Exercices dirigés
séance n°1

Exercice 1 : compilation, analyse d'un programme

On définit le langage simplifié suivant au moyen de règles BNF :

```
<phrase> ::= <déclaration> | <instruction>
<déclaration> ::= <identificateur> : <type> ;
<type> ::= réel | entier
<instruction> ::= <identificateur> = <nombre> ;
<nombre> ::= <nombre entier> | <nombre réel>
```

Les identificateurs sont des unités lexicales constituées d'une lettre.
Les nombres entiers sont des unités lexicales constituées d'un chiffre et les nombres réels des unités lexicales constituées de deux chiffres séparés par une virgule.

Question 1

Rappeler les différentes phases de l'analyse d'un programme et leur rôle.

Question 2 : analyse lexicale

Un chiffre et une lettre sont définis par :

```
<chiffre> ::= 0 | 1 | 2 | ….| 9
<lettre>  ::= A | B …| Z
```

- Définir sous forme de règles BNF identificateur, nombre entier et nombre réel.
- Quelles sont les unités lexicales du langage autres que identificateur, nombre entier, nombre réel ?

L'espace est sans signification mais ne peut se trouver à l'intérieur d'une unité lexicale.
Faire l'analyse lexicale de chaque ligne (phrase) du programme :

```
A : reel $
B : entier ;
= A 5,3,2 ;
A B = 6,2 ;
C = 4 ;
```

Pour cela, on recopiera les lignes en encadrant chaque unité lexicale reconnue par l'analyseur et on identifiera les erreurs lexicales en justifiant pourquoi ce sont des erreurs.

Question 3 : analyse syntaxique

On considère le programme suivant lexicalement correct :

```
A : reel ;
B : entier ;
= A 5,3 ;
A B = 6,2 ;
C = 4 ;
```

Décrire sous forme d'arbres, l'analyse syntaxique de chaque ligne du programme. On commentera les erreurs syntaxiques rencontrées.

Question 4 : analyse sémantique

On considère le programme suivant lexicalement et syntaxiquement correct :

```
A : réel ;
B : entier ;
A = 5,3 ;
B = 6,2 ;
C = 4 ;
```

Un programme est en fait défini syntaxiquement comme une suite de déclarations, suivie d'une suite d'instructions :

```
<programme> ::= {<déclaration>} {<instruction>}
```

Le programme ci-dessus est correct vis à vis de cette définition.
On introduit les règles suivantes pour l'analyse sémantique :

- tout identificateur utilisé dans une instruction doit être déclaré.
- dans une instruction, l'identificateur à gauche du = et le nombre à droite doivent être de même type.

Quelles sont les erreurs signalées par l'analyse sémantique du programme ? Justifier la réponse.

Exercice 2 : une grammaire pour exprimer des dates

On souhaite écrire une grammaire pour définir un petit langage permettant d'exprimer des dates sous les formes suivantes :

- `jj/mm/aaaa`
- `mm/jj/aaaa`
- `jj mois aaaa`
- `mois jj aaaa`

Les symboles du vocabulaire terminal de ce langage sont les caractères alphanumériques.

Question 1

Parmi les dates qui suivent, lesquelles seraient valides dans ce langage :

`12/08/1992,13/15/2000,05/30/2002,08 juillet 1954,juin 43 1912, avril 23 1964,38/09/3456`

Question 2

Ecrire une grammaire pour ce langage

Exercice 3 : une grammaire simplifiée pour les expressions booléennes

```
<expr_bool> ::= <facteur> or <expr_bool> |<facteur>
<facteur> ::= <terme> and <facteur> |<terme>
<terme> ::= <identificateur>
            |<valeur>
            |(<expr_bool>)
            |not <terme>
<identificateur> ::= <lettre>{<lettre>}
<valeur> ::= vrai|faux
<lettre> ::= 'a'…'z'
```

Question 1

Les expressions suivantes sont elles syntaxiquement valides ?

```
a and b or not (a or c)
a and (b or not a) or c
```

Question 2

Exprimer l'analyse des expressions précédentes sous la forme d'arbres syntaxiques.

Question 3

Evaluer les expressions ci-dessus à partir de l'arbre syntaxique lorsque :

```
a=vrai, b=faux, c=vrai
```

Exercice 4 : un petit langage de programmation

Soit la grammaire définissant un petit langage de programmation :

```
<programme> ::= <instruction>{<instruction>}
<instruction> ::= <affectation>|<conditionnelle>
<conditionnelle> ::= ? <condition> : <affectation>
<condition> ::= <expression> (=|#) <expression>
<affectation> ::= <identificateur> = <expression>
<expression> ::= <facteur> (+|-) <expression>|<facteur>
<facteur> ::= <terme> (*|/) <facteur>|<terme>
<terme> ::= <identificateur>|<nombre>|(<expression>)
<identificateur> ::= 'a'…'z'
<nombre> ::= <chiffre><nombre>|<chiffre>
<chiffre> ::= 0…9
```

Les symboles terminaux sont en gras.

Question

Les programmes suivants sont-ils syntaxiquement corrects ? Tracer l'arbre syntaxique. Si l'analyse n'aboutit pas, proposer une forme syntaxiquement valide.

Chaque ligne est considérée comme un programme :

```
c=6
b=c+(?(c=a):b=c)
b=?(c#a):b=a
b=b*a
?(b+c)#0:b=c
y=+a/z
```