Master 1 – Compilation
Écriture d'une grammaire simple avec PLY

Dans ce TP, nous allons écrire une grammaire pour un langage simple que nous appellerons « mon langage » et dont l'extension sera mon langage. L'extension pour les programmes « mon langage » sera « .ml »

Contexte

« Mon Langage » a une structure très simple découpée en deux parties. Une première partie appelée DECLARATIONS permet de déclarer des variables entières et réelles et de leur affecter éventuellement des valeurs.

La deuxième partie commence par le mot-clef CODE et permet de réaliser des opérations simples comme lire des entiers au clavier, calculer des expressions arithmétiques et écrire des résultats à l'écran.

Vous avez ci-dessous un exemple de programme ml :

```
// La double barre introduit les commentaires
// pas d'autres formes de commentaires

// Une partie déclarative qui commence par DECLARATIONS
// et une partie code qui commence par CODE

DECLARATIONS

// deux types de variables double ou int
// pour les double une valeur numérique est
//indispensable avant et après le "."
// les chaînes sont entre  " et "

double  d ;
int i  ;
int maValeur ;

int valeurNulle ;
int calcul1 ;
double calcul2;

CODE
```

```
read(i);
calcul1 =-(i+1);
calcul2 = d*5.26*3 - 2.0;
write(calcul1);
write(calcul2);
write("fin de mon programme") ;
```

Analyse lexicale

Comme vous pouvez le remarquer le programme est très simple. Il comporte une partie déclarative qui débute par le mot-clef DECLARATIONS puis la partie instruction proprement dite qui débute elle par le mot-clef CODE.

Il ne permet de déclarer que des entiers ou des double et chaque variable doit être déclaré par l'instruction **int identificateur** ; ou **double identificateur ;**

Nous partirons du principe que les identificateurs sont écrits en minuscules, qu'un identificateur commence systématiquement par une lettre et se poursuit par un mélange de lettres et de chiffres.

L'instruction read() servira à lire un élément, l'instruction write permet d'afficher un identificateur.

Les autres éléments terminaux de notre langage sont outre les nombres entiers, les parenthèses ouvrantes et fermantes, et les quatre opérateurs classiques (+,*,/,-).

Travail à réaliser

Écrire l'analyse lexicale complète de ce langage en utilisant PLY.

Par exemple, le programme proposé devrait vous sortir l'analyse suivante :

```
1   COMMENT    // Une partie déclarative qui commence par DECLARATIONS

2   COMMENT    // et une partie code qui commence par CODE

4   MC     DECLARATIONS

6   COMMENT    // deux types de variables double ou int

7   COMMENT    // pour les double une valeur numérique est

8   COMMENT    //indispensable avant et après le "."
```

```
9   COMMENT     // les chaînes sont entre  " et "


11   MC     double
11   ID     d
11   PTVIRG     ;

12   MC     int
12   ID     i
12   PTVIRG     ;

13   MC     int
13   ID     maValeur
13   PTVIRG     ;


15   MC     int
15   ID     valeurNulle
15   PTVIRG     ;

16   MC     int
16   ID     calcul1
16   PTVIRG     ;

17   MC     double
17   ID     calcul2
17   PTVIRG     ;


19   MC     CODE


21   MC     read
21   PARG     (
```

```
21 ID i
21 PARD )
21 PTVIRG ;

22 ID calcul1
22 EGAL =
22 MOINS -
22 PARG (
22 ID i
22 PLUS
22 ENTIER 1
22 PARD )
22 PTVIRG ;

23 ID calcul2
23 EGAL =
23 ID d
23 MUL *
23 REEL 5.26
23 MUL *
23 ENTIER 3
23 MOINS -
23 REEL 2.0
23 PTVIRG ;

24 MC write
24 PARG (
24 ID calcul1
24 PARD )
24 PTVIRG ;

25 MC write
25 PARG (
25 ID calcul2
25 PARD )
```

```
25    PTVIRG     ;

26    MC     write
26    PARG     (
26    CHAINE     "fin de mon programme"
26    PARD     )
26    PTVIRG     ;
```

Un fichier qui a une erreur ne doit pas provoquer une erreur à l'analyse lexicale.

Extension de la grammaire

Modifier la grammaire de manière à prendre en charge l'instruction if

```
if ( x > 0)
        z = -2 ;
else
        z = x+ 4 ;
```