Documentation

du projet de fin de semestre

Analyse Syntaxique 2020

MURATI Vidan L3 197841

Introduction

L'enseignement pourvu dans notre licence nous a permis au fil de ces 2 dernières années, de découvrir des langages de programmation variés.
Mais une question se posait : Par quel procédé notre ordinateur est il capable de comprendre puis d'éxecuter nos instructions malgré les, parfois grandes, différences de grammaire entre les langages ? L'analyse syntaxique nous donne une réponse partielle à cette question complexe.

Choix

J'ai décidé de redéfinir YY_INPUT pour l'affichage des lignes qui provoque une ou plusieurs erreurs. J'avais tout d'abord opté pour une première solution utilisant REJECT, mais il s'est avéré ensuite bien plus facile d'intégrer le curseur qui pointe le caractère qui provoque l'erreur en utilisant YY_INPUT plutôt que REJECT.
Je l'ai donc redéfinit comme suit :

- Je copie l'entiereté de la ligne courante dans une variable globale
- Je définit un curseur au debut de la ligne
- A chaque tour de boucle j'inscris le caractère présent a l'emplacement du curseur dans le buffer
- Quand le curseur est à la fin de la ligne : retour a l'étape 1

J'ai décidé de représenter une ligne comme étant un tableau d'au maximum 200 caractères, c'est un choix arbitraire.

Difficultés

Une des grandes difficulté de ce projet a été de maintenir la grammaire non ambigüe, surtout lors de l'ajout des structures. Il a aussi été difficile de comprendre l'ordre des opérations effectuées par flex et bison et en ce sens, correctement redéfinir YY_INPUT

J'ai aussi eu des difficultés a résoudre les conflits décalages/réductions de la grammaire ou encore à comprendre les examples donnés par bison pour faciliter leur résolution.

Bien que non demandé dans le rendu final, le redémarrage après erreur a aussi été source de conflits, et cette fonctionnalité n'est pas encore entièrement opérationnelle. (Après un redemarrage, certaines lignes détectées comme fausses par l'analyseur syntaxique sont pourtant syntaxiquement correctes.)

Les types

Ce langage reconnaît 3 différents types de variables :
CHAR, INT, et les types structurés

Les variables

Une variable peut être déclaré a l'intérieur de fonctions, comme à l'extérieur (mais pas après une déclaration de fonction).
Elles se déclarent comme suit :

```
<type_de_la_variable> <identificateur_de_la_variable>;
```

Les structures

Les structures sont définies comme suit :

```
struct <identificateur_de_la_structure> {
<déclarations de variables>
};
```

elles peuvent êtres définies avant et après les variables globales mais seulement avant la déclaration de fonctions.
Les structures sont déclarées comme suit :

```
struct <identificateur_de_la_structure> <idenficateur_de_variable> ;
```

Les instructions

De nombreuses instructions sont disponibles, (if, while, print) (voir grammaire).
Une instruction doit impérativement terminer par un ;

Les fonctions

Une fonction est définie comme suit :

```
<type_de_retour> <identificateur_de_la_fonction> ( <parametres> ) {
      <instructions>
}
```

Elle sont déclarées a la fin du programme soit après les structures et les déclarations de variables globales.

Pour + d'informations sur le langage voir

`sujet-projet-AS-2020-2021.pdf`

Conclusion

La définition d'éléments lexicaux couplé a une analyse syntaxique nous permettent de séparer les caractères d'un texte et de les regrouper en différents éléments porteur de sens. C'est ainsi qu'on définit un langage.
Il nous reste maintenant à exprimer le langage engendré en un ensemble d'instructions que l'ordinateur sera capable d'interpréter.

J'ai trouvé le projet très intéressant, car il permets de comprendre le langage, d'un point de vue très différent de ce que l'on a l'habitude d'étudier quand on parle de langues.
Et malgré le caractère très 'informatique' de l'analyse syntaxique, j'ai l'impression que ces règles et méthodes sont aussi très présentes dans les langues parlées.