LFSAB1403 : DJ'Oz

Edward Nicol (27101300)
Virgile Goyens(83391300)

4 décembre 2014

1 Structure du programme

Le programme est divisé en deux grandes fonctions : la fonction `fun {Interprete Partition}` et la fonction `fun {Mix Interprete Music}`.

Traitons dans un premier temps la fonction `fun {Interprete Partition}`. Cette fonction prend une partition comme argument et renvoie une voix, c'est à dire une liste d'échantillons. Cette fonction possède trois fonction locales : `fun{ToNote Note}`, `fun{CountNotes Partition Acc}`, `fun{GetEchantillon Note Facteur Transposer}` et fun `{SuperInterprete Partition Bourdon Facteur Transposer}`.

Les 3 premières sont des fonctions utilisées dans le programme. `{ToNote}` permet d'uniformiser le format des notes. `{CountNotes}` permet de compter le nombre de notes d'une partition (utile dans le cas d'une transformation du type `duree()`). Finalement, `{GetEchantillon}` renvoie un échantillon d'une note quelconque, en prenant en compte plusieurs paramètres.

La fonction `{SuperInterprete}` quant à elle représente le corps de la fonction. Grâce à ses paramètres supplémentaires, elle permet de tenir compte des modifications à apporter lors de la création d'un échantillon (du point de vue de la durée et/ou de la hauteur de la note. Elle utilise le principe du pattern matching pour interpréter la liste partition. Premièrement, un appel à la fonction `{Flatten}` est effectué pour ne plus avoir de liste imbriquées. Ensuite, pour chaque élément de la partition que la fonction rencontrera, elle vérifiera s'il s'agit d'une modification : le cas échéant, elle effectuera une modification dans les paramètres de la fonction `{SuperInterprete}`. Sinon, la fonction considérera que l'élément de partition est une note : elle fera donc appel à la fonction `GetEchantillon` et un appel récursif sera fait sur la queue de la liste. Le cas ou la partition ne comporte qu'un élément est aussi pris en compte.

2 Constructions non-délaratives

Notre code est entièrement construit à partir de constructions déclaratives. L'utilisation de cellules nous a paru superflue et celle des objets, inutile. En effet, nous avons uniquement procédé avec des fonctions récurrentes. Le seul avantage qu'aurait pu avoir l'utilisation de cellules aurait été un code plus succinct et des fonctions moins complexe.

3 Complexité des fonctions