Contrôle de TP

Cadavre exquis

« jeu qui consiste à faire composer une phrase, ou un dessin, par plusieurs personnes sans qu'aucune d'elles puisse tenir compte de la collaboration ou des collaborations précédentes.» (Dictionnaire abrégé du surréalisme)

Ce TP consiste à écrire un programme qui va écrire des phrases en choisissant les mots au hasard à la manière d'un cadavre exquis. L'ordre syntaxique : `nom-adjectif-verbe-COD-adjectif` doit être respecté pour que la phrase soit grammaticalement correcte.

Ce TP se décompose en deux phases. La première consiste à la lecture de la grammaire définissant une phrase. La seconde est la génération de la phrase proprement dite.

Les fichiers à disposition

Pour faciliter le TP des fichiers vous sont fournis, il faudra les remplir au fur et à mesure du TP. Il vous est conseillé de lire l'ensemble de ces fichiers pour repérer où doivent s'insérer vos ajouts.

`main.cpp` : Ce fichier contient le programme principal qui lira une grammaire et générera une phrase

`test.cpp` : Ce fichier contiendra les tests de vos fonctions (pourra être appelé avec `make test`)

`split.h, split.cpp` : contient la fonction qui découpe une ligne en mot

`grammar.h` : contient la définition des types utilisés ainsi la fonction de la lecture de la grammaire

`grammar.cpp` : Implémentation de la lecture de la grammaire

`generation.h, generation.cpp` : Fonctions de générations d'une phrase

`grammaire.txt` : fichier texte contenant la description de la grammaire

`Makefile` : Makefile

Les fichiers se trouvent dans `/home/shared/ELI5_C++/TP_TEST/fournis/`

L'ensemble de ce TP doit être fait dans un répertoire `votre_login_test_c++/`.
Il devra être rendu en copiant ce répertoire dans :
`/home/shared/ELI5_C++/TP_TEST/soumission/`

La grammaire

La grammaire est défini dans le fichier `grammaire.txt` de la façon suivante :

<phrase>	<nom-compose> <verbe> <nom-compose>
<nom-compose>	<nom>
<nom-compose>	<nom> <adjectif>
<nom>	la colombe
<nom>	le ciel
<adjectif>	rouge
<verbe>	voit

Avec cet exemple on peut construire les phrases suivantes :
la colombe rouge voit le ciel
le ciel voit la colombe
...

Nous avons défini une phrase comme étant un nom composé suivit d'un verbe et d'un autre nom composé. Un nom composé peut être un nom tout court ou un nom et un adjectif. Enfin, nom, adjectif, verbe sont des mots qui vont bien.

Exercice 1 : Lecture de la grammaire

Le but est de lire un fichier de grammaire décrit plus haut et d'obtenir une structure de donnée qui représente cette grammaire. Le fichier contient plusieurs lignes qui commencent toujours (si elles ne sont pas vide) par un nom de règle (label) suivit de règles ou de mots. De plus, on peut avoir un même label associé à des règles différentes (cf. <nom-compose>).

On a donc :
- Une grammaire = (label, collection de règle)
- Une collection de règle = un ensemble de règle
- Une règle = un ensemble de mot

Question 1

Définissez les types grammaire, collection de règle et règle. Vérifier votre idée avec les types défini dans `grammaire.h`

Question 2

Définissez la fonction de lecture d'une grammaire (fonction `read_grammar` du fichier `grammar.cpp`). Vous pouvez utiliser la fonction `split` qui découpe une ligne en vecteur de mot.

Question 3

Ecrire le test de votre fonction (fonction `testGrammaire` du fichier `test.cpp`), puis tester votre fonction en faisant `make test`.

Exercice 2 : Génération de phrase

La génération d'une phrase se fait en suivant les règles défini par la grammaire.

1. On commence au premier label `<phrase>` qui décrit les règles de construction d'une phrase.
2. On choisit une des règles au hasard dans la collection de règles associées au label.
3. On regarde la règle choisie.
4. Si la règle contient un label alors faire l'étape 2. Sinon, insérer le mot dans la phrase

Une règle se distingue d'un mot par le fait qu'elle se trouve entre "bracket" (<label>).

Question 1

Ecrire la fonction de génération des phrases (fonction `gen_aux` du fichier `generation.cpp`). La fonction de test est déjà écrite, il s'agit de la fonction `main` du fichier `main.cpp`. Tester donc votre fonction par `make` suivi de `./main`.

Exercice 3 : Bonus

Ajouter le mode question/réponse. On voudrait maintenant pouvoir générer une phrase question suivit d'une phrase réponse.

Exemple : Que voit la colombe rouge ?
L'arbre change le ciel