo` – Forma de la ventana. 0 = Hamming, 1 = von Hann.

pvscent Calcula el centroide de un espectro. Su sintaxis es:

```
kcent  pvscent  fsig
```

donde `fsig` – espectrograma previamente calculado con un operador **pvsanal**.
`kcent` – el centroide espectral del espectro que corresponda al instante en el que se haga el cálculo (esta familia de operadores sigue su propia temporización que depende del avance de la ventana de análisis).

pvscale Escala las frecuencias de un espectro *fin* de acuerdo con un factor. Su sintaxis es:

```
f_out  pvscale  f_in, ktransp
```

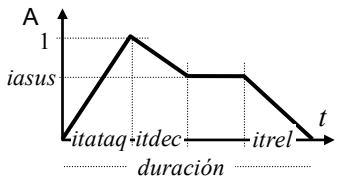
donde `f_in` – Espectrograma a procesar.
`ktransp` – Factor de escala que se aplicará al espectro = Relación de transposición.

pvsynth Reconstruye una señal a partir de un espectrograma *fin*. Su sintaxis es:

```
ares  pvsynth  f_out
```

Generadores de señales de control:

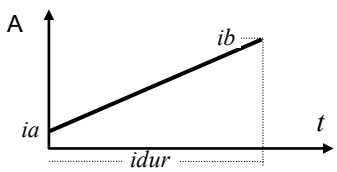
adsr Genera una curva como la mostrada en la figura.




```
xres  adsr  itataq, itdec, iasus, itrel
```

donde `itataq` – tiempo de ataque en segundos
`itdec` – tiempo de caída en segundos
`iasus` – amplitud de sostenimiento entre 0 y 1. Su duración depende de la nota.
`itrel` – tiempo de relajación en segundos

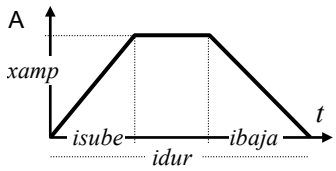
line Permite calcular segmentos lineales entre dos valores.




```
kres  line  ia, idur1, ib
```

donde `ia` – valor inicial (> 0)
`idur1` – duración del segmento
`ib` – valor final

linen Genera una curva tipo ASR como la mostrada en la figura.




```
xres   linen   xamp, isube, idur, ibaja
```

donde `xamp` – valor de amplitud
`isube` – tiempo de ataque en segundos (si = 0 no existirá fase de ataque)
`idur` – duración total
`ibaja` – tiempo de relajación (si *ibaja* > *idur* la caída resultará truncada)

Si *isube* + *ibaja* > i*dur*, la línea se curvará durante el tiempo que dure el solapamiento de ambas fases. Si *xamp* es una señal de audio, la curva se aplica sobre su amplitud directamente, comportándose en este caso como un modificador.

Modificadores de señales:

delay1 Retrasa una señal en una muestra (en un tiempo equivalente a 1 período de muestreo). Si la de entrada es $x[n]$, la de salida sería $x[n–1]$. Su sintaxis es:

```
asal   delay1   asenal
```

donde `asenal` – señal de entrada
`asal` – señal retrasada

delay Retrasa una señal en un tiempo t_D especificado en segundos. Si la de entrada es $x[n]$, la de salida sería $x[n–t_D×f_s]$. delay1 *ain* es equivalente a **delay** *ain*, 1/*sr*. Su sintaxis es:

```
asal   delay   asenal, itdelay
```

donde `asenal` – señal de entrada
`asal` – señal retrasada
`itdelay` – tiempo de retraso en segundos

vdelay Aplica un retardo variable a la señal de entrada. Su sintaxis es:

```
ares   vdelay   asig, adel, imaxdel
```

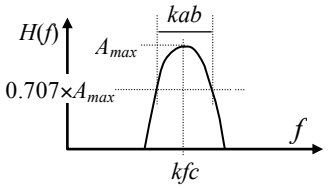
donde `asig` – Señal de entrada al filtro.
`adel` – Valor del retardo en milisegundos.
`imaxdel` – Valor máximo del retardo. Si *adel* lo supera *imaxdel* lo limita.

pconvolve Realiza la convolución de una señal con una respuesta al impulso. Su sintaxis es:

```
ares   pconvolve   asig, ifich, isize [, icanal]
```

donde `asig` – señal de entrada
`ifich` – nombre entre comillas del fichero que contiene la respuesta al impulso
`isize` – tamaño (2^n) de la ventana para la partición para aplicarla. 2048 es un valor adecuado.
`icanal` – si se pone, se procesa el canal del fichero que se indica, si no se procesan todos.

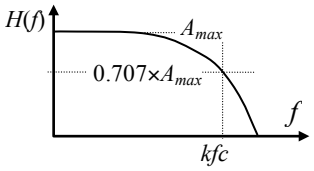
reson Es un filtro pasa-banda IIR de pendiente –12 dB/octava. Su sintaxis es:




```
ares   reson   asig, kfc, kab, inorm
```

donde `asig` – señal de entrada al filtro
`kfc` – la frecuencia de central del filtro (en Hz).
`kab` – el ancho de banda (en Hz) entre las frecuencias en las que la potencia se reduce a la mitad.
`inorm` – factor de escala de la amplitud de salida (= 1, no la altera, = 2 la normaliza)

tone Es un filtro pasa-baja IIR de pendiente –6 dB/octava.. Su sintaxis es:




```
ares   tone   asig, kfc
```

donde `asig` – es la señal de entrada al filtro
`kfc` – la frecuencia de corte (en Hz)

comb Es un filtro peine con realimentación (tipo IIR). Su sintaxis es:

```
ares   comb   asig, kgfb, ilpt
```

donde `asig` – Señal de entrada al filtro.
`kgfb` – Ganancia del lazo de realimentación (controla el tiempo de duración del efecto).
`ilpt` – Tiempo de retardo (controla la densidad y la frecuencia de resonancia, su inversa).

alpass Implementa un filtro pasa-toda. Su sintaxis es:

```
ares  alpass  asig, ktr60, itd
```

donde `asig` – Señal de entrada al filtro.
`ktr60` – Tiempo en segundos de duración del efecto como el que tarda en caer –60 dB.
`itd` – Tiempo en segundos de retardo.

delayr Crea una línea de retardo de *itD* segundos y lee su contenido desde el final de la misma. Debe ir emparejado y preceder a un **delayw**. Su sintaxis es:

```
asal  delayr  itD
```

delayw Escribe una señal al principio de una línea de retardo creada con **delayr**. Su sintaxis es:

```
delayw  asig
```

deltap Lee una señal desde un punto *ktD* (s) de una línea de retardo creada con **delayr**. Su sintaxis es:

```
asal  deltap  ktD
```

Otros:

init Asigna a una variable de control o de audio el resultado de evaluar una expresión de tipo i-. Esta acción sólo se realiza una vez, al principio de la activación del instrumento (en tiempo de inicialización):

```
xvar  init  ivalor
```

donde `xvar` – Es una variable de tipo k- o a-, en este segundo caso rellena todos sus valores con ese valor.
`ivalor` – Es un valor o expresión que se pueda evaluar en tiempo de inicialización del instrumento.

metro Emisor periódico de pulsos. Su sintaxis es:

```
kpulso  metro  kfrec
```

donde `kpulso` – variable que vale siempre 0 menos cuando se dispara el metrónomo, que valdrá 1.
`kfrec` – frecuencia en Hz de disparo de los pulsos.

out , outs Para enviar señales a la salida (mono y estéreo respectivamente). Su sintaxis es:

```
out   aout
outs  aoutL , aoutR
```

pitchamdf Detecta y sigue la frecuencia fundamental y amplitud de una señal monofónica. Su sintaxis es:

```
kfrec,kamp  pitchamdf  asig, ifmin, ifmax
```

donde `asig` – Señal de entrada a analizar.
`ifmin` – Estimación de la frecuencia mínima prevista en la señal.
`ifmax` – Estimación de la frecuencia máxima prevista en la señal.

rms Calcula la envolvente RMS de la señal de entrada *asig*. Su sintaxis es:

```
kres  rms  asig
```

times Devuelve el tiempo en segundos desde el inicio del funcionamiento. Su sintaxis es:

```
kres  times
```

timek Devuelve el número de ciclos de control realizados desde el inicio del funcionamiento. Su sintaxis es:

```
kres  timeinstk
```

timeinstk Devuelve el número de ciclos de control realizados desde cada activación del instrumento. Su sintaxis es:

```
kres  timeinstk
```

Generadores de tablas:

En todos estos operadores, `time` es el tiempo desde el cuál la tabla está disponible (siempre 0 para nosotros) y `size` es el tamaño en muestras de la tabla, que debe ser 2^n salvo que se indique lo contrario en algún caso.

GEN 1 Transfiere el contenido de un fichero de audio a una tabla hasta que se llega al final del fichero o hasta que se llena la tabla. Su sintaxis es:

```
f  numtabla  time  size  1  filcod  skiptime  format  canal
```

`filcod` es un entero que designa un fichero de nombre "soundin.*filcod*" o una cadena entre dobles comillas con el nombre del fichero. Si no se da toda la ruta, el fichero se busca en el directorio actual. `skiptime` permite omitir tiempo (en segundos) del principio del fichero. `format` es para indicar el formato del fichero, el valor 0 indica que se deduzca de su cabecera. `canal` indica el canal del que leer, un 0 hace leerlos todos.

GEN 10 Genera formas de onda mediante una suma ponderada de parciales armónicos sinusoidales.

```
f  numtabla  time  size  10  arm1  arm2  arm3  arm4  . . .
```

`arm`*N* son las amplitudes relativas de los distintos armónicos que intervienen en la construcción de la onda. `arm1` será la amplitud de la fundamental, `arm2` la del segundo armónico, etc.

GEN 23 Genera formas de onda según los valores leídos de un fichero de texto separados por comas, líneas o espacios.

```
f  numtabla  time  size  23  "fichero.txt"
```

ftgen Crea una tabla en la orquesta. Su sintaxis básica es:

```
gitabla  ftgen  itabla, time, size, igen, iarg1, iarg2, . . .
```

donde
`gitabla` – número asignado a la tabla (> 100 si se asigna automáticamente).
`itabla` – número que se quiere para la tabla (si = 0 se asigna automáticamente).
`itime` – tiempo en el que se crea la tabla.
`size` – tamaño de la tabla en muestras.
`igen` – número de rutina GEN usada para crearla.
`iarg1...` – argumentos de la GEN utilizada.

Conversores de unidades:

cpspch(.) : de altura (octava.nota) a Hz
pchmidinn(.) : de altura MIDI a altura (octava.nota)
cpsmidinn(.) : de altura MIDI a Hz

Equivalencia entre las notas de la cuarta octava, la codificación octava.nota de Csound y su frecuencia bien temperada

Nota	DO4	DO#4 REb4	RE4	RE#4 MIb4	MI4	FA4	FA#4 SOLb4	SOL4	SOL#4 LAb4	LA4	LA#4 Sib4	SI4	DO5
altura	8.00	8.01	8.02	8.03	8.04	8.05	8.06	8.07	8.08	8.09	8.10	8.11	8.12=9.00
Hz	261,6	277,2	293,7	311,1	329,6	349,2	370,0	391,5	415,3	440,0	466,2	493,9	523,3
MIDI	60	61	62	63	64	65	66	67	68	69	70	71	72

ampdbfs(.) : de dBFS a PCM (16 bits enteros con signo)
dbfsamp(.) : de dBFS (16 bits enteros con signo) a PCM

Tabla de conversión entre los valores en dBFS, amplitud normalizada y valores de amplitud PCM para 16 bits

dBFS	–54	–48	–42	–36	–30	–27	–24	–21	–18	–15	–12	–9	–6	–3	0
Amp.Norm.	0.002	0.004	0.008	0.016	0.032	0.045	0.063	0.089	0.125	0.17	0.25	0.35	0.5	0.7	1
Amp.PCM	65	130	260	519	1036	1464	2068	2920	4125	5827	8230	11627	16422	23197	32768