|  |  |
| --- | --- |
|  |  |

**Mini Projet de Théorie des Langages et Compilation**

**II1- 2015-2016**

L’objectif est d’étendre les connaissances en théorie des langages et des automates afin de pouvoir décrire des langages de programmation et leur analyse syntaxique en vue de leur compilation. Un mini compilateur est enfin mis en œuvre. Ce mini projet est divisé en trois parties :

**Partie 1. Développement d’un analyseur lexical**

Il est demandé ici de

1. Construire un automate fini qui accepte l’ensemble suivant

des mots (unités lexicales ou jetons, terminaux d’une grammaire) { Deb, Fin, Var, :, id, :=, ;, entier, reel, Lire, Ecrire, nb, nbr, +, \*, (, )}

Où ***id*** représente les identificateurs alphanumériques qui commencent par un caractère alphabétique (**l(l+c**)\*), ***nb*** représente les nombres entiers c+ et ***nbr*** représente les nombres réels c+.c+

1. Etendre l’automate pour pouvoir retourner à l’état final deux valeurs. La première valeur est l’unité trouvée selon l’état final atteint et la deuxième est un attribut supplémentaire.

Si l’unité trouvée est un id qui n’est pas un mot clé alors la première valeur retournée est id et la deuxième est une entrée dans une table contenant les identificateurs.

Si l’unité trouvée est un mot clé (sachant que les mots clés sont : Deb, Fin, Var, entier, reel (Il faut sauvegarder quelque part cette liste de mots), alors, les valeurs retournées sont le mot clé et 0.

Si l’unité est nb alors la deuxième valeur est la valeur de ce nombre. L’automate doit pouvoir sauter les espaces, les retours à la ligne et les tabulations.

1. Implémenter le parcours de cet automate par une fonction **(Anal\_lex)** qui retourne à chaque fois un enregistrement à deux champs contenant l’unité et un attribut (Symbole).

Voici un exemple d’automate :

Voici un exemple de résultat retourné par l’analyseur lexical

**Partie 2. Développement d’un analyseur syntaxique**

Soit la grammaire G suivante

G = (V, T, S, R)/

V = {S, L\_DCL, DCL, L\_INST, INST, Exp}

T = { Deb, Fin, Var, :, id, :=, ;, entier, reel, Lire, Ecrire, nb, nbr, +, \*, (, )}

1. Eliminer la récursivité à gauche et l’ambiguité si elles existent
2. Développer un analyseur syntaxique **prédictif récursif** (vu au cours et présenté dans le support numérique) qui fait appel à la fonction **Anal\_lex** pour analyser un programme existant dans un fichier texte.
3. Introduire le contrôle sémantique statique dans le code donné en 2.