Sylvain Tran

Rapport : Projet Analyse Syntaxique

1) Mes choix

-Dans le module Tree :

Pour représenter les attributs des nœuds, j'ai ajouté dans **Node** une **union** 'attr' pouvant supporter comme types :
int num → pour la valeur des constantes numériques.
char op_bytes → pour les opérateurs : +, -, *, /, %, = et !.
char op[3] → pour les opérateurs de comparaisons et booléens : ==, !=, <, <=, >, >=, | |, &&.
char name[64] → pour tous les autres types d'attributs : mots-clés (void, if, else, while, return), caractères ('a', 'b', '\n', etc.), types (int, char) et enfin les identificateurs.

J'ai également ajouté un type **enum** 'type_attr' qui va nous permettre d'identifier le type de l'attribut dans le nœud.
Donc typiquement par exemple, si on a un INT comme type d'attribut, alors on va remplir le champs num de 'attr'.

Pour créer mes nœuds, j'ai changé la fonction **makeNode()** pour en faire une fonction à nombre variable d'arguments (avec les …) et comme on veut des nœuds différents et qu'on ne sait pas à priori quelle sera le type de l'attribut, j'ai décidé d'employé un paramètre « pivot » qui est de type 'type_attr'. Selon sa valeur, on va cibler le champ de l'union 'attr' qui nous intéresse.

Comme les arguments supplémentaires sont optionnels, pour créer un nœud de type Node,
il suffit d'écrire : makeNode(label_au_choix, NODE);

-Pour les Messages d'Erreurs :
J'ai décidé de faire comme les compilateur en C :

Quand il s'agit d'une erreur lexicale : le numéro dans la ligne du caractère proche de l'erreur c'est le premier caractère du lexème causant l'erreur. Idem lorsqu'il s'agit d'une erreur syntaxique.

2) Difficultés rencontrées

La seul problème que j'ai eu, c'est qu'à un moment j'ai eu une segfault qui m'a contraint de modifier les règles de grammaire du non-terminal InitVars (sans pour autant modifié le langage engendré).
Je suis passé de ça:

```
    ;
DeclarateursLocal:
      DeclarateursLocal ',' IDENT InitVars
    | IDENT InitVars
    ;
InitVars:
      '=' Exp
    |
    ;
```

à :

```
DeclarateursLocal:
      DeclarateursLocal ',' InitVars
    | InitVars
    ;
InitVars:
      IDENT '=' Exp
    | IDENT
    ;
```