Analyse Syntaxique - Annee 2009-2010
Feuille 6

Construction d'analyseurs syntaxiques avec `yacc`

Exercice 6.1

Soit G la grammaire suivante écrite dans le format reconnu par `yacc`, :

```
%token ID CHAR INT DOUBLE STRUCT NUM
%%
declar : type ID tab ';'
       ;

type : CHAR
     | INT
     | DOUBLE
     | STRUCT '{' list '}'
     ;

list : list declar
     |
     ;

tab : '[' NUM ']'
    |
    ;
%%
```

Afin de réserver l'espace mémoire correct dans la pile lorsqu'un identificateur est déclaré, on veut connaître la taille en octets de chaque variable.
Ajoutez dans le fichier `yacc` des règles de calcul d'attributs qui permettent de connaître la taille en octets d'une variable déclarée selon les règles de cette grammaire. Lorsque l'analyseur a reconnu une déclaration de variable, il devra afficher le nombre d'octets nécessaires à l'initialisation de cette variable.
On considèrera qu'un entier ou un caractère occupe 1 octet, un réel double occupe 4 octets. Une variable de type structure occupe la place nécessaire pour tous les champs de la structure. Une variable de type tableau de taille n occupe la place nécessaire pour les n cases du tableau. On supposera que l'attribut du token `NUM` a été initialisé avec sa valeur dans le fichier `lex`.

Exercice 6.2

Soit la grammaire suivante qui permet de calculer des sommes d'entiers, y compris des sommes de fonctions entières sur un intervalle. Par exemple :

`x = somme (i = 1 .. 3, f(i))`.
On considère seulement deux fonctions : $carre(i)$ et $cube(i)$. La grammaire est donnée dans le format reconnu par `yacc` :

```
%token ID  NUM INT SOMME CARRE CUBE
%left PLUS

%%
axiom : E ';'
;

E :  E  PLUS E
  |  S
  |  INT
;

S : SOMME '(' ID '='  INT  '.''.'  INT  ','  F '(' ID ')'  ')'
;

F : CARRE
  | CUBE
;

%%
```

Ajouter pour chaque règle de grammaire, un calcul d'attributs qui permette d'évaluer les expressions. Le résultat sera afficher lors de l'action associée à la règle de l'axiome..

Exercice 6.3

1. Reprendre l'exercice 3 de la feuille 5, avec les attributs des nombres qui sont de type `double` et les attributs des variables qui sont des couples (`nom`, `valeur`). Creer dans le fichier `yacc` une table de symboles, destinée à contenir les attributs des variables.
   On pourra utiliser les fichiers `symboles.c`, `symboles.h` pour le traitement de la table de symboles (recherche et insertion).
   Le programme devra afficher en fin d'analyse la liste de tous les identificateurs, avec leur valeur.
2. Modifier votre analyseur pour pouvoir évaluer des expressions contenant des variables auquelles on a précédemment affecté des valeurs. Exemple :
   $x = 9.3$;
   $y = 1.5 + 17 * x$;.

Exercice 6.4

Écrire un source `lex` et `yacc` permettant de reconnaître les expressions booléennes correctement formées, utilisant l'alphabet de terminaux $\{0, 1, (,), \vee, \wedge, \neg\}$. Le programme devra évaluer la valeur de l'expression. On utilisera une grammaire qui respecte la priorité des opérateurs.