Master Info - 2019-2020
MIF08 Compilation

Contrôle continu 2 (CC-TP) SUJET Exemple - Durée 20 min

Éléments de correction

1 Manipulations préliminaires

— Sauvegardez vos modifications faites dans le git étudiant, par exemple :

```
git commit -a -m "mes modifs"
```

— Récupérer le sujet de contrôle :

```
git pull
```

— Vous travaillez dans le répertoire `CC2/CC2-ex`. Un Makefile, un main et une grammaire commentée est fournie :

```
grammar AnB2n;

start : EOF ;

COMMENT : '//' ~[\r\n]* -> skip ; // for unit tests in comments
WS : [ \t\r\n]+ -> skip ; // skip spaces, tabs, newlines
```

— Ouvrir le Makefile et changer `JohnDoe` en votre prénom suivi de votre nom (sans accent, sans espace, sans caractère spécial, tirets autorisés).

— Vérifier que `make; make tests` fonctionnent. Les tests doivent retourner des erreurs, bien sûr.

2 Exercice - grammaire avec ANTLR

L'objet de cet exercice est d'écrire un analyseur qui reconnaît le langage $a^n b^{2n}$, $(n > 0)$. Les autres caractères alphabétiques (c,d, ... z, A, ... Z) ainsi que les blancs et tabulations seront ignorés, les autres caractères feront une erreur de syntaxe.

1. On vous fournit deux fichiers de tests `tests/ex0.txt` et . `tests/ex1.txt`
2. Éditer le `.g4` pour coder l'analyseur (lexical/syntaxique). Tester avec :

```
make
python3 sujetEx.py testsfiles/ex0.txt
```

On a écrit pour vous dans le main un affichage de "ok" (avec `EXITCODE 0`) si le fichier est accepté par la grammaire, par exemple ici. Dans le cas d'un fichier non accepté par le lexer, le programme affiche "syntax error" (avec `EXITCODE 2`), et dans le cas d'une erreur de lexicographie, "lexical error" (avec `EXITCODE 1`). En cas d'erreur, on peut obtenir le diagnostique d'ANTLR avec `python3 sujetEx.py --verbose`

3. Dans le répertoire `testfiles/` ajouter 5 tests pertinents pour cet analyseur (positifs, négatifs) et faire en sorte que cela fonctionne avec `make tests`.
4. Pour déposer :

   `make clean; make tar`

   vous fournit le tgz à déposer sur TOMUSS.

Solution:

```
grammar AnB2n;


//UNCOMMENT start: EOF;
// BEGIN CUT

start: akbk EOF;

akbk:
        A akbk B B
    |
;


A:'a' ;
B:'b' ;
CHARS: [c-z,A-Z] -> skip ; //skip chars


// END CUT



COMMENT
 : '//' ~[\r\n]* -> skip
 ;

WS : [ \t\r\n]+ -> skip ; // skip spaces, tabs, newlines
```