Université Bordeaux 1
Licence, module Compilation, 2017/2018

TD2-Analyse lexicale

Exercice 2.1

Dans un langage programmation, on ne manipule pas des lettres mais des mots. Ces mots sont des éléments du langage, comme les mots clés `if`, `while` ou `return` en C, ou comme tous les noms de variable. Les mots sont séparées par des caractères de type espace, l'espace, le saut à la ligne et la tabulation. Réalisez un programme qui compte le nombre de mots dans son entrée. Améliorez ce programme pour qu'il affiche le nombre de mot par ligne en indiquant le numéro de la ligne.

Exercice 2.2

Pour gérer les variables, un compilateur doit mémoriser les noms de variable dans un programme. Réalisez un programme qui compte les occurrences de chaque mot de l'entrée. On pourra supposer qu'il existe au plus 100 noms de variable différents dans votre entrée.

Par exemple, sur l'entrée

```
voila un programme dans un langage inattendu.
voila un langage inattendu pour un programme.
```

la sortie du programme est

```
voila:2 un:4 programme:2 dans:1 langage:2 inattendu:2 pour:1
```

Exercice 2.3

Un mot bien parenthésé est un mot dont tous les préfixes ont plus de parenthèses ouvrantes que fermantes. Intuitivement, quand on lit un tel mot, on ne peut pas fermer une parenthèse s'il n'y a pas de parenthèse ouvrante correspondante avant. `(()())` est bien parenthésé mais `())(()` ne l'est pas. En ignorant les caractères qui ne sont pas des parenthèses, réalisez un programme qui dit si oui ou non, son entrée est bien parenthésée.

Exercice 2.4

Un code bien indenté, c'est un code plus facile à lire. Habituellement, on indente des blocs d'instructions, ici, on simplifie en supposant que l'entrée est bien parenthésée et peut comporter plusieurs types de parenthèses, autrement dit, des crochets `[]` et des accolades `{}`. Réalisez un programme qui indente à chaque saut de ligne l'entrée en fonction du nombre de parenthèses ouvrantes non fermés par une parenthèse fermante.

Par exemple, sur l'entrée

```
for (k=0; i < 10; k++) {
tab[k] = { {42,8,
 16,8},9,
 7 };
}
  return tab;
```

la sortie sera

```
for (k=0; i < 10; k++) {
tab[k] = { {42,8,
    16,8},9
  7 };
}
return tab;
```

Note : ce système n'est pas parfait, notamment, en C, on indente l'instruction qui suit un test `if`, ce programme ne le fera pas.

Note bis : le programme `clang-format` fourni avec le compilateur `clang` permet d'uniformiser l'indentation, et d'autres choix d'écriture. `clang-format -help` pour plus d'information.

Exercice 2.5

Les commentaires font partie du code. Cependant, ils doivent être ignoré par le compilateur. On considère les commentaires C du type `/*...*/`.

— Écrivez un expression régulière simple qui repère un commentaire C.

— Quand il y a plusieurs commentaires, imbriqués ou non, l'expression précédente peut échouer. Utilisez les *start-conditions* afin pour corriger la solution précédente

On pourra observer le résultat du programme soit en extrayant uniquement les commentaires, soit en affichant tout sauf les commentaires