Projet d'Analyse Syntaxique - Annee 2009-2010

Le pojet doit être réalisé en binome. Vous devrez remettre par mail à votre chargé de TD les fichiers suivant :

1. un fichier `projet<vosNoms>.tar.gz` contenant votre programme.
2. Une explication de votre travail, sous la forme d'un texte en format `.pdf`.
3. Des fichiers présentant des exemples de code C et la traduction obtenue avec votre analyseur.

Le projet consiste à écrire un analyseur syntaxique avec `yacc` qui analyse des blocs de code C et les traduise en python3. Les blocs (entre accolades) pourront contenir des déclaration et des affectations de variables.

Etape 1
Reprendre l'exercice 6.3 de la feuille 6. Ajouter une règle de grammaire permettant d'analyser un bloc d'instructions entre accolades. Le bloc se terminera par une instruction de la forme `return expression;` et le progamme devra calculer la valeur de cette dernière expression.

Etape 2
Utiliser une table de symboles qui contiendra pour chaque entrée dans la table une chaîne de caractères, un type et une valeur. Les valeurs seront initialisées à zéro. Les types seront tous $DOUBLE$ ou INT, $DOUBLE$ et INT étant des pseudo-constantes.

Etape 3 :
Le bloc devra maintenant commencer par des déclarations de variables. Les variables pourront être de deux types : *int* ou *double*. L'analyseur devra produire le code équivalent à ce bloc écrit en Python3 : les déclarations de variables seront supprimées, mais les variables et leur type déclaré seront stockés dans la table de symboles. Le point-virgule à la fin des instructions d'affectation sera supprimé et remplacé par un passage à ligne. Le symbole de division "/" sera remplacé par "//" dans le cas d'une division entre deux expressions à valeur entière. Par exemple :

```
{int x; int y;

x = 25;
y = x/2;
return y;}
```

sera traduit en :

```
x = 25
y = x//2
return y
```

Etape 4 :

Ajouter des règles de grammaire pour que l'analyseur traduise une boucle `for` simple en python. Exemple :

```
for( int i=1; i<10; i=i+2)
    x=i*x;
```

sera traduit en

```
for i in range(1,10,2):
    x=i*x
```

Etape 5 facultative :

Vous pourrez apporter des améliorations à votre programme. Par exemple :
- gérer des boucles imbriquées avec l'indentation appropriée
- analyser des appels à la fonction `printf`. Exemple :

```
printf("The answer is %d",2*x);
```

sera traduit en

```
print("The answer is ", 2*x)
```