Master Info - 2024-2025
MIF08 Compilation

Contrôle continu 1, sujet EXEMPLE - Durée 24 min (tiers-temps 32 min)

Documents autorisés : le git étudiant de mif08 uniquement. Ce TP est à rendre sur TOMUSS.

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 `CC1/CC1-ex`. Une grammaire (`.g4`) à compléter est fournie, ainsi que le nécessaire pour exécuter et tester (code Python et Makefile).

— Ouvrir le Makefile et changer `JohnDoe` en votre prénom suivi de votre nom (sans accent, ni caractères spéciaux).

— Vérifier que `make; make test` fonctionnent. Les tests doivent retourner des erreurs pour l'instant, bien sûr.

— Ouvrir un navigateur avec un onglet TOMUSS pour le rendu.

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 commentaires correspondent à `//` suivis d'une suite quelconque de caractères, jusqu'à la fin de ligne incluse, et ils sont ignorés. Les autres caractères alphabétiques (c,d, ...z, A, ...Z) ainsi que les blancs et tabulations seront ignorés, les autres caractères feront une erreur lexicale (sauf à l'intérieur d'un commentaire).

1. On vous fournit des fichiers de tests `testfiles/simple-valid.txt`, `testfiles/syntax-error.txt` et `testfiles/lexical-error.txt`.
2. Éditer le `.g4` pour coder l'analyseur (lexical/syntaxique). Tester avec :

   ```
   make
   # Test à la main sur un fichier
   python3 main.py testfiles/simple-valid.txt
   # Test automatique sur tous les fichiers
   make test
   ```

   On a **écrit pour vous dans le main** un affichage de "ok" si le fichier est accepté par la grammaire, par exemple ici. Dans le cas d'une erreur de lexicographie, "lexical error", et dans le cas d'un fichier non accepté par le lexer, le programme affiche "syntax error". En cas d'erreur, on peut obtenir le diagnostic d'ANTLR avec `python3 main.py --verbose`
3. Comme en TP, chaque fichier de tests doit contenir l'entrée du programme, la sortie attendue en commentaire derrière l'annotation `// EXPECTED`. En plus, vous devez spécifier le status de retour en cas d'erreur via `// EXITCODE 1` pour les erreurs lexicales et `// EXITCODE 2` pour les erreurs de syntaxe.
4. Dans le répertoire `testfiles/` ajouter au moins 5 tests pertinents pour cet analyseur (positifs, négatifs) et faire en sorte que cela fonctionne avec `make test`.
5. Pour déposer :

   ```
   make clean; make tar
   ```

   vous fournit le tgz à déposer sur TOMUSS.