Compilation - N3, S5
TP 7 : Génération de code MVaP (3)

1. On enrichit le langage de Rationnel avec de nouvelles instructions et MVaP reste le langage cible.
2. Votre ami pour la documentation de ANTLR :
   `https://github.com/antlr/antlr4/blob/master/doc/index.md`
3. Pour la liste des instructions de la machine MVaP et leur description, voir le fichier MVaP_cheat_sheet.

Question 7.1 On va enrichir le langage de Rationnel en ajoutant les instructions suivantes
— Utilisation de variables globales : déclaration, affectation et utilisation dans des expressions.
— Blocs.
— Structures conditionnelles.
— Boucles.

La structure d'un programme dans le langage de Rationnel est maintenant : une suite de déclarations de variables suivie d'une suite d'instructions.

Une instruction dans le langage de Rationnel est maintenant soit :
— l'affichage d'une expression (voir fiche de TP 5),
— l'affectation d'une expression à une variable,
— une structure conditionnelle,
— une boucle.

Ce qui suit est la description de la syntaxe et de la sémantique de chacune des nouvelles instructions.

Variables. On a trois types de variables : `booleen`, `entier` et `rationnel`. La syntaxe pour la déclaration d'une variable est la suivante : `type var` où `var` est une liste d'identifiants séparés par une virgule `,`.

L'affectation d'une expression à une variable : `idVar = expr` où `expr` doit être du même type que `idVar`. La sémantique associée est le stockage en mémoire, à l'addresse de la variable `idVar`, la valeur de l'expression `expr`. On autorisera les déclarations-affectations de la forme `type idVar=expr`.

Une variable de type `T` peut être utilisée dans toute expression de type `T`. L'utilisation d'une variable `x` dans une expression consiste à récupérer la valeur stockée à l'addresse de la variable `x` et à l'utiliser dans l'expression.

Blocs. Un bloc est une suite d'instructions délimitée par les caractères `{` et `}`.

Structure conditionnelle. On accepte les instructions conditionnelles, possiblement imbriquées, de la forme suivante : `exprC? A : S` où `exprC` est une expression booléenne et la sémantique est celle classique : Si `exprC` est vraie, alors on n'exécute que `A`, sinon on n'exécute que `S`. Le `: S` est facultatif. De plus, l'ensemble des instructions `A` (ou `S`) peut être soit une seule instruction, soit un bloc d'instructions. Par exemple le code source suivant devrait être accepté :

```
true? x=35;
false? x=y; : z=x;
x==y? { x=y; z=x*y;}
10==13? {x=y; z=x*y} : z=x;
10==13? {x=y; z=x*y} : {z=x;x=t;}
```

Boucles. On accepte deux types de structures itératives :

— Les instructions itératives, possiblement imbriquées de la forme,
`Pour index=debut .. fin Faire instr`
où `index` est un identifiant d'une variable globale déjà déclarée, `debut` et `fin` sont des expressions de type entier, et `instr` est soit une instruction, soit un bloc d'instructions. Le sens d'une telle instruction est la répétition `fin-debut` fois de `instr`. La valeur de `index` à la fin de la boucle est égale à `fin`.

— Les instructions itératives, possiblement imbriquées de la forme suivante :
`repeter instr jusque exprC`
où `instr` est soit une seule instruction soit un bloc d'instructions et `exprC` est une expression booléenne. Le sens d'une telle instruction itérative est l'exécution de `instr` au moins une fois et tant que la condition `exprC` n'est pas satisfaite.