****

**LOG3210 – Éléments de langages et compilateurs**

**Hiver 2021**

**Examen Intra**

**Groupe 01**

**1947497 – Yuhan Li**

**Soumis à : Madame Guerrier**

**15 mars 2021**

**Grammaire**

D’abord, pour l’importation j’ai créé la fonction ImportStmt(). J’ai appelé cette fonction une seule fois, soit au début de la fonction Program, sachant qu’on effectue les importations seulement une fois au début du code. ImportStmt() comprend deux jetons (début et fin d’importation) ainsi qu’une fonction Identifier() qui est une des fonctions de base du langage utilisée pour les expressions quelconques commençant par une lettre (ex. : noms de fonction, de variable, etc).

Outre Identifier(), ma grammaire comporte 4 autres expressions qui sont à la base de la syntaxe. Ces fonctions sont dédiées aux String, Bool, Real et Int. Chacune de ces fonctions possèdent un jeton que je sauvegarde au niveau du nœud. J’ai regroupé le tout dans une fonction nommé PrimitiveExpr() pour simplifier l’appel à ces valeurs de base. Sachant qu’on peut assigner des valeurs directement par l’appel de fonction, j’ai également ajouté CallFunction() dans les options de PrimitiveExpr().

CallFunction() est aussi appelé dans Stmt() puisqu’un appel de fonction peut être une ligne de code à lui seul. En effet, j’ai assumé que chaque fonction appelée dans Stmt() était des lignes ou des blocs d’expressions qui ont une exécution à part entière. Donc, en plus de CallFunction(), j’ai inclus le cœur, la déclaration de variable et de fonction, ainsi que les If/While.

Pour la déclaration, j’ai divisé ça en deux fonctions (*DeclareStmt* et *AssignStmt*), simplement pour avoir une meilleure visualisation de ce qui se produit. Pour DeclareStmt(), j’ai ajouté un jeton (“compi”) pour l’annonce d’une déclaration. J’ai également ajouté d’autres terminaux au niveau du jeton TYPE afin de représenter chaque type nécessaire à EstheRust. Pour AssignStmt(), j’ai implémenté Expr() qui est appelée à la droite du terminal “= ”. Cette fonction représente des valeurs concrètes qui peuvent provenir d’une opération logique ou arithmétique. C’est pourquoi, Expr() est implémentée de manière récursive afin d’offrir une possibilité d’accès à chaque type d’opération et aux valeurs de base de PrimitiveExpr().

Opérations : récursivité suit l’ordre d’importance inspiré du tp1 et tp2. 4 fonctions. comp, add, una et paren. C’est dans paren que ca reach les valeurs de base (int, real, string, etc).

Déclaration de fonction. J’ai implémenté deux fonctions pour, encore une fois, mieux visualiser les évènements.

If et While

Enfin, pour l’opérateur *Cœur*, ma syntaxe consiste uniquement d’une fonction comprenant le jeton <HEART> que j’ai ajouté au lexique.

**Visiteur**