TP 1 : Utilisation de `flex`

Exercice 1 - Extraction des chaînes de caractères

L'objectif de cet exercice est de produire un programme dont le comportement est similaire à celui de la commande `strings`. Le programme va lire en continu sur le flux standard d'entrée, et afficher sur le flux standard de sortie uniquement les mots (suites de lettres minuscules ou majuscules) rencontrés.

1.1 Écrivez un fichier `strings.l` contenant les filtres et actions nécessaires pour réaliser le traitement décrit ci-dessus.

1.2 Générez le fichier `.c` correspondant à `strings.l`, compilez le et testez votre programme.

1.3 Dessinez l'automate non déterministe utilisé par `flex`.

1.4 Déterminez les classes de caractères, puis dessinez l'automate déterministe utilisé par `flex`.

1.5 Modifiez votre fichier `strings.l` pour que le programme, lorsqu'il arrive à la fin du flux de lecture, affiche en plus le nombre total de mots rencontrés.

note : Dans `flex`, la fin du flux d'entrée est représentée par `<<EOF>>`.

1.6 Modifiez votre fichier `strings.l` pour que celui-ci lise ses données dans le fichier dont le chemin est donné en argument, si un tel argument existe.

Exercice 2 - Mots contenant exactement un b

On considère l'alphabet $A = \{a,\ b\}$ et l'ensemble L des mots de A^* qui contiennent exactement une fois la lettre b.

2.1 Donnez une expression régulière correspondant au langage L.

2.2 Dans un fichier `only_one_b.l`, utilisez cette expression régulière afin d'afficher un message lorsque la ligne saisie sur l'entrée standard est un mot de L.

note : Vous aurez en fait besoin de trois règles =
- une règle pour filtrer les mots de L et afficher un message,
- une règle pour ignorer a et b lorsque l'entrée est un mot de A^* qui n'est pas dans L,
- une règle pour afficher un message d'erreur et arrêter le programme lorsqu'on rencontre un caractère n'appartenant pas à l'alphabet A.

2.3 Générez le fichier `.c` correspondant à `only_one_b.l`, compilez le et testez votre programme.

2.4 L'ordre des règles dans `only_one_b.l` est-il important ?

2.5 Lorsqu'on saisit une ligne sur l'entrée standard, le dernier caractère de cette ligne est un saut de ligne (caractère `'\n'`).

Pourquoi n'a-t-on pas besoin d'une règle supplémentaire pour gérer ce caractère ?

2.6 Modifiez votre fichier `only_one_b.l` pour afficher uniquement le pourcentage de mots valides saisis sur l'entrée standard.

2.7 Examinez l'automate déterministe généré par `flex`. À quoi sert l'état 2 ?

Exercice 3 - Suppression des commentaires avec `flex`

L'objectif de cet exercice est de réaliser un programme `cleanup` dont le but est de supprimer les commentaires d'un code (`sh` ou `C`).

3.1 Écrivez un filtre (fichier `cleanup_sh_regex.l`) qui, sur la donnée d'un script shell lu depuis l'entrée standard, enlève les commentaires et affiche le résultat sur la sortie standard.

On utilisera pour ce faire une unique expression régulière permettant de détecter (et ainsi de supprimer) les commentaires.

3.2 Générez le fichier `.c` correspondant à `cleanup_sh_regex.l`, compilez-le et testez sérieusement votre programme.

note : Normalement, vous devriez identifier plusieurs problèmes avec cette approche.

3.3 Refaites les questions précédentes, mais dans un fichier `cleanup_c_regex.l` et en supprimant cette fois les commentaires `C` (entre `/*` et `*/` inclus).

3.4 À ce stade, l'approche à base d'expressions régulières a du montrer ses limites.

Une solution alternative est de lire les caractères de l'entrée un par un. On peut alors maintenir à jour plusieurs variables booléennes, dans le but de savoir à tout moment si (entre autres) :

— on est actuellement à l'intérieur d'un commentaire,
— le précédent caractère était un `\`.

Mettez en place cette nouvelle approche pour le cas du shell dans un fichier `cleanup_sh.l`.

3.5 Faites de même pour le cas du `C` dans un fichier `cleanup_c.l`.

Comparez le résultat avec celui obtenu précédemment.

3.6 [⋆] Écrivez un filtre `cleanup.l` qui, sur la donnée d'un texte sur l'entrée standard, supprime à la fois les commentaires `C` et les commentaires `C++` (de `//` inclus au saut de ligne suivant exclus).