Especificação da Linguagem

Grupo 12, LFA

Maio 2018

Conteúdo

1 Estruturas de Dados (Tipos de Variáveis)

1.1 Sequências

Uma sequência de notas e silêncios pode ser definida, através da palavra chave `sequence`, como:

```
1 sequence melody = [C C G G A A G R F F E E D D C R]; // os espacos
      sao opcionais
```

1.1.1 Notas musicais

A notas são identificadas pela sua letra:

- `C` para Dó,
- `D` para Ré,
- `E` para Mi,
- `F` para Fá,
- `G` para Sol,
- `A` para Lá,
- `B` para Si.

A letra `R` é usada para silêncios (derivada de *Rests*).

Um cardinal (`#`) a seguir à letra sobe a respetiva nota por meio tom; um b minúsculo (`b`) a seguir à letra desce a respetiva nota por meio tom.

Pode, depois do tom, aparecer um número, entre 1 e 8. Este especifica a oitava a usar, sendo 1 a mais grave e 8 a mais aguda. Por omissão, a quarta oitava é usada.

Assim, a sequência já apresentada podia reescrita da seguinte forma:

```
sequence melody = [C4 B# G G4 A4 A G R F E#4 Fb4 E D D4 C4 R];
```

1.1.2 Duração duma nota

Por omissão, uma nota demora um tempo[1]. A sua duração depende do *tempo*, ou `BPM`[2], da música. Esta configuração é explorada na secção 5.

Uma nota pode, no entanto, tocar mais ou menos tempo. Há duas formas de especificar a duração duma dada nota:

1. **Por extensão.** A duração da nota é especificada através de chavetas ({ e }), relativamente à duração unitária. Por exemplo, `C{4}` demora o quádruplo do tempo de `C`, e `C{0.75}` demora três quartos do tempo de `C`.

2. **Notação simplificada.** Utilizam-se apóstrofos para reduzir a duração duma nota em metade, havendo uma correspondência direta com a duração das notas musicais convencionadas [3]. Por exemplo, `C'` demora metade do tempo de `C`, e `C''` apenas um quarto.

1.1.3 Acordes

O símbolo `|` é usado para tocar várias notas em simultâneo, numa só sequência. Para tocar o acorde de Dó maior (C, E e G), na quarta oitava, podíamos então escrever:

```
sequence intro = [C|E|G];
```

1.2 Performances

A associação duma sequência musical com um instrumento representa um terceiro tipo de dados, uma `performance`:

```
sequence twinkle = [CC GG AA G{2} FF EE DD C{2}];
performance p = twinkle on guitar;

// ou, alternativamente, definindo a sequencia implicitamente
performance p = [CC GG AA G{2} FF EE DD C{2}] on guitar;
```

[1] Em termos musicais, uma semínima.

[2] *Beats Per Minute*

[3] Semínima (duração de 1), colcheia (duração de 1/2), semicolcheia (duração de 1/4), etc.

1.3 Marcos de Tempo

Um marco de tempo é uma variável que referencia um dado momento entre o início e o fim, inclusivé, da peça musical. `start` é um marco de tempo pré-definido, representando o início da peça.

1.3.1 Funções auxiliares

A função auxiliar `duration`, aceita um parâmetro, do tipo sequence ou performance, e devolve um time igual à sua duração.

```
sequence intro = [R{4} C{4} G{4} C5{3.5} E|G|C5{.5} Eb|G|C5{8} C{4} G{4}]; // Strauss - Also Sprach Zarathustra - Intro (https://www.8notes.com/scores/7213.asp)
time endIntro = start + duration(intro);
```

1.4 Inteiros

Inteiros também são suportados:

```
int octave = 4;
```

1.5 Arrays (TODO)

Um array é uma coleção de várias instâncias da mesma estrutura de dados. Suportam-se arrays de sequências, instrumentos e performances.

2 Geração de aúdio

2.1 Instrumentos

Para tocar sequência musical é, naturalmente, necessário especificar que instrumento deve ser utilizado. Assim, para tocar *Twinkle, Twinkle, Little Star* com uma piano, teríamos:

```
// definir a sequencia
sequence twinkle =  [CC GG AA G{2} FF EE DD C{2}];

// utilizando performances
performance p = twinkle on piano;
play p;

// ou, alternativamente, definindo a performance implicitamente
play twinkle on piano;

// ou definindo a performance e a sequencia implicitamente
play [CC GG AA G{2} FF EE DD C{2}] on piano;
```

Vários instrumentos podem ser usados para tocar uma dada sequência. No entantom nem todos suportam o mesmo registo (por exemplo, um piano suporta uma maior gama de notas que um violino). Se a um instrumento é dada uma

sequência que este não suporta, as notas não suportadas são *aparadas*, sendo substituídas pela nota mais próxima que é suportada.

Os instrumentos disponíveis são os seguintes:

- `piano`;
- `guitar`;
- `violin`;
- `cello`;
- `bass`;
- `drums`.

A criação de novos instrumentos é suportada, estando detalhada na secção 5.

2.2 Modos de reprodução

No exemplo anterior, não foi especificado quando começar a tocar a sequência. Por omissão, a sequência começa a ser tocada no início da peça (por outras palavras, no tempo 0). Este tempo também pode ser obtido através da palavra chave `start`.

Averiguemos os diferentes modos de reprodução:

2.2.1 Simultâneo

Por omissão, todas as sequências são tocadas começando no tempo 0, ou `start`. Se há mais que uma sequência a ser tocada, todas as sequências são tocadas em paralelo.

```
play [CC GG AA G{2} FF EE DD C{2}] on piano;

// e equivalente, em termos do som produzido no ficheiro final, a
play [CC RR RR G{2} FR RE DR C{2}] on piano;
play [RR GG AA RR RF ER RD RR] on piano;
```

Um outro exemplo de reprodução simultânea é o seguinte, que separa melodia e harmonia em duas performances diferentes:

```
// definir sequencias
sequence twinkle = [CC GG AA G{2} FF EE DD C{2}];
sequence twinkle_bass = [C|E|G{2} F|A|C{2} C|E|G{4} F|A|C C|E|G G|B
    |D C|E|G];

// tocar performances
play twinkle on guitar;
play twinkle_bass on guitar;
```

2.2.2 Sequencial

A palavra chave `after` indica que uma dada performance deve começar imediatamente após o fim doutra.

```
// definir sequencias
sequence first_line = [CC GG AA G{2}];
sequence second_line = [FF EE DD C{2}];

// definir performances
performance first_line = twinkle on guitar;
performance second_line = twinkle_bass on guitar;

// tocar performances
play first_line;
after first_line play second_line;
```

Num exemplo mais avançado, onde a sequência de referência (no exemplo acima, `first_line`) passada a after é tocada mais que uma vez, pode ser especificado após que performances deve a sequência alvo (no exemplo acima, `second_line`) ser tocada. Por omissão, a sequência alvo é tocada apenas 1 vez, após a primeira reprodução da sequência de referência.

Para obter outros comportamentos, a palavra chave `always` pode ser utilizada.

```
// definir sequencias
sequence first_line = [CC GG AA G{2}];
sequence second_line = [FF EE DD C{2}];

// definir performances
performance first_line = twinkle on guitar;
performance second_line = twinkle_bass on guitar;

// tocar performances
play first_line;
after first_line always play second_line;
after second_line play first_line;
// toca first_line (FL), seguido de second_line (SL), e repete uma
    vez, ou seja, FL, SL, FL, SL
```

2.2.3 Usando Marcos de Tempo

A palavra chave `at` específica um Marco de Tempo específico no qual a performance deve começar, independentemente de haver outras performances a decorrer nesse momento.

```
performance verse = [CC E{2} GG B{2} C5C5 GG C{4}] on violin;
performance chorus = [GAGA ABBA GEGE EBBE] on violin;

time chorusStart = start + 2*duration(verse);

// tocar performances
play verse;
play verse;
at chorusStart play chorus;
```

Uma performance pode ser tocada em mais que um momento. Para isso, além da palavra chave `at`, utiliza-se também a palavra chave `and`.

```
performance verse = [CC E{2} GG B{2} C5C5 GG C{4}] on violin;
performance chorus = [GAGA ABBA GEGE EBBE] on violin;

time chorusStart = start + 2*duration(verse);
time otherTimeChorusStarts = start + 3.14*duration(verse);

// tocar performances (chorus e tocada 2 vezes)
play verse;
at chorusStart and otherTimeChorusStarts play chorus;
```

2.2.4 Repetição

Há duas palavras chaves que permitem a repetição: o uso da palavra chave `times` permite repetir uma performance 0 ou mais vezes. `loop` permite repetir uma performance até ao fim da peça (`loop` = `play` ∞ `times`).

```
performance verse = [CC E{2} GG B{2} C5C5 GG C{4}] on violin;
performance chorus = [GAGA ABBA GEGE EBBE] on violin;
performance bass = [G|B|D{4} A|D|F#{4} G|B|D{4} A|D|F#{4}] on bass;

time chorusStart = start + 2*duration(verse);

// tocar performances
play verse 2 times on piano;
at chorusStart play chorus;

loop bass; // repeta ate ao fim da musica
```

2.3 Uso de arrays (TODO)

Palavras chaves all e sequentially

```
// perform
play sequentially all melody_lines // , back-to-back, consecutively
repeat_times times on piano;


(AFTER _ (ALWAYS?) | AT _ (AND _)* )? PLAY (SEQ? ALL)? _ (INT TIMES
    )? ON INST;
```

2.4 Modulações (TODO)

2.4.1 de Tom

Permitir slurring (mudança dinâmica) ou só pitch change (mudança "estática")? (Depende da biblioteca usada no backend)

2.4.2 de Tempo

Acelerar ou desacelarar o BPM da muúsica, temporariamente ou nao

3 Interação com o exterior

3.1 Estruturas de dados auxiliares

3.1.1 Strings

Strings são sequências de caracteres, números e símbolos delimitadas por aspas (`"`). Dentro duma string, aspas podem ser escapadas através de `\"`.

Não existe um tipo de dados String explícito, sendo este usado apenas como paraâmetro opcional para funções de I/O.

3.2 getInt(string?)

`getInt()` permite obter um inteiro através do *Standard In*.

Opcionalmente, pode ser passada uma String, que será impressa antes de aguardar a resposta do utilizador (uma String de *prompt*).

3.3 getSequence(string?)

À semelhança de `getInt()`, `getSequence()` permite obter uma sequência através do *Standard In*.

Opcionalmente, pode ser passada uma String, que será impressa antes de aguardar a resposta do utilizador (uma String de *prompt*).

4 Controlo de fluxo (TODO)

4.1 Instruções condicionais

4.1.1 if

```
1 if :
```

4.1.2 with ... choose

```
1 with getInt() choose:
2     case 1:
3         play seq on violin;
4     case 2:
5         play seq on guitar;
6     default:
7         play seq on piano;
```

4.2 Intruções de repetição

loops - Faz sentido ter loops? Talvez forEach de instrumentos?

Ter arrays? De sequencias, instrumentos(?), times e performances?

5 Configurações (TODO)

5.1 BPM (Beats Per Minute)

5.2 Especificação de (novos) instrumentos

(Depende da biblioteca externa...)

6 Exemplos (TODO)

6.1 Parabéns

```
// Happy Birthday
// override BPM setting
// BPM = 160;

// have time signature?

// get user input
int repeat_times = getInt("Number of repetitions: ");

// define melody sequences
sequence[] melody_lines = [
    [D{1.5} D{0.5}   E  D G F#{2}],
    [D{1.5} D{0.5}   E  D A G{2}],
    [D{1.5} D{0.5}   D5 B G F# E],
    [C5{1.5} C5{0.5} B  G A G{3}]];

// perform
play sequentially all melody_lines repeat_times times on piano;
```

6.2 Looping machine like thingy?

```
// tipo, add looping track, add looping track, etc.
```

6.3 Mais exemplos