Глава 12

«Программирование» музыки

12.1. Общие сведения о C-Sound

Практически все рассмотренные выше музыкальные программы (кроме программ набора и верстки) представляют собой, по сути, набор алгоритмов синтеза звука, его обработки, а также работы со звуковыми последовательностями (то же можно сказать о «некомпьютерных» электронно-музыкальных модулях — синтезаторах, сэмплерах, драм-машинах и т. п.(Их называют «некомпьютерными», хотя каждый из них все равно представляет собой небольшой компьютер, только специализированный, с «намертво» зашитыми программами.). Каждая программа имеет ограниченный набор перечисленных выше алгоритмов, а в них для редакции открыто, как правило, лишь небольшое количество параметров. У многих музыкантов, работавших с этими программами, возникало желание редактировать все параметры алгоритмов и вообще создавать свои собственные. И начинались сетования на ограниченность средств данной программы или системы.

На самом деле все эти ограничения — лишь «плата» за удобство и красоту пользовательского интерфейса. Интерфейс сковывает свободу действий — иначе и быть

не может (сравните, например, «пользовательский интерфейс» профессионального фотоаппарата, имеющий кучу настроек, и обычной «мыльницы» с едва ли не одной кнопкой «щелканья»).

Однако есть программный продукт, позволяющий легко редактировать музыкальные алгоритмы и создавать новые, тонко регулировать мельчайшие параметры каждого звука — просто мечта музыканта. Кроме прочего, он еще и совершенно бесплатный! Речь идет об универсальной системе «программирования» музыки — программе C-Sound.

И пусть людей, далеких от компьютеров и не знающих языков программирования, не пугает вынесенное в заголовок слово «программирование». Вспомните, что и современную стиральную машину нужно немножко «программировать».

Программу C-Sound можно, в принципе, назвать и «языком программирования», однако, поскольку «программируются» здесь музыкальные явления, а не какие-нибудь там «ядра» и «интерфейсы», CSound не просто легок для понимания музыкантов — более того, его легче понять музыканту, не знакомому с программированием, чем программисту, не знакомому с музыкой.

Итак, рассмотрим вкратце основы работы с CSound. Ниже будет рассмотрен «классический» вариант этой программы, не рассчитанный на воспроизведение в реальном времени и вполне сносно работающий даже на

медленных компьютерах (например, 486x с тактовой частотой 33 МГц).

Классическая схема работы в CSound такова. Вы создаете два текстовых файла (в любом текстовом редакторе, например, в стандартном Windows-блокноте Notepad). Первый из них описывает свойства ваших «инструментов», а второй — последовательность звуков, играемых этими инструментами, и их параметры. Грубо говоря, можно сказать, что первый файл — это описание «оркестра» (он должен иметь расширение .orc, и в дальнейшем мы будем называть его orc-файлом), а второй файл — это «партитура» (расширение .sco, будем называть его sco-файлом). Собственно говоря, программа C-Sound занимается тем, что «превращает» эти два файла в обычный звуковой файл, который можно прослушать любой воспроизводящей программой.

В своем базовом варианте C-Sound запускается из командной строки, в которой указываются имена входных файлов, формат и имя выходного файла и еще некоторые параметры. В Windows-варианте имеется небольшая оболочка, которая, однако, принципиально ничем не отличается от командной строки (рис. 12.1).

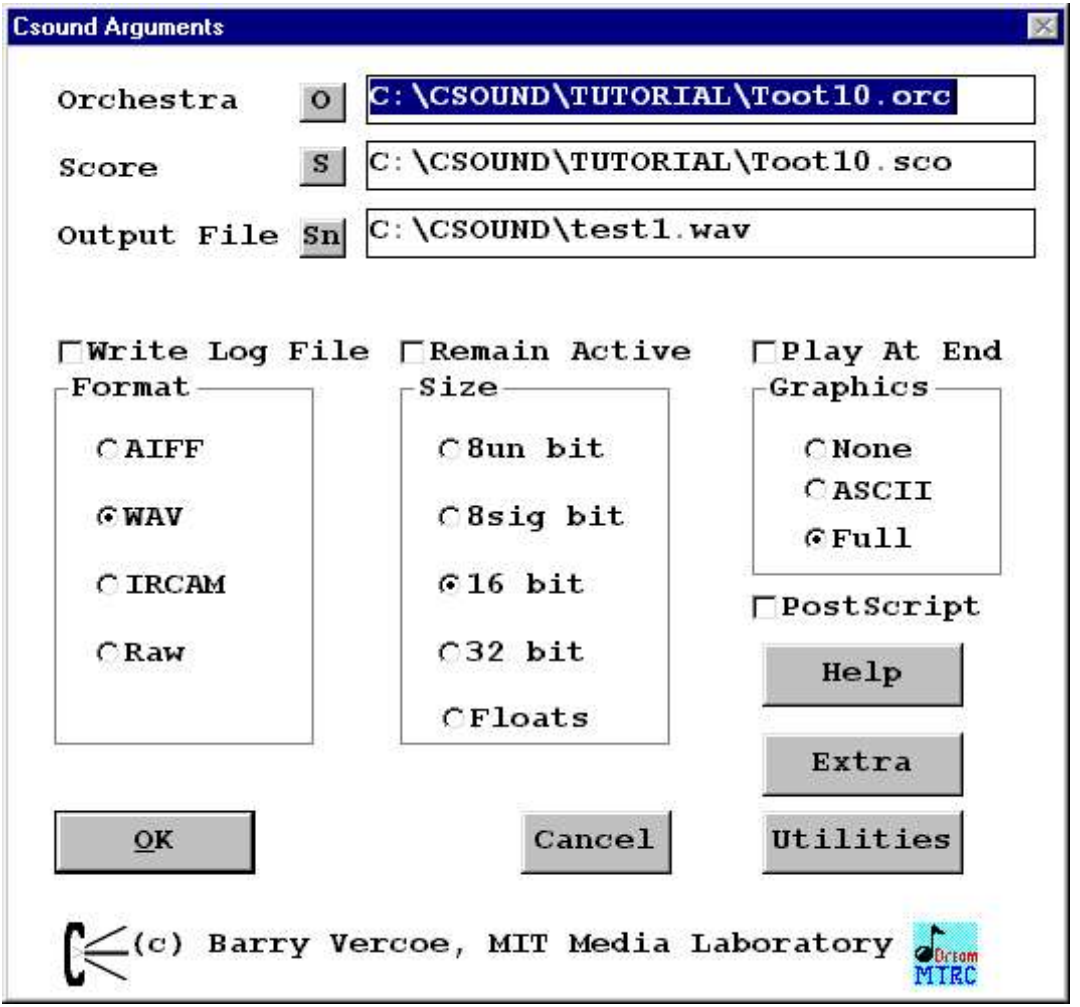

Рис.12.1. Окно оболочки C-Sound для Windows

12.2. Построение простейшего «инструмента»

Рассмотрим вкратце, что же представляют собой текстовые файлы-источники orc и sco и как они превращаются в нечто звучащее.

Как уже говорилось, orc-файл представляет собой описание «инструментария» для вашей музыки. Он может содержать описание одного или нескольких «инструментов», ссылки на которые будут появляться у каждой «ноты» в sco-файле. Описание каждого «инструмента» должно начинаться с ключевого слова instr и заканчиваться словом endin. После слова instr обязательно должен быть указан идентификационный номер этого «инструмента». Описание каждого «инструмента» должно быть завершено словом endin до начала описания следующего инструмента. Описания «инструментов» не могут быть вложены друг в друга:

```
  instr 1
… (описание)
endin
instr 2
… (описание)
endin
instr 3
… (описание)
endin
  и т. д.
```

ВАЛЕРИЙ БЕЛУНЦОВ — фрагмент книги «Компьютер для музыканта»

6-я страница фрагмента

Однако еще до начала описания первого «инструмента» в orc-файле необходимо поместить так называемый заголовок, в котором указываются ключевые параметры для данной композиции. Этих параметров четыре:

- sr — частота вычисления аудиосигнала (собственно говоря, частота дискретизации, sample rate);
- kr — частота вычисления контрольного сигнала (control rate): для ускорения вычислительного процесса некоторые медленно изменяющиеся значения рекомендуется вычислять с этой частотой;
- ksmps — это просто-напросто величина sr/kr. Вообще говоря, загадка, почему не поручили вычисление этой величины компьютеру. Может быть, для проверки способностей пользователя?
- nchnls — количество каналов на выходе; здесь возможны только следующие значения: 1 — моно, 2 — стерео или 4 — квадро.

Вот, например, типичный заголовок orc-файла (для Windows-окружения):

```
sr=44100
kr=3675
ksmps=12
nchnls=2
… (и т. д.)
```

ВАЛЕРИЙ БЕЛУНЦОВ — фрагмент книги «Компьютер для музыканта»

7-я страница фрагмента

После заголовка следуют собственно описания инструментов. Ключевым в описании инструмента является оператор out (для двух каналов — outs). В качестве его операнда<$FОперандом называется необходимый параметр для оператора.> нужно указать объект, который вы как бы «присоединяете» к звуковому выходу — другими словами, то, что звучит. Этим объектом обычно является ссылка на строку с соответствующим оператором. Рассмотрим такой элементарный пример:

```
  sr=44100
kr=3675
ksmps=12
nchnls=1
instr  1
asnd  oscil15000,220,1
out    asnd
endin
```

Этот orc-файл описывает простейший инструмент, который «умеет» играть только одну ноту («ля» малой октавы) на одной громкости, одним тембром (каким — указывается в sco-файле). Здесь строка «out asnd» означает, что выходным результатом данного инструмента является сигнал, который генерируется в строке, помеченной asnd. В этой строке вы можете видеть оператор oscil, один из важнейших в CSound, —

он генерирует периодический сигнал. Оператор oscil требует трех операндов: первый определяет амплитуду сигнала, второй — его частоту и третий — номер волновой таблицы, находящейся в sco-файле. В данном примере амплитуда сигнала равна 15000 единицам, а частота — 220 герцам. Тембр определяется волновой таблицей № 1 (должна быть в sco-файле).

Теперь приведем пример sco-файла для описанного выше простейшего «инструмента»:

```
f1 0 512 10 1
i1 0 1
i1 1 1
i1 3 10
e
```

Если скомпилировать два этих файла, получим в результате звук «ля» малой октавы, исполняемый синусоидальным тембром три раза: два раза по одной секунде и затем, после секундной паузы — 10 секунд.

Первая строка этого sco-файла содержит волновую таблицу, на что указывает буква f в начале строки. Ее цифровые параметры объясняются следующим образом:

- первый параметр (в данном случае 1) — номер таблицы (помните, на этот номер ссылался третий операнд оператора oscil в orc-файле);
- второй параметр (0) — время создания таблицы (в секундах от начала звучания);

- третий параметр (512) — размер таблицы в точках. Волновая форма будет представлена, в сущности, ограниченным количеством точек; чем больше размер таблицы, тем чище, «аккуратнее» звучание, однако тем больше время компиляции. Этот параметр должен быть степенью двойки или превышать ее на единицу;
- четвертый параметр (10) — номер подпрограммы генерации таблицы. 10я подпрограмма (называемая обычно GEN10) генерирует гармонические спектры.
- все последующие параметры зависят от номера подпрограммы генерации. Например, GEN10 требует просто указать относительные амплитуды гармоник, начиная с основного тона. В данном случае генерируется простая синусоида; если бы строка имела вид «f1 0 512 10 1 1 1 1 1», то был бы сгенерирован звук из пяти гармоник, равных по амплитуде, а в случае «f1 0 512 10 5 4 3 2 1» эти пять гармоник убывали бы по амплитуде от нижней к верхней. Заметим, что указывается не абсолютная амплитуда гармоник, а их соотношение: строки «f1 0 512 10 5 4 3 2 1» и «f1 0 512 10 50 40 30 20 10» дадут идентичные результаты.

Каждая строка sco-файла, начинающаяся с буквы i — это «нота», исполняемая инструментам. В этой строке должно быть три обязательных параметра:

- первый параметр — номер инструмента, исполняющего звук (соответствует строке instr в orc-файле);
- второй параметр — время начала звука (измеряется в «долях» от начала звучания, по умолчанию 1 доля равна 1 секунде);
- третий параметр — длина звука.

В строке i могут быть еще и другие параметры, если они необходимы для «инструмента». Однако для приведенного выше простейшего инструмента они не нужны.

Sco-файл должен заканчиваться строкой, содержащей единственную букву e. Кстати, обратите внимание, что в начале каждой строки sco-файла обязательно стоит буква, а больше букв в этом файле нет, есть только цифры.<$FCтрого говоря, буквы могут присутствовать в любой строке sco-файла после знака комментария (точки с запятой). Однако комментарии есть комментарии, они к тексту программы не относятся.>

Буква, стоящая в начале строки, определяет тип этой строки:

- f — функциональная (или волновая) таблица;
- i — звук, исполняемый «инструментом», определенным в orc-файле;
- e — конец файла;
- s — граница секции;

- t — темп в долях в минуту (по умолчанию равен 60, т. е. 1 доля равна 1 секунде). Первым значением t-строки должен быть 0 (то есть для указания темпа 120 долей в минуту нужно написать строку: «t 0 120»).

12.3. Получение необходимого тембра

Разобравшись с простейшим инструментом, попробуем его усложнить. Например, не очень удобно, что инструмент может сыграть только ноту одной высоты. Сделаем так, чтобы высота ноты регулировалась из sco-файла:

orc-файл

```
  sr=44100
kr=3675
ksmps=12
nchnls=1
instr   1
asnd   oscil15000,p4,1
out     asnd
endin
```

sco-файл

```
f1 0 512 10 1
i1 0 1 110
i1 1 1 220
i1 2 1 330
i1 3 1 440
e
```

В этом «инструменте» в операторе oscil также указана постоянная амплитуда, однако вместо постоянной частоты стоит указание «p4», то есть «parameter 4». Оно означает, что значение частоты в герцах будет таким, каким оно указано в четвертом параметре i-строки в sco-файле. Таким образом, теперь мы можем указывать в

sco-файле частоту исполняемого звука. В приведенном выше примере наш инструмент «сыграет» четыре ноты, каждая длиной в 1 секунду: «ля» большой октавы, «ля» малой октавы, чуть повышенное «ми» первой октавы и «ля» первой октавы.

Однако трудно не согласиться, что указывать ноты в виде частоты основного тона не всегда удобно. Поэтому в CSound предусмотрена функция cpspch, преобразующая ноту из более привычного формата «октава.пичкласс» в частоту. При этом октавы нумеруются снизу вверх, а «первая» октава имеет здесь номер 8. Пичкласс ноты обозначается обычным образом: 00 — «до», 01 — «до-диез», 02 — «ре» и т. д., до 11 — «си». Таким образом, «до» первой октавы будет обозначаться как 8.00, «соль» малой октавы — как 7.07 и т. д. В следующем примере наш простой инструмент играет первую фразу мелодии песни «Подмосковные вечера».

orc-файл	sco-файл

```
  sr=44100
kr=3675
ksmps=12
nchnls=1
instr   1
asnd   oscil15000,cpspch(p4),1
out     asnd
endin
```

```
f1 0 512 10 1
i1 0 1 8.00
i1 1 1 8.03
i1 2 1 8.07
i1 3 1 8.03
i1 4 2 8.05
i1 6 1 8.03
i1 7 1 8.02
i1 8 2 8.07
i1 10 2 8.05
i1 12 4 8.00
e
```

Точно таким же образом вы можете указывать амплитуду сигнала в децибелах, используя функцию ampdb, как будет показано в следующем примере.

Настало время сказать несколько слов о переменных. Переменные существуют, грубо говоря, для того, чтобы можно было ссылаться на строки операторов в orc-файле. Например, в приведенных выше примерах строка оператора oscil была помечена как asnd, а оператор out содержал ссылку на эту строку.

Имена переменных в CSound могут состоять из практически любого набора букв и цифр (исключая зарезервированные слова). Однако первой буквой любого имени переменной должна быть либо буква a (как в примерах выше), либо k, либо i. Значения всех

элементов, имя которых начинается с буквы i, вычисляются один раз для каждого звука. Если имя элемента начинается с k, то его значение обновляется с контрольной частотой (control rate), указанной в заголовке orc-файла (kr). Такие имена используются для сигналов, которые изменяются в течение времени звучания одной «ноты», но для которых не критична скорость обновления (огибающие, LFO и пр.). И, наконец, если имя элемента начинается с a, то его значение обновляется с частотой сэмплирования, указанной в заголовке как sr. Эти имена используются для тех сигналов, для которых частота обновления играет ключевую роль — в основном, это аудиосигналы. Пример использования i-переменных приведен ниже (для разнообразия в sco-файле здесь записано начало мелодии «Легко на сердце от песни веселой»). Заметьте, что каждая i-строка sco-файла здесь содержит уже пять параметров — три стандартных, четвертый — громкость и пятый — высота.

orc-файл	sco-файл

```
sr=44100
kr=3675
ksmps=12
nchnls=1
instr   1
ivlm=ampdb(p4)
ipch=cpspch(p5)
asnd   oscil ivlm,ipch,1
out     asnd
endin
```

```
f1 0 512 10 1
i1 0    .5    60    8.03
i1 .5   .5    65    8.07
i1 1    .5    70    8.10
i1 1.5 1      80    9.03
i1 2.5 .75   75    9.02
i1 3.25       .25   65    8.10
i1 3.5 1      75    9.02
i1 4.5 .75   70    9.00
i1 5.25       .25   65    8.08
i1 5.5 1      70    9.00
i1 6.5 2      60    8.10
e
```

Все это, разумеется, хорошо, но в реальной музыке, как правило, громкость и высота звука не являются постоянными на протяжении всей ноты. Заставить параметры звука изменяться во время звучания оного можно, применив к осциллятору (oscil) постоянно меняющиеся значения:

orc-файл

sco-файл

```
  sr=44100
kr=3675
ksmps=12
nchnls=1
instr   1
ivlm=ampdb(p4)
ipch=cpspch(p5)
kenv   linenivlm,p6,p3,p7
asnd   oscilkenv,ipch,1
out     asnd
endin
```

```
f1 0 512 10 1
i1 0    .5    60    8.03 .05   .1
i1 .5   .5    65    8.07 .05   .1
i1 1    .5    70    8.10 .05   .1
i1 1.5  1     80    9.03 .08   .2
i1 2.5  .75   75    9.02 .08   .2
i1 3.25       .25   65    8.10 .02   .1
i1 3.5  1     75    9.02 .06   .2
i1 4.5  .75   70    9.00 .06   .2
i1 5.25       .25   65    8.08 .02   .1
i1 5.5  1     70    9.00 .06   .2
i1 6.5  2     60    8.10 .09   .3
e
```

В этом примере в качестве амплитуды в операторе oscil указана не постоянная величина ivlm, а динамически изменяющаяся kenv (начинается с k!). Данной меткой обозначена строка оператора linen, строящего огибающую для нашего звука. Оператор linen требует наличия четырех операндов: максимальной амплитуды, времени атаки, общей продолжительности звука и времени затухания. В качестве значения максимальной амплитуды мы берем ivlm, а общая продолжительность звука всегда находится в параметре p3 (помните — это обязательный для всех i-строк стандартный параметр). Что касается времени атаки и времени затухания, для них мы здесь определили специальные параметры p6 и p7, это и отразилось в соответствующем sco-файле.

ВАЛЕРИЙ БЕЛУНЦОВ — фрагмент книги «Компьютер для музыканта»

17-я страница фрагмента

Дополним картину динамически изменяющихся параметров добавлением легкого вибрато в «песню». Для этого к постоянной базовой высоте ноты ipch в операторе oscil добавим медленно изменяющийся синусоидальный сигнал kvbr. При этом предоставим возможность выбирать в sco-файле частоту и глубину вибрато, а также время его задержки. Задержку вибрато осуществим с помощью постепенно возрастающей амплитудной огибающей на генераторе «медленной синусоиды»:

orc-файл

```
  sr=44100
kr=3675
ksmps=12
nchnls=1
instr   1
ivlm=ampdb(p4)
ipch=cpspch(p5)
kenv  linenivlm,p6,p3,p7
kvdel linseg      0,p9,0,.1,p10,p3-p9-.1,p10
kvbr   oscilkvdel,p8,2
asnd   oscilkenv,ipch+kvbr,1
out     asnd
endin
```

sco-файл

```
f1 0 512 10 3 2 1
f2 0 512 10 1
i1 0 .5 60 8.03   .05 .1 5 .1 5
i1 .5 .5 65 8.07 .05 .1 6 .1 7
i1 1 .5 70 8.10   .05 .1 5 .1 7
i1 1.5 1 80 9.03 .08 .2 7 .3 11
i1 2.5 .75 75 9.02 .08 .2 5 .2 7
i1 3.25       .25 65 8.10 .02 .1 5  .1 7
i1 3.5 1 75 9.02 .06 .2 6 .25 11
i1 4.5 .75 70 9.00 .06 .2 6 .2 7
i1 5.25       .25 65      8.08 .02 .1 5 .1 7
i1 5.5 1 70 9.00 .06 .2 5 .2 7
i1 6.5 2 60 8.10 .09 .3 6 .4 8
e
```

Для амплитудной огибающей на «медленной синусоиде» был использован генератор непериодического сигнала, состоящего из линейных сегментов linseg. При этом операторе может находиться любое нечетное число операндов. Все операнды, стоящие на нечетных позициях, обозначают уровень сигнала, а стоящие на четных позициях — время перехода между уровнями. В нашем примере («linseg 0,p9,0,.1,p10,p3-p9-.1,p10»): 0 — уровень, p9 — время перехода, 0 — уровень, .1 — время перехода, p10 — уровень, p3–p9–.1 — оставшееся время, и p10 — последний уровень (если ваш сигнал будет состоять всего из одного сегмента, вместо

оператора linseg используйте оператор line, это намного ускорит процесс компиляции).

Теперь в качестве иллюстрации возможных улучшений алгоритма добавим к нашему «инструменту» небольшой хорус:

orc-файл

```
  sr=44100
kr=3675
ksmps=12
nchnls=1
instr   1
ivlm=ampdb(p4)/5
ipch=cpspch(p5)
kenv   linenivlm,p6,p3,p7
kvdel linseg      0,p9,0,.1,p10,p3-p9-.1,p10
kvbr   oscilkvdel,p8,2
asnd1 oscilkenv,ipch+kvbr,1
asnd2 oscilkenv,ipch*1.002+kvbr,1
asnd3 oscilkenv,ipch*1.004+kvbr,1
asnd4 oscilkenv,ipch*.998+kvbr,1
asnd5 oscilkenv,ipch*.996+kvbr,1
asnd=asnd1+asnd2+asnd3+asnd4+asnd5
out     asnd
endin
```

sco-файл

```
f1 0 512 10 3 2 1
f2 0 512 10 1
i1 0 .5 60 8.03   .05 .1 5 .1 5
i1 .5 .5 65 8.07 .05 .1 6 .1 7
i1 1 .5 70 8.10   .05 .1 5 .1 7
i1 1.5 1 80 9.03 .08 .2 7 .3 11
i1 2.5 .75 75 9.02 .08 .2 5 .2 7
i1 3.25       .25 65 8.10 .02 .1 5  .1 7
i1 3.5 1 75 9.02 .06 .2 6 .25 11
i1 4.5 .75 70 9.00 .06 .2 6 .2 7
i1 5.25       .25 65      8.08 .02 .1 5 .1 7
i1 5.5 1 70 9.00 .06 .2 5 .2 7
i1 6.5 2 60 8.10 .09 .3 6 .4 8
e
```

Для получения этого хоруса мы использовали пять почти одинаковых сигналов, только чуть-чуть различающихся частотой. Обратите внимание, что, поскольку перед выходом эти сигналы складываются, указанная амплитуда при инициализации уменьшается в пять раз.

12.4. Состав операторов и возможности C-Sound

Если вы дочитали до этого места и поняли смысл приведенных выше простых файлов CSound, то вы легко сможете строить алгоритмы и дальше, вооружившись собственной фантазией. Поэтому теперь я лишь кратко

перечислю основные операторы CSound, а также некоторые другие его возможности.

Итак, начнем. В C-Sound можно определить глобальные переменные, имена которых должны начинаться с буквы g. После буквы g должна следовать бука a, k или i, а потом — любой набор символов. Глобальная переменная отличается от локальной тем, что область ее действия распространяется на весь orc-файл (область действия обычных, локальных переменных — один «инструмент», т. е. ограничивается пространством между instr и endin). Глобальные переменные полезны для создания «инструментов», обрабатывающих сигнал, — реверберatoров, эквалайзеров и т. п.

В CSound можно использовать арифметические операции сложения (+), вычитания (–), умножения (*) и деления (/). Возможны также логическое И (&&) и логическое ИЛИ (||), но они не применимы к a-параметрам, обновляемым с частотой сэмплирования (sr). Порядок действий при этом обычный: сначала умножение и деление, потом сложение и вычитание, а затем логические операции. Для указания иного порядка действий можно использовать скобки.

Можно применять также условные функции, которые возвращают то или иное значение в зависимости от выполнения условия. Например, выражение a>b?c:d возвращает значение c, если a действительно больше, чем b; в противном случае возвращается значение d. В качестве операций сравнения могут использоваться:

больше (**>**), меньше (**<**), больше или равно (**>=**), меньше или равно (**<=**), равно (**==**) и не равно (**!=**).

Существует небольшой набор встроенных функций:

- ftlen(x) возвращает размер таблицы с номером x;
- int(x) возвращает целую часть числа x;
- frac(x) возвращает дробную часть числа x;
- dbamp(x) преобразует линейное значение амплитуды в децибелы;
- ampdb(x) преобразует значение амплитуды в децибелах в линейное значение;
- i(x) преобразует k-элемент или a-элемент в i-элемент;
- abs(x) возвращает абсолютное значение числа x;
- exp(x) возводит число e в степень x;
- log(x) возвращает натуральный логарифм числа x;
- sqrt(x) извлекает из числа x квадратный корень;
- sin(x) возвращает синус числа x (значение x указывается в радианах);
- cos(x) возвращает косинус числа x (значение x указывается в радианах);
- octpch(x) преобразует пич-класс в десятичное значение октавы/ноты;
- pchoct(x) преобразует десятичное значение октавы/ноты в пич-класс;

- cpspch(x) преобразует пич-класс в частоту основного тона;
- octcps(x) преобразует частоту в десятичное значение октавы/ноты;
- cpsoct(x) преобразует десятичное значение октавы/ноты в частоту.

В CSound можно применять операции ветвления (перехода). Для безусловного ветвления можно использовать оператор goto, после которого должна стоять метка. Операторы igoto и kgoto используются для ветвления, соответственно, только во время «инициализации» звука или только во время обновления значений. Для условного ветвленеия можно использовать оператор if … goto, где после оператора if должна быть операция сравнения, например: if k2>p4 goto next. Здесь сравниваются значения k2 и p4, и если значение k2 больше, то происходит ветвление на метку next.

Для условного ветвления можно использовать также оператор timout. Это ветвление в зависимости от времени, прошедшего с момента начала звука. Оператор требует трех операндов: времени начала ветвления, продолжительности временного интервала ветвления и метки. Например: timout 1, 2, next. Здесь ветвление происходит в том случае, если от начала звука прошло более 1 и менее 3 секунд (точнее, долей).

Во всех приведенных выше примерах для генерации периодических сигналов использовался оператор oscil.

Однако CSound содержит еще целый набор генераторов периодических (и непериодических) сигналов. Рассмотрим некоторые из них.

Оператор foscil предназначен для FM-синтеза. Он требует шести операндов: амплитуда, базовая частота, коэффициент несущей частоты<$FCтрого говоря, надо употреблять выражение «частота несущего оператора», а не несущая частота. Но для упрощения мы используем тут принятые в обиходе выражения.>, коэффициент модулирующей частоты, индекс модуляции, номер таблицы. Амплитуда и номер таблицы здесь по значению идентичны соответствующим операндам оператора oscil. Несущая и модулирующая частоты получаются от умножения коэффициентов на базовую частоту. Например, если базовая частота равна 220 Гц, а коэффициенты — 1 и 2, то в результате при FM-синтезе несущая частота будет равна 220 Гц, а модулирующая — 440 — Гц.

Оператор loscil предназначен для WT-синтеза, то есть использования CSound в качестве сэмплера. Он требует наличия трех обязательных операндов, после которых могут следовать семь или менее необязательных. Обязательными являются амплитуда, частота и номер волновой таблицы (имеет смысл выбирать таблицу, заполненную волновой формой сэмплированного звука). Затем могут следовать: основная частота оригинального сэмпла (если этот операнд равен 0 или отсутствует, то данное значение берется из заголовка звукового файла-источника), режим петли (0 — нет петли, 1 — обычная

петля, 2 — двустороння петля), точка начала петли, точка конца петли. Потом могут следовать те же параметры для второй петли.

А каким же образом сэмплированный звук из файла-источника попадает в таблицу? Для этого существует несколько способов. Один из них — воспользоваться подпрограммой GEN01, например, вот так: «f1 0 8192 1 "piano_c3.aiff" 0 4 0». Как вы уже, вероятно, догадались по первым четырем параметрам этой f-строки, здесь вызывается подпрограмма генерации сигнала GEN01. После номера подпрограммы должны быть указаны следующие параметры: имя файла-источника в двойных кавычках, точка (время) начала чтения в звуковом файле, формат звукового файла (4 означает обычный 16-битный формат) и номер канала, который следует прочитать из файла (для стерео- или квадро-файлов; 0 означает чтение всех каналов).

Оператор pluck используется при синтезе методом Карплуса — Стронга для получения звука щипка струны или ударных звуков. Напомню, что в основе этого метода лежит периодическое повторение первоначального (обычно шумового) посыла с частотой, лежащей в звуковом диапазоне. Этот оператор требует наличия пяти обязательных операндов: амплитуда, частота повторения посыла, базовая частота самого посыла, номер волновой таблицы (для инициализации посыла) и номер метода синтеза. Номер метода может принимать значения от 1 до 6, при этом для использования алгоритма Карплуса — Стронга «в чистом виде» нужно

применить метод № 1. Применение других методов (кроме № 6) требует ввода дополнительных параметров. Например, при выборе метода № 3 (звуки ударного характера) следует ввести еще «коэффициент жесткости» звучания, лежащий в диапазоне от 0 до 1. Поскольку в «классическом» варианте алгоритма Карплуса — Стронга используется шумовой посыл, в качестве номера волновой таблицы можно указать 0, и в этом случае для посыла будет использована случайная последовательность. Базовая частота посыла обычно должна равняться частоте повторения и соответствовать желаемой высоте звука.

Оператор rand предназначен для генерации случайных последовательностей, которые могут использоваться, например, для создания шумовых сигналов. Этот оператор требует единственный операнд — амплитуду генерируемых значений. Новые значения последовательности генерируются с частотой sr или kr. В некоторых случаях может потребоваться генерировать случайные значения с меньшей частотой. Для этого предназначен оператор randh, которому необходим второй операнд — частота генерации случайных значений. Можно использовать также оператор randi, который действует так же, как и randh, однако осуществляет линейную интерполяцию между сгенерированными значениями.

Оператор fof предназначен для формантного синтеза. Он воспроизводит основные алгоритмы знаменитой IRCAM’овской программы CHANT<$FB основе алгоритмов

CHANT и оператора fof просто-напросто лежит один и тот же программный код.>. Детально его объяснять довольно долго, поэтому приведу лишь список обязательных операндов этого оператора:

- максимальная амплитуда импульсов;
- базовая частота следования импульсов;
- центральная частота формантной области (она же частота сигнала в каждом импульсе);
- октавный индекс (обычно равен 0);
- ширина формантной области (в герцах);
- время атаки импульса (типичное значение для имитации голоса — 0.003);
- продолжительность импульса (типичное значение для имитации голоса — 0.02);
- время затухания импульса (типичное значение для имитации голоса — 0.007);
- количество зарезервированного места в памяти для сохранения информации от «перекрывающихся» импульсов;
- номер таблицы с волновой формой импульса (рекомендуется синусоида в таблице размером не менее 4096);
- номер таблицы с возрастающим сигналом (используется как огибающая для атаки и затухания импульса). Рекомендуется линейный сигнал. Для его

получения используйте подпрограмму GEN07, действующую аналогично рассмотренному ранее оператору linseg;

- общее время «активности» fof, обычно равное p3.

Оператор grain применяется для гранулярного синтеза. Как и в предыдущем случае, ограничусь перечислением его обязательных операндов:

- общая амплитуда звука;
- частота данного «зерна»;
- плотность (частота следования «зерен»);
- величина максимального отклонения от заданной амплитуды звука;
- величина максимального отклонения от заданной частоты звука;
- длина данного «зерна»;
- номер таблицы с волновой формой «зерна» (обычно там содержится синусоида или сэмплированный звук);
- номер таблицы, содержащей огибающую «зерна»;
- максимальное значение длины «зерна».

Оператор vdelay применяется для обработки сигнала с помощью задержки. Он требует трех операндов: исходного сигнала, текущего времени задержки и максимального значения времени задержки. Обратите внимание, что текущее время задержки лучше всего

контролировать с частотой sr, иначе могут появиться непредвиденные щелчки.

Оператор reverb применяется для реверберации исходного сигнала. Он требует наличия двух обязательных операндов — исходного сигнала и времени реверберации. Для использования других типов реверберации существуют операторы comb и alpass, которые требуют наличия еще одного дополнительного операнда — «размера временной петли», практически означающего плотность эха.

Оператор multitap применяется для обработки сигнала с помощью мульти-задержки. В качестве первого операнда здесь должен быть указан исходный сигнал, а затем могут следовать несколько пар операндов. В каждой паре первый из операндов определяет время задержки, а второй — уровень задержанного сигнала.

Существует и множество других операторов CSound, позволяющих осуществить самые фантастические звуковые мечты, но подробно их перечислять не буду. Если вы научитесь эффективно применять рассмотренные выше возможности CSound (а рассмотрены все основные операторы и функции), то остальные операторы и подпрограммы генерации сможете изучить самостоятельно, заглянув в описание CSound, свободно лежащее в Сети (на английском языке) или просто в файл csound.hlp, присутствующий практически во всех версиях и вариантах программы.

ВАЛЕРИЙ БЕЛУНЦОВ — фрагмент книги «Компьютер для музыканта»

31-я страница фрагмента

Ну, и напоследок замечу, что все, что относится к программе CSound (и саму программу) можно найти в Интернете на «The Csound Front Page» по адресу http://www.leeds.ac.uk/music/Man/c_front.html.