II3516 : REPL

Marc Rozanc

2 février 2014

J'ai choisi de continuer l'interpréteur commencé en cours pour ce projet. Les ajouts se composent de la gestion des expressions booléennes, de la strucure `if`, de l'opérateur ternaire, de la boucle `while` et des fonctions.

1 Gestion des booléens

La gestion des booléens a été introduite par l'ajout d'un nouveau type d'expression, avec un type booléen représenté par les symboles `true` et `false`. De ce fait, les booléens ne sont pas compatibles pour des opérations avec les nombres.

Cette gestion séparée des booléens et des nombres m'a amené à définir un troisième type d'expression pour les valeurs dont il n'est pas possible de connaître le type par la syntaxe utilisée, par exemple lorsqu'on invoque une variable, son nom ne laissant pas transparaître le type de son contenu. Ce type a été nommé dans le code `dynamicExpression`.

2 La structure `if`

La structure `if` nécessite des parenthèses autour de sa condition ainsi que des accolades autour des blocs d'instructions qu'elle englobe afin d'éviter toute ambiguïté.

Afin de gérer le traitement de plusieurs instructions à la suite, il est nécessaire d'exécuter une instruction par ligne.

Exemple de code avec `if` :

```
if (a >= 3) {
    b = 6
} else {
    c = a
}
```

3 L'opérateur ternaire

L'opérateur ternaire ressemble beaucoup au `if` si ce n'est qu'il a forcément une branche `else` et qu'il renvoie une valeur.

4 Fonctions

Le corps des fonctions n'est pas évalué à la déclaration d'une fonction car je n'avais pas les moyens d'éviter la transformation *via* instaparse de l'AST pour le corps de la fonction.

Pour être utilisée, une fonction doit être nommée, donc stockée dans une variable :

```
f = ([n] n * n)
(f 4) // 16
```

Le corps de la fonction n'est évalué qu'à l'appel de la fonction et on utilise notre fonction `interpret` afin d'évaluer le contenu du corps. L'environnement est à chaque fois remplacé par un environnement vide peuplé avec les paramètres passés à la fonction. Les fonctions, au même titre que les variables, ne sont pas globales et donc non disponibles dans le scope de plus haut niveau.

5 La boucle `while`

La boucle `while` partage son environnement avec le scope de niveau supérieur et fonctionne sur le même principe que la fonction. La condition et les instructions restant identiques, l'AST de la condition et celui du corps sont conservés pour la boucle.