Projet Analyse Syntaxique

Table des matières

Introduction

○ *Objectif*

Le but de ce projet est de construire un analyseur syntaxique permettant de reconnaître le langage de programmation TPC, en utilisant flex et bison.

Il fallait afficher la ligne d'erreur ainsi que le caractère qui était la source de l'erreur à l'aide d'une flèche qui montrait ce même caractère.

De plus, il fallait ajouter dans la grammaire bison de quoi détecter les structures.

Développement

○ *Choix d'implémentation*

Tout d'abord nous avons écrit un programme flex permettant de reconnaître la grammaire donnée pour ce projet.

Reconnaître les commentaires :

Pour reconnaître les commentaires nous utilisons une condition de démarrage exclusive déclarée avec '%x'.

Comptage des lignes et caractères :

Pour connaître la ligne et à quel caractère se situe une erreur s'il y en a une nous utilisons 2 variables permettant de compter efficacement la ligne et le caractère :

- Pour la ligne on incrémente le compteur qui lui est associé à chaque rencontre avec un '\n'

- Pour le caractère on incrémente son compteur à chaque mots-clés, caractère, identificateur…

  rencontré par le nombre de caractères que contient celui-ci. Le compteur est remis à 0 lorsque l'on

  rencontre un saut de ligne '\n'.

Affichage de la ligne d'erreur :

En ce qui concerne le stockage de la ligne en cas d'erreur nous utilisons la méthode 'REJECT' du cours.

Nous définissons une variable qui représente un tableau de char afin de recopier grâce à la fonction 'strcpy' yytext dans la variable définie.

L'instruction qui réalise cette tâche ne copie que la ligne d'erreur.

Pour l'affichage il ne reste plus qu'à afficher la variable dans le fichier bison dans la fonction yyerror.

Affichage de la flèche d'erreur :

Pour l'affichage de la flèche nous utilisons tout simplement la variable qui compte la position du caractère où se situe l'erreur.

Implémentation des structures :

Pour que notre programme puisse reconnaître les structures il nous a fallu rajouter dans la grammaire bison le code suivant :

```
DeclVarsGlob:
        DeclVarsGlob TYPE Declarateurs ';'
    |   DeclVarsGlob STRUCT IDENT CorpsS
    |
    ;
DeclVars:
        DeclVars TYPE Declarateurs ';'
    |   DeclVars STRUCT IDENT Declarateurs ';'
    |
    ;
```

et de rajouter dans le fichier flex l'instruction permettant de reconnaître une structure.

Réglage du conflit de la grammaire :

Dans la grammaire donnée lors de la compilation il y avait un conflit pour y remédier nous avons ajouté dans le fichier bison la ligne :

```
%expect 1
```

Affichage en cas d'erreur sur la sortie standard :

En cas d'erreur dans un programme TPC l'affichage sur la sortie standard se fait de la manière suivante :

```
Fichier trinomeV9.tpc
syntax error near line 1, character 1:
/ int trinome(float a, float b, float c) ;
^
```

En première ligne nous avons le nom du fichier où il y a l'erreur, suivi du message d'erreur qui indique la ligne d'erreur et le caractère à laquelle l'erreur se situe.

A la troisième ligne nous avons la ligne de code où il y a l'erreur avec une flèche qui pointe vers le caractère où se situe l'erreur.

Dans le cas ou il n'y a pas d'erreur l'affichage se fait par le nom du fichier suivi du caractère '0'.

Conclusion

○ *Compilation*

L’utilisation du programme se fait de la manière suivante :

Lancer dans un terminal la commande : **make**

Puis la commande : **./Test.sh**

○ *Difficultés rencontrées*

Nous avons rencontrés des difficultés lors de la création de la grammaire pour détecter les structures.

Nous avions au début plusieurs conflits ce qui n'est plus le cas avec la grammaire que l'on utilise actuellement.