LAB 5: Abstract Syntax Tree (AST) Generation

Vo Hoang Nhat Khang

Examples:

Example 1:

Consider the following grammar for a simple program:

```
program
    : ( expression )* EOF
    ;

expression
    : Integer
    | Identifier
    ;

Integer: [0-9]+ ;
Identifier: [a-z]+ ;
```

This grammar defines a program consisting of zero or more expressions, where each expression can be either an integer or an identifier. Generate the AST for the grammar.

Example 2:

Consider the following grammar for expressions involving addition:

```
program
    : ( expression )* EOF
    ;

expression
    : expression '+' term
    | term
    ;

term
    : Integer
    | Identifier
    ;

Integer: [0-9]+ ;
Identifier: [a-z]+ ;
```

This grammar defines a program consisting of zero or more expressions, where each expression can be an addition operation between two terms, or simply a term itself. Generate the AST for the grammar.

Exercises:

Exercise 1:

Generate the AST for the following grammar, which defines expressions involving addition, subtraction, multiplication, and division:

```
program
    : ( expression )* EOF
    ;

expression
    : expression (Add|Sub) factor
    | factor
    ;

factor
    : factor (Mul|Div) term
    | term
    ;

Add : '+';
Sub : '-';
Mul : '*';
Div : '/';

term
    : Integer
    | Identifier
    ;

Integer: [0-9]+ ;
Identifier: [a-z]+ ;
```

Ensure correct handling of precedence and associativity for each operator.

Input 1:

```
5 + 4 * 3 - x / 2
```

Output 1:

```
Program(
    BinOp(
        BinOp(
            Integer(5),
            '+',
            BinOp(
                Integer(4),
                '*',
                Integer(3)
            )
        ),
        '-',
        BinOp(
            Identifier(x),
            '/',
            Integer(2)
        )
```

```
    )
)
```

Input 2:

```
2 + 3 * 4 / 5 - x
```

Output 2:

```
Program(
    BinOp(
        BinOp(
            Integer(2),
                '+',
                BinOp(
                    BinOp(
                        Integer(3),
                        '*',
                        Integer(4)
                    ),
                    '/',
                    Integer(5)
                )
            ),
        '-',
        Identifier(x)
    )
)
```

Exercise 2:

Generate the AST for the following grammar, which defines statements declaring variables of integer or float types:

```
program
    : ( statement )* EOF
    ;

statement
    : (IntType | FloatType) Identifier (',' Identifier)* ';'
    ;

IntType: 'int';
FloatType: 'float';
Identifier: [a-z]+ ;
```

Ensure that variables are correctly declared with the specified types and that multiple variables can be declared in a single statement.

Input 1:

```
int x, y, z;
```

Output 1:

```
Program(
```

```
    Statement(
        IntType,
        [
            Identifier(x),
            Identifier(y),
            Identifier(z)
        ]
    )
)
```

Input 2:

```
float a, b;
```

Output 2:

```
Program(
    Statement(
        FloatType,
        [
            Identifier(a),
            Identifier(b)
        ]
    )
)
```