Grammaire Officielle du Langage LNG-255
Document Complet et Détaillé

1. Programme

```
program :=
      (import_statement | declaration | statement)* EOF

- Un programme est composé d’une suite :
  • d’imports
  • de déclarations (variables, constantes, fonctions, classes)
  • d’instructions
- EOF indique la fin du fichier.

Exemple :
    import math;
    var x = 10;
    print(x);
```

2. Imports

```
import_statement :=
      'import' IDENT ('.' IDENT)* ('as' IDENT)? ';'

- 'import' : mot-clé pour importer un module.
- IDENT : nom du module.
- '.' IDENT : import d’un sous-module.
- 'as' IDENT : alias (optionnel).
- ';' : fin d’instruction.

Exemples :
    import math;
    import util.string as str;
```

3. Déclarations globales

```
declaration :=
      variable_declaration
    | constant_declaration
    | function_declaration
    | class_declaration

C’est tout ce qui peut être déclaré au niveau global du programme.
```

4. Variables

```
variable_declaration :=
      'var' IDENT (':' type)? ('=' expression)? ';'

Explication :
- var : mot-clé de déclaration classique.
- IDENT : nom de variable.
- : type : optionnel → typage statique.
- = expression : valeur initiale optionnelle.
- ; : fin.
```

```
Exemples :
    var age: int = 20;
    var nom = "Luna";
    var compteur;
```

Constantes

```
constant_declaration :=
      'const' IDENT (':' type)? '=' expression ';'

- 'const' : crée une valeur immuable.
- Typage optionnel.
- Une constante DOIT avoir une valeur.

Exemples :
    const PI: float = 3.14;
    const VERSION = "1.0.0";
```

5. Types

```
type :=
      'int' | 'float' | 'string' | 'bool' | 'char'
    | IDENT
    | type '[]'
    | '{' (IDENT ':' type (',' IDENT ':' type)*)? '}'

- Types natifs : int, float, string, bool, char
- IDENT : type personnalisé (class, alias…)
- type[] : tableau dynamique
- { field: type, ... } : type objet / struct

Exemples :
    int
    string[]
    {x:int, y:int}
```

6. Fonctions

```
function_declaration :=
      'function' IDENT '(' parameter_list? ')' (':' type)? block

- 'function' : mot-clé
- IDENT : nom
- '(' param_list ')' : paramètres
- : type : type de retour optionnel
- block : corps de fonction

Exemple :
    function add(a:int, b:int): int {
        return a+b;
    }
```

7. Classes

```
class_declaration :=
      'class' IDENT ('extends' IDENT)? '{' class_member* '}'

- class : mot-clé
```

```
- extends : héritage optionnel
- class_member : variable, méthode ou constructeur

Exemple :
    class Point {
        var x:int;
        var y:int;
        function move(dx:int){ x += dx; }
    }
```

Constructeur

```
constructor_declaration :=
      'constructor' '(' parameter_list? ')' block

Exemple :
    class Person {
        var name;
        constructor(n) { name = n; }
    }
```

8. Statements

```
statement :=
      block
    | expression_statement
    | if_statement
    | while_statement
    | for_statement
    | return_statement
    | break_statement
    | continue_statement

Instruction générique du langage.
```

9. Conditions

```
if_statement :=
      'if' '(' expression ')' statement ('else' statement)?

Syntaxe classique.

Exemple :
    if (x > 10) print(x); else print(0);
```

10. Boucles

```
while_statement :=
      'while' '(' expression ')' statement

for_statement :=
      'for' '(' (variable_declaration|expression_statement|';')
                expression? ';'
                expression_statement? ')' statement

Exemples :
    while (x < 10) x++;
    for (var i=0; i<10; i=i+1)
```

```
        print(i);
```

11. Contrôle de flux

```
return_statement := 'return' expression? ';'
break_statement := 'break' ';'
continue_statement := 'continue' ';'
```

12–18. Expressions (toutes les catégories)

```
Toute expression suit une hiérarchie stricte :
1. assignment_expression
2. logical_or_expression
3. logical_and_expression
4. equality_expression
5. relational_expression
6. additive_expression
7. multiplicative_expression
8. unary_expression
9. primary_expression

Exemples :
    a = 3 + 4 * 2
    x += 10
    !flag
    (a + b) * c
```

19. Tableaux et objets

```
array_literal := '[' (expression (',' expression)*)? ']'
object_literal := '{' (IDENT ':' expression (',' IDENT ':' expression)*)? '}'

Exemples :
    [1,2,3]
    {name:"Zed", age:30}
```

20. Appels de fonction

```
IDENT '(' argument_list? ')'

Exemple :
    add(3, 5)
```

21. Commentaires

```
comment :=
      '//' ... endline
    | '/*' ... '*/'

Exemples :
    // commentaire
    /* commentaire multi-ligne */
```

Exemple d'arbre binaire

Pour l’expression : a = 3 + 4 * 2