Arbres abstraits et table des symboles

Exercice 1 : Construction des arbres abstraits

Cet exercice propose une méthode pour construire les arbres abstraits à partir d'une grammaire.

On rappelle qu'un *arbre abstrait* (ou encore AST) est un arbre qui ne garde plus trace des détails de l'analyse syntaxique d'un texte source, c'est à dire qu'il est indépendant de la grammaire, mais il mémorise la structure du programme lu en entrée.
On va dans un premier temps définir une *grammaire abstraite* , qui peut être vue comme une *grammaire d'arbres*.

Soit une grammaire dont l'axiome est expleC et qui reconnait un sous-ensemble d'instructions du langage C : elle est présentée page suivante, dans le fichier expleC.g, fichier défini au format Antlr.

▷ **Question 1.** Définir la *grammaire abstraite* correspondant à ce langage.

▷ **Question 2.** Dessiner ensuite l'AST correspondant au texte source suivant :

```
char c;
int x;
int f(int y, char a) {
 int i;
 for(i=0; i!=3; i=i+1) {
   x = 3;
   y = 5;
 }
}
```

Fichier `expleC.g`:

```
grammar expleC;

program
    : declaration+
    ;
declaration
    : variable
    | function
    ;
variable
    : type ID ';'
    ;
type
    : 'int'
    | 'char'
    ;
function
    : type ID '(' (formalParam (',' formalParam)* )? ')'  block
    ;
formalParam
    : type ID
    ;
block
    : '{' variable* inst* '}'
    ;
inst
    : forInst
    | expr ';'
    | block
    | affect ';'
    | ';'
    ;
forInst
    : 'for' '(' affect ';' expr ';' affect ')' block
    ;
affect
    : ID '=' expr
    ;
expr
    : condExpr
    ;
condExpr
    : expr1 ( ('==' | '!=') expr1 )?
    ;
expr1
    : expr2 ('+' expr2)*
    ;
expr2
    : atom ('*' atom)*
    ;
atom
    : ID
    | INT
    | '(' expr ')'
    ;
// définitions des expressions régulières reconnaissant les tokens
ID  : ('a'..'z' | 'A'..'Z')('a'..'z' | 'A'..'Z'|'0'..'9')*
    ;
INT : ('0'..'9')+
    ;
WS  : [ \n\t\r]+ ->skip ;
```

Exercice 2 : Application à un exemple de programme Java

Soit le programme Java suivant :

```
public class MyString{

 // attributs
 private String content;
 private int nb;

 // constructeur
 public MyString(String content,int i){
    this.content=content;
    nb=i;
 }

 // fonctions
 public String toString(){
    String res="content"+"("+nb+"):="+content+";";
    return res;
 }

 // main
 public static void main(String[] args){
    MyString test=new MyString("test",1);
    System.out.println(test);
 }
}
```

Dessinez un AST de ce programme.
Pour cela, la racine pourra être le nom de la classe (`MyString`), et vous pourrez créer les noeuds `ATTR` pour les attributs, `CONS` pour les constructeurs, `FUNCTS` pour les fonctions et `MAIN` pour la fonction main (d'autres noeuds seront certainement à définir...) Vous dessinerez ces 4 sous-arbres.

Remarque : on suppose qu'on ne construit pas de TDS pendant l'analyse syntaxique. L'AST construit comprendra donc l'ensemble des déclarations du programme.

Exercice 3 : Table des symboles

Dans cet exercice, on vous demande de réfléchir autour de la structure et de l'utilisation d'un AST et de la TDS pour effectuer des contrôles sémantiques.

Soit le code Java suivant :

```
1  public class Person {
2          private String firstName;
3          private String lastName;
4          private int age;
5          private String firstName;
6
7          public Person(String f, String l) {
8                  firstName = 2;
9                  lastName = l;
10                 age = a;
11         }
12         public String getFirstName() {
13                 return firstName;
14         }
15 }
```

1. Construisez l'AST ainsi que la ou les TDS associés à ce code.
2. Ce code contient 3 erreurs sémantiques, décrivez-les. On attend le numéro de ligne et une courte description expliquant pourquoi l'instruction est incorrecte (une phrase maximum).
3. Pour chacune des 3 erreurs sémantiques, on veut savoir comment un compilateur pourrait les repérer. Pour cela, et pour chacune des 3 erreurs, répondez aux 3 questions :
   (a) A quel(s) noeud(s) dans l'arbre le contrôle s'effectue ?
   (b) Sur quelle(s) structure(s) (AST, TDS, Pile Régions Ouvertes) s'appuie le contrôle ?
   (c) Décrivez le processus du contrôle (3 lignes maximum).