Compilation - N3, S5
TP 4 : Variable et Contrôle sémantique

1. L'objectif de ce TP c'est l'utilisation de variables et le contrôle sémantique. On va enrichir la grammaire `Expr_Calculette` du TP 3.
2. Votre ami pour la documentation de Antlr :
   `https://github.com/antlr/antlr4/blob/master/doc/index.md`
3. Se référer à la fiche de TP 1 pour l'installation et l'utilisation de Antlr
4. On va continuer à enrichir la grammaire `Expr_Calculette` de la fiche de TP 3.

Question 4.1 Une composante utile dans les langages de programmation est l'utilisation de variables. Les variables permettent entre autres de stocker des expressions que l'on voudra ré-utiliser.

1. Modifiez la grammaire `Expr_Calculette` pour ajouter une instruction permettant de déclarer une variable. On aura deux types de variable : le type `entier` pour les variables de type entier et le type `booleen` pour les variables de type booléen. La déclaration d'une variable se fera avec l'instruction :

   ```
   type idVariable
   ```

   Il faudra aussi permettre la déclaration de plusieurs variables du même type avec l'instruction

   ```
   type idVariable1, idVariable2, ..., idVariablen
   ```

2. Modifiez la grammaire `Expr_Calculette` pour ajouter l'instruction d'affectation d'une expression à une variable. L'affectation se fera avec l'instruction :

   ```
   idVariable = expr
   ```

   L'interprétation d'une instruction d'affectation comme `x=expr` consiste à affecter la valeur de `expr` à la variable nommée `x`.
3. Modifiez la grammaire `Expr_Calculette` pour que l'on puisse utiliser des variables dans les expressions arithmétiques, booléennes et dans les instructions d'affichage. L'interprétation d'une variable dans une expression arithmétique ou booléenne ou dans une instruction d'affichage est la valeur de la variable.
4. Modifiez votre compilateur pour faire les contrôles suivants dans les variables :
   — Toute variable doit être déclarée.
   — Toute variable doit avoir une valeur lorsqu'elle est utilisée dans une expression.
   — Une variable de type booléen ne peut être utilisée que dans des expressions booléennes. De même une variable de type entier ne peut être utilisée que dans des expressions arithmétiques.
5. Modifier la grammaire pour ajouter des instructions d'incrémentation et de décrémentation de variables entières. Si `x` est une variable, alors
   — `++x` (resp. `--x`) dans une expression signifie l'utilisation de `x` et ensuite on incrémente (resp. décrémente) `x`,

— `x++` (resp. `x--`) dans une expression signifie l'incrémentation (resp. la décrémentation) de `x` et ensuite son utilisation dans l'expression.

Question 4.2 Une autre composante utile dans les langages de programmation est une instruction de lecture.

Modifier la grammaire `Expr_Calculette` pour que l'on puisse utiliser l'instruction `lire` dans des expressions booléennes et arithmétiques. Lorsqu'elle est utilisée dans une expression arithmétique, il faudra lire un entier, et lorsqu'elle est utilisée dans une expression booléenne, il faudra lire un booléen.

Question 4.3 Pour pouvoir avoir des instructions de contrôle et des fonctions, les langages de programmation doivent disposer de la notion de bloc. Un bloc est un ensemble d'instructions et tout objet d'un bloc n'est visible qu'à l'intérieur du bloc.

Modifier votre grammaire pour ajouter les instructions de bloc. Tout bloc sera délimité par une accolade ouvrante `{` et une accolade fermante `}`.

Voici quelques rappels sur les variables locales :
— Une variable `x` déclarée dans un bloc n'est pas visible à l'extérieur du bloc, mais est visible dans les sous-blocs du bloc de déclaration de `x`.
— Une variable modifiée dans un sous-bloc retrouve sa valeur initiale à la sortie du sous-bloc. La valeur initiale d'une variable dans un sous-bloc c'est sa valeur avant entrée dans le sous-bloc.
— Une variable locale `x` d'un bloc peut avoir le même nom qu'une variable globale, et dans ce cas la variable locale cache la variable globale.
— Des variables locales de blocs différents peuvent avoir le même nom et donc peuvent être de types différents.

Voici un exemple, qui devrait afficher dans l'ordre `'false', '2', 'false', '5', '7', '2' et '3'`,

```
int x, y, z;
x=3; y=2; z=7;
{ bool x; x=false; Afficher(x); Afficher(y);
    { int y; y=5; Afficher(x); Afficher(y); Afficher(z); };
    Afficher(y); };
Afficher(x);
```