Université Bordeaux 1 Licence Sciences et Technologies

Projet d'Analyse Syntaxique - Annee 2010-2011

Le projet consiste à écrire un analyseur qui analyse du code C, en rassemblant dans un même fichier `yacc` les règles permettant de déclarer des variables, d'imbriquer des blocs d'instructions et d'écrire des affectations.

Etape 1

Reprendre l'exercice 6.3 de la feuille 6, en vous aidant éventuellement du corrigé.
Modifier la table de symbole, pour qu'elle contienne pour chaque variable son nom , son type et sa valeur. Les valeurs seront initialisés à zéro. Les types seront tous DOUBLE dans cette étape, DOUBLE étant une pseudo-constante. Le programme devra afficher en fin d'analyse la liste de tous les identificateurs, avec leur type et leur valeur.

Etape 2

Ajouter des règles de grammaires permettant de reconnaître des blocs d'instructions, éventuellement imbriquées. Le bloc externe se terminera par une instruction de la forme `return expression ;` et le progamme devra calculer la valeur de cette dernière expression.

Etape 3 :

Le bloc externe devra maintenant commencer par des déclarations de variables. On considèrera deux types de variables : les réels et les pointeurs sur réels. Une déclaration de pointeur aura la forme suivante :
`double *p ;`
Rajouter les règles de grammaires correspondant aux déclarations de pointeurs. Modifier le champ `type` de l'attribut d'une variable en fonction de sa déclaration (on utilisera une pseudo-constante POINT pour le type pointeur).

Etape 4 :

Une initialisation de pointeur aura la forme suivante :
`p = &x ;`
où x sera une variable de type réel déclarée auparavant.
Modifier les règles des affectations afin de prendre en compte les initialisations de pointeurs. Il sera nécessaire, dans les attributs d'une variable pointeur de mémoriser le réel x pointé.
Modifier les règles des expressions arithmétiques, pour qu'elles puissent utiliser (*p), où p est un pointeur sur réel. Exemple :

```
double x; double y;
double *p;

x = 5/3;
y = 7.5 + 2*(x - 1);
p = &x;
x = 33;
y = 12 + (*p);
```

Etape 6 facultative :

Vous pourrez apporter des améliorations à votre programme. Par exemple :
- vérifier lors d'une affectation, que la variable a bien été déclarée,
- introduire des fonctions mathématiques simples (`sin`, `cos`, ...) dans les calculs arithmétiques.