TP 1 : Utilisation de Lex

J. Brunel, M. Couzinier, M. Strecker

Maîtrise IUP ISI

1 Transformation de HTML

Le langage HTML permet d'annoter un texte d'étiquettes qui spécifient comment certains éléments du texte sont affichés par un browser. Ainsi, un texte qui se trouve entre les étiquettes `<b>` et `</b>` est affiché en gras (*bold*), un texte entre `<i>` et `</i>` en italique. De la même manière, une liste numérotée (*ordered list*) se trouve entre `<ol>` et `</ol>`, une liste sans numérotation (*unordered list*) entre `<ul>` et `</ul>`, les éléments de la liste étant entre `<li>` et `</li>`. Un nouveau paragraphe est introduit par `<p>`.

Écrivez un fichier Lex `orderlist.l` qui transforme toute liste non numérotée en liste numérotée.

Le fichier aura la forme

```
%option main

%%
```

Dans la ligne suivant le `%%`, on écrit les expressions régulières et les actions correspondantes.

Lancez `lex orderlist.l`. Lex génère un fichier `lex.yy.c` avec une fonction `main` "standard" (à cause de `option main`). Après compilation (`gcc lex.yy.c -o orderlist`), vous obtenez un programme `orderlist` qui lit de l'entrée standard `stdin` et écrit sur la sortie standard `stdout`.

Vous pouvez donc tester votre programme sur le fichier `orderl_in.html` qui se trouve sur la page Web[1], en invoquant

```
orderlist < orderl_in.html > orderl_out.html
```

Étendez votre programme de façon à ce qu'il mette le mot *bold* en gras et le mot *italics* en italique. Comment (et jusqu'à quel point) peut-on éviter un étiquettage redondant (de la forme `<b><b> ... </b></b>`) ?

[1] http://www.irit.fr/recherches/TYPES/SVF/strecker//Teaching/M1_S8_IUP_ISI_Traduction/

2 Traduction de HTML en LaTeX

Dans le même style que l'exercice précédent, transformez un fichier HTML en LaTeX. Un document LaTeX est inclus entre

```
\documentclass{article}
\begin{document}
...
\end{document}
```

Le titre d'une section resp. sous-section est inclus entre `\section{...}` resp. `\subsection{...}`. Pour créer une liste numérotée, on utilise l'environnement

```
\begin{enumerate}
\item ....
\end{enumerate}
```

Pour une liste non numérotée, on utilise le mot-clé `itemize`.

Un nouveau paragraphe commence tout simplement avec une ligne vide.

Créez un ficher `html2tex.l` et procédez de la même manière que dans la première partie, puis testez votre programme en invoquant `latex` avec le fichier généré.

3 Conversion de nombres

Un fichier contient des nombres entiers codés dans différentes bases, selon la convention suivante :

- Les nombres octaux sont précédés par `0`. Exemple : `045`, correspondant à 37 en base 10.
- Les nombres décimaux s'écrivent "comme d'habitude". Pour ne pas les confondre avec des nombres en base 8, le premier chiffre ne doit pas être `0`, sauf s'il s'agit du nombre 0, dont la représentation est égale pour toute base.
- Les nombres hexadécimaux commencent avec `0x` et peuvent contenir les lettres `A` à `F` (majuscules ou minuscules), représentant les nombres 10 à 15. Exemple : `0x3A`, correspondant à 58 en base 10.

La transformation de chaînes de caractères représentant des nombres peut être accomplie à l'aide de la fonction C `sscanf`. Ainsi,

- `sscanf(string, "%x", &i)` interprète `string` comme un nombre en base 16, conformément aux conventions énoncées plus haut, et l'emmagasine dans la variable `i`.
- `sscanf(string, "%o", &i)` interprète `string` comme un nombre en base 8.

Procédez en 2 étapes :

1. Écrivez un traducteur, dans le style des exercices précédents, qui transforme tout nombre (quelle que soit sa base) en un nombre décimal. Testez votre programme !

   Rappel : La chaîne de caractères du terminal que Lex vient de reconnaître est emmagasinée dans la variable prédéfinie `yytext`. Avec `ECHO;`, on transfère la valeur de `yytext` au fichier de sortie de Lex.

2. Écrivez un traducteur qui ne lit pas de l'entrée standard, comme précédemment, mais d'un fichier, et écrit sur un autre fichier.

   Rappel : Lex gère deux variables, `yyin` et `yyout`, qui pointent sur le fichier de lecture resp. écriture de Lex (par défaut : `stdin` et `stdout`). Il vous faut donc initialisier ces deux variables correctement. Pour cela, effacez `%option main` et écrivez votre propre fonction `main`.

On appelera *mal formé* tout nombre

- qui commence avec `0`, mais n'est pas un nombre en base 8 valable.
  Exemple : `039`.
- qui commence avec `0x`, mais n'est pas un nombre en base 16 valable.
  Exemple : `0x9z`.

Si vous rencontrez un nombre mal formé, imprimez un message d'erreur sur la sortie standard. A la fin de la lecture d'un fichier, imprimez la somme de tous les nombres qui y sont contenus.