Especificação da Linguagem

Grupo 12, LFA

Junho 2018

Conteúdo

1 Estruturas de dados (e 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. Mais que um `#` ou `b` podem ser aplicados a uma nota (por exemplo, `C##`).

Pode, depois do tom, aparecer um número, entre -2 e 8. Este especifica a oitava a usar, sendo -2 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 são também suportados:

```
int octave = 4;
```

1.5 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 entanto, nem todos suportam o mesmo registo (por exemplo, um piano suporta uma gama maior de notas que um violino). Se a um instrumento é dada uma sequência que este não suporta, as notas não suportadas são 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, num ficheiro externo de tipo auxiliar, estando detalhada na secção 5. No ficheiro de tipo principal, não é possível definir ou redefinir novos instrumentos, sendo apenas possível usá-los como constantes pré-definidas.

1.6 Arrays

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

Um array pode ser definido de duas formas distintas:

```
// criar um array com 4 intrumentos - e necessario usar a palavra
    chave instrument para indicar que e um array de instrumentos
instrument[] band = [piano, guitar, bass, drums];

// outra forma de definir o mesmo array
instrument[] band = piano and guitar and bass and drums;

// criar um array com 3 performances
performance[] p = [
    [D{1.5} D{0.5}   E  D G F#{2}] on piano,
    [D{1.5} D{0.5}   E  D A G{2}] on bass,
    [D{1.5} D{0.5}   D5 B G F# E] on guitar];
```

Pode usar-se um array para criar outro. A palavra chave `and` anexa ao fim do array já existente o elemento ou elementos dados. A palavra chave `except` retira o elemento ou elementos dados do array, se lá estiverem presentes.

```
instrument[] band = [piano, guitar, bass, drums];

// definir arrrays a partir de outros arrays
instrument[] new_band = band and violin;
instrument[] left_over_band = band except bass;
```

Para se aceder a um instrumento, a notação de parêntises retos, começando a contar no 0, é usada. A notação de intervalo, com dois pontos (`:`) é também suportada.

```
performance[] p = [
    [D{1.5} D{0.5}   E  D G F#{2}] on piano,
    [D{1.5} D{0.5}   E  D A G{2}] on bass,
    [D{1.5} D{0.5}   D5 B G F# E] on guitar];

// aceder ao primeiro elemento
performance intro = p[0];

// exemplo da notacao de intervalo
performance[] q = p[0:1] and [C5{1.5} C5{0.5} B G A G{3}] on violin
    and p[2:];
```

2 Geração de aúdio

2.1 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:

2.1.1 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;
```

2.1.2 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
```

2.1.3 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;
```

2.1.4 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; // repete ate ao fim da musica
```

2.2 Uso de arrays

Um array pode ser reproduzido de forma simultânea (forma utilizada por omissão) ou sequencialmente (através do uso da palavra chave `sequentially`).

```
// exemplo com sequencias
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}]];

play melody_lines /*sequentially*/ on piano; // descomentar
    sequentially para tocar melody_lines de forma sequencial
```

```
// exemplo com instrumentos
instrument[] band = [piano, guitar, bass, drums];

play melody_lines[0] /*sequentially*/ on band;
```

```
// exemplo com performances
performance[] perfors = [
    [D{1.5} D{0.5}   E  D G F#{2}] on piano,
    [D{1.5} D{0.5}   E  D A G{2}] on bass,
    [D{1.5} D{0.5}   D5 B G F# E] on guitar];

play perfors /*sequentially*/;
```

No entanto, não é possível reproduzir um array de sequências num array de instrumentos. Isto é, por exemplo, não é possível fazer o seguinte:

```
// definir array de sequencias
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}]];

// definir array de instrumentos
intrument[] band = [piano, guitar, bass, drums];

play melody_lines/*sequentially*/ on band; // ilegal!!
```

Para se obter este tipo de resultados, têm que ser utilizadas instruções de repetição (ver secção 4).

2.3 Modulações

Podem obter-se versões modificadas de sequências ou performances através das funções de modulação. Estas funções devolvem uma nova sequência ou performance, alterada em algum aspeto (tom ou tempo) em relação a uma dada sequência ou performance original, respetivamente.

2.3.1 de Tom

Pode mudar-se o tom de uma dada sequência ou performance de duas formas diferentes: `changeTone()` e `changeOctave()`. `changeTone()` aceita dois parâmetros - a sequência ou performance a alterar, e quantos meios-tons[4] deve subir ou descer; `changeOctave()` aceita, de forma semelhante, dois parâmetros - a sequência ou performance a alterar, e quantas oitavas deve subir ou descer.

```
// sequencia original
sequence s = [D{1.5} D{0.5} E D G F#{2}];

sequence s1 = changeOctave(s, -3); // diminuir a oitava por 3
// equivalente a dizer:
//    sequence s1 =  [D1{1.5} D1{0.5} E1 D1 G1 F#1{2}];

// mudar oitava da sequencia
sequence s5 = changeOctave(s, 1); // aumentar a oitava por 1
// equivalente a dizer:
//    sequence s5 =  [D5{1.5} D5{0.5} E5 D5 G5 F#5{2}];
// ou a dizer:
//    sequence s5 =  changeTone(s, 12);
```

```
// performance original
performance p = [D{1.5} D{0.5} E D G F#{2}] on bass;

play changeTone(p, 5);
// equivalente a:
//    play  [F#{1.5} F#{0.5} G# F# B A#{2}] on bass;
```

2.3.2 de Tempo

Pode mudar-se o tempo (ou seja, a velocidade) de um dada sequência ou performance através da função `changeTempo()`. Esta função aceita dois parâmetros - a sequência ou performance a alterar, e um número maior que 0 que representa a relação entre o novo tempo e o tempo original. A nova velocidade é dada por `fator * tempo original`.[5]

```
// performance original
performance p = [D{1.5} D{0.5} E D G F#{2}] on bass;

play changeTempo(p, 5); // toca a sequencia 5x mais rapido (cada
    nota dura 1/5 do seu tempo original)
```

[4] Doze meios-tons constituem uma oitava.

[5] Assim, um número maior que 1 acelera a velocidade, e um número menor que 1 reduz a velocidade.

```
// equivalente a:
//     play  [D{.3} D{0.1} E{.2} D{.2} G{.2} F#{.4}] on bass;
```

```
// performance original
performance p = [D{1.5} D{0.5} E D G F#{2}] on bass;

play changeTempo(changeTone(p, 5), 5); // toca a sequencia 5x mais
    rapido, com todas as notas 5 meios-tons mais agudas
// equivalente a:
//     play  [F#{.3} F#{0.1} G#{.2} F#{.2} B{.2} A#{.4}] on bass;
```

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 parâmetro 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

4.1 Instruções condicionais

4.1.1 if

As palavras chave `if`, `else if` e `else` permitem testar condições. Os operadores suportados numa condição são os de igualdade (`==`), desigualdade (`!=`), menor (`<`), maior (`>`), menor ou igual (`<=`), e maior ou igual (`>=`).

```
sequence s = getSequence("Enter a sequence: ");

if duration(s) > 5:
    play s on piano;
```

```
else if duration(s) > 2:
    play s on cello;
else:
    play s on guitar;
```

4.1.2 with ... choose

As palavras chave `with`, `choose`, `case` e `default` permitem testar a igualdade entre uma dada variável e um conjunto de valores.

```
with getInt("Pick a number: ") choose:
    case 1:
        play seq on violin;
    case 2:
        play seq on guitar;
    default:
        play seq on piano;
```

4.2 Intruções de repetição

4.2.1 for

As palavras chave `for` e `in` permitem definir instruções de repetição, ou seja, permitem que um dado código seja executado múltiplas vezes, iterando sobre todos os elementos de um dado array.

```
// instrumentos
instrument[] band = [piano, guitar, bass, drums];
for instrument inst in band:
    play [ABCDCBA] on inst;

// sequencias
sequence[] sequences = [
    [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}]];

for sequence seq in sequences:
    play seq on piano;

// performances
performance[] perfors = [
    [D{1.5} D{0.5}   E  D G F#{2}] on piano,
    [D{1.5} D{0.5}   E  D A G{2}] on bass,
    [D{1.5} D{0.5}   D5 B G F# E] on guitar];

time t = start;
for performance perfor in perfors:
    at t play perfor;
    t = t + duration(perfor);

// marcos de tempo
start_times = [start, 1, 3, 7];
for time t in start_times:
```

```
    at t play [C1 E1 G1 E1] on piano;
```

Caso se pretenda iterar sobre algum código um dado número de vezes, sem haver correspondência direta entre esse número e o conteúdo de um dado array, pode usar-se inteiros num dado intervalo, usando a função `range(a,b)`, que devolve um array de inteiros de tipo `[a, a+1, ..., b-2, b-1]`:

```
instrument[] band = [piano, guitar, bass, drums];
sequence[] sequences = [
    [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}]];

time t = start;
for int i in range(0:4):
    at t play sequences[i] on band[i];
    t = t + duration(sequences[i]);
```

5 Configurações (ficheiro auxiliar)

Um ficheiro do tipo principal (ficheiros com extensão `.principal`) suporta um todas as operações descritas até este ponto. Várias configurações podem ser feitas no ficheiro auxiliar (ficheiros com extensão `.auxiliar`). Nesta secção, vamos abordar as diferentes configurações que podem ser definidas através do ficheiro auxiliar.

5.1 BPM (Beats Per Minute)

`BPM` é uma palavra reservada[6] usada para configurar o *tempo* da música. A configuração deve ser feita no ficheiro de configuração, mas é possível reescrevê-la no ficheiro principal (o que define a música a gerar).

```
BPM = 160;
```

5.2 Especificação de (novos) instrumentos

O formato *midi* suporta 128 instrumentos diferentes[7].

É possível criar novos instrumentos associando a uma palavra um número, através da palavra chave `instrument`. O nome dado ao instrumento deixa de poder ser usado como nome de variável.

```
// definir dois novos intrumentos
instrument strings = 49;
instrument synth1 = 81;
```

Os instrumentos já definidos estão associados aos seguintes códigos: 1(`piano`), 25(`guitar`), 41(`violin`), 43(`cello`), 44(`bass`), e 119(`drums`).

[6] BPM não pode ser usado como nome de uma variável.

[7] Para obter mais informação sobre os diferentes instrumentos disponíveis, ver `http://www.ccarh.org/courses/253/handout/gminstruments/`.

6 Exemplos (TODO)

6.1 Parabéns

parabens.auxiliar

```
// set default BPM setting
BPM = 160;
```

parabens.principal

```
// 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 melody_lines sequentially repeat_times times on piano;
```

6.2 Looping machine like thingy?

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

6.3 Mais exemplos