Mini rapport explicatif

Grammaire traitée

```
Prog ::= '[' Cmds ']'

Cmds ::= Stat
       | Dec ';' Cmds
       | Stat ';' Cmds

Dec ::= 'VAR' ident Type
      | 'CONST' ident Type Expr
      | 'PROC' ident Prog
      |  'PROC' ident '[' Args ']' Prog

Stat ::= 'SET' ident Expr
       | 'IF' Expr Prog Prog
       | 'WHILE' Expr Prog
       | 'ECHO' Expr
       | 'CALL' ident
       | 'CALL' ident Exprs

Expr ::= 'true'
       | 'false'
       | num
       | ident
       | '(' 'not' Expr ')'
       | '(' 'and' Expr Expr ')'
       | '(' 'or' Expr Expr ')'
       | '(' 'eq' Expr Expr ')'
       | '(' 'lt' Expr Expr ')'
       | '(' 'add' Expr Expr ')'
       | '(' 'sub' Expr Expr ')'
       | '(' 'mul' Expr Expr ')'
       | '(' 'div' Expr Expr ')'

Exprs ::= Expr
        | Expr Exprs

Args ::= Arg
       | Arg ';' Args

Arg ::= ident ':' Type

Type ::= 'void'
       | 'bool'
       | 'int'
```

Analyseur

L'analyseur est composé des fichiers suivants :

- « aps.l » décrivant l'analyseur lexical.
- « aps.y » décrivant l'analyseur syntaxique.
- « nœud.c » et « nœud.h » décrivant les différents types de nœuds formant l'AST engendré par l'analyseur syntaxique. La méthode « toStringNoeud » permettant d'obtenir le terme prolog correspondant à l'AST se trouve dans ces fichiers.
- « liste.c » et « liste.h » décrivant une liste correspondant au contenu de chaque nœud de l'AST. Une méthode « toStringListe » est également présente.

Typeur et Evaluateur

Le fichier « typeur.pl » est très largement commenté. Cependant quelques points nécessitent tout de même quelques explications.

Il a été choisi d'utiliser deux environnement, un pour les informations de typage des variables et un deuxième pour les constantes. Par soucis pratique les informations de typage des procédures sont également stockées dans l'environnement des constantes, celles-ci ne disposant également pas de la possibilité d'être redéfinies ou réaffectées.

Il a également été choisi de ne pas permettre la déclaration d'une variable/constante du même nom qu'une variable/constante déjà existante. Les procédures ayant accès aux variables/constantes présentent dans l'environnement lors de leur déclaration, il est également impossible de déclarer à l'intérieur de celles-ci une variable/constante ou un paramètre de même nom qu'une variable déjà existante.

Le typeur on interdit l'affectation d'une constante, cette vérification n'est donc plus utile lors de l'évaluation. Les variables et les constantes sont donc stockées uniformément sans distinction lors de l'évaluation.

De ce dernier fait il a été décidé de ne pas utiliser une mémoire en plus d'un environnement lors de l'évaluation. Cela engendre néanmoins un changement de comportement de l'évaluateur lors de la création de la fermeture d'une procédure. En effet si l'environnement est capturé à cet instant, alors les valeurs des variables le sont également contrairement au cas où l'on dispose d'un système d'adresse/mémoire. Pour remédier à cela on décide de ne pas capturer l'environnement au moment de la création d'une fermeture. Cette dernière modification entraîne le passage d'une portée statique à une portée dynamique à première vue. Cependant, lors du typage il a été vérifié que chaque identifiant utilisé correspondait bien à une variable ou une constante déjà déclarée. Cette vérification nous assure donc la propriété de portée statique.

En ce qui concerne les procédures, comme décrit dans le cours, le passage de paramètre se fait par copie de valeur et tout les paramètres sont considérés comme des constantes.

Remarque :

Seul l'analyseur et le typeur sont capable de prendre en compte les procédures. L'évaluateur ne peut prendre en entrer que des programmes formé par la grammaire APS0.