TP Analyse lexicale.

L'objectif de cette séance est de réaliser un analyseur lexical[1] pour une partie du langage Java[2] qui sera utilisé dans l'Unité d'Enseignement de Technologies Objets (UE TOB) au second semestre. Cette partie du langage sera étudiée dans une future séance de TD sur les grammaires. La partie lexicale du langage est spécifée par le document `http://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html`.

1 Analyseur lexical

Un analyseur lexical est la première étape de l'analyse d'un langage. Celle-ci est suivie par l'analyse syntaxique[3] qui sera l'objet des futures séances de TPs et l'analyse sémantique[4] qui sera l'objet de l'UE Sémantique et Traduction des Langages de la majeure Informatique en deuxième année.

Un analyseur lexical est spécifié sous la forme d'une séquence d'association entre une expression régulière et une action. L'action est effectuée lorsque l'analyseur lexical reconnait un mot qui fait parti du langage décrit par l'expression régulière associée. Si plusieurs expressions régulières reconnaissent le même mot, seule l'action associée est la première expression régulière est exécutée. Si plusieurs mots reconnus par des expressions régulières contiennent un même préfixe, alors l'action exécutée est celle associée à l'expression régulière qui reconnait le mot le plus long.

2 L'outil camllex

Pour réaliser cette séance, nous allons utiliser l'outil `camllex`[5] qui traduit la spécification d'un analyseur lexical en un programme `caml`[6] qui implante un analyseur lexical sous la forme d'un automate fini.

3 Etude d'un exemple

L'objectif est d'étuder l'exemple suivant en s'appuyant sur la documentation de l'outil `camllex` : `http://caml.inria.fr/pub/docs/manual-ocaml-4.02-1/lexyacc.html`. Pour compiler le programme, utilisez la commende `make` qui produit l'exécutable `mainJava`. Cet exécutable prend en paramètre le nom du fichier que l'on veut analyser.

4 Construction d'un analyseur lexical

L'objectif de cet exercice est de compléter le fichier fourni en exemple `lexerJava.mll` pour traiter l'ensemble des unités lexicales de Java telles qu'elles sont décrites dans la documentation : `http://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html`.

Vous vous attacherez à traiter les différentes formes de constantes (entiers, réels, caractères) ainsi que les différentes formes de commentaires.

[1] `http://en.wikipedia.org/wiki/Lexical_analysis`
[2] `http://en.wikipedia.org/wiki/Java_(programming_language)`
[3] `http://en.wikipedia.org/wiki/Parsing`
[4] `http://en.wikipedia.org/wiki/Semantics`
[5] `http://caml.inria.fr/pub/docs/manual-ocaml-4.02-1/lexyacc.html`
[6] `http://caml.inria.fr/pub/docs/manual-ocaml-4.02-1/`

Listing 1: Fichier `lexerJava.mll`

```
{

  open TokenJava
  exception LexicalError

}

let chiffre = ['0'-'9']
let alpha = ['a'-'z' 'A'-'Z']
let alphanum = alpha | chiffre | '_'

rule lexer = parse
  | ['\n' '\t' ' ']+     { lexer lexbuf }
  | ","                  { VIRG }
  | "int"                { INT }
  | "while"              { TANTQUE }
  | chiffre+ as texte    { ENTIER (int_of_string texte) }
  | eof                  { FIN }
  | _ as texte           { raise LexicalError }

{

}
```

Listing 2: Fichier `mainJava.ml`

```
open TokenJava;;

if (Array.length Sys.argv > 1)
then
  let lexbuf = (Lexing.from_channel (open_in Sys.argv.(1))) in
  let token = ref (LexerJava.lexer lexbuf) in
  while ((!token) != FIN) do
    (printToken (!token));
    (token := (LexerJava.lexer lexbuf))
  done
else
  (print_endline "MainJava fichier")
```