CSL302: Compiler Design

Assignment-2

Due Date: 10-October-2025

Logistics on Submissions:

- Implement your solution using the syntax analyzer tool, for example yacc.
- The assignment teams have to be the same as that of Assignment-1.
- Prepare your submission in a zip file and name it as <ROLL NO1_ROLL NO2>.zip. If the team size is other than two, name your submission accordingly.
- Your submission should include a README file containing the instructions to execute your program(s).
- Upload your assignment in the canvas portal.
- Note that the weightage of each assignment will be different.
- It is sufficient if one member of the team submits the file.
- Any form of plagiarism is strictly prohibited; all the text must be your own, and all the references must be cited.

Assignment Problem Statement:

In this assignment, you are required to implement the syntax analyzer for the same language as used in lexical analyzer. You can reuse the lexical analyzer developed in Assignment-1 for solving this assignment.

The programming language supports the following features.

Language Specifications

Program Structure

- Programs consist of sequences of declarations. Where each declaration can be a variable declaration (VarDecl), function declaration (FunDecl), or class declaration (ClassDecl). They are listed below.
- A program must include a **main** function at the global scope, with no parameters and a **void** return type, which serves as the entry point.

Keywords and Identifiers, Constants, Operators, and Comments have the same features listed in Assignment-1.

Declarations:

The following declarations are supported. The Type can be **int, double, bool**, **string**, or **void**.

(a): Variable Declaration (VarDecl):

Simple variable declaration: Type Identifier;

Array declaration: Identifier NewArray(N, Type);

(b): Function Declaration (FunDecl): Type Identifier (FormalParameters) StmtBlock

Formal parameters can be empty (or) it can be listed a list Type followed Identifier. The details of StmtBlock are discussed in the Expression and Statements section below.

(c): Class Declaration (ClassDecl): class Identifier { Field* }, where a Field can be a VariableDecl or a FunctionDecl.

Expressions and Statements:

Expression (EXP):

- Expressions support assignments (=), arithmetic operators **(+, -, *, /, %)**, relational **(<, <=, >, >=)**, equality **(==, !=)**, logical operators **(&&, ||, !)**, unary minus, parentheses, and function/method calls.
- Assignment expression should have Lvalue=EXP, where Lvalue can be identifier, EXP.identifier, or EXP [EXP]
- Expression can also have **call** which have the format **Identifier (ActualParameters) | EXP.Identifier (Actuals)**. The Actual parameters is a list of EXP separated by comma.
- Expression can have **New (Identifier)** for object creation.
- Expressions respect the following operator precedence (highest to lowest):
    a. Member and array access (., [ ])
    b. Unary operators (!, unary -)
    c. Multiplicative operators (*, /, %)
    d. Additive operators (+, -)

e. Relational operators (<, <=, >, >=)

f. Equality operators (==, !=)

g. Logical AND (&&)

h. Logical OR (||)

i. Assignment (=)

Statement (Stmt):

<EXP> ; | IfStmt, | WhileStmt | ForStmt | BreakStmt | ReturnStmt | StmtBlock

Blocks (StmtBlock)

- A block group is zero or more statements (including variable declarations) enclosed in **{** and **}.** In other words, Block is {VariableDecl* Stmt* }

Conditional Statements

- **If: if** (EXP) Stmt
- **If-else:**
  if (EXP) Stmt else Stmt

Loop Statements

- **While loop:** while (EXP) Stmt
- **Do-while loop:** do Stmt while (EXP);
- **For loop:** for (EXP; EXP ; EXP) Stmt
  The for-loop includes optional initialization, condition, and iteration expressions, separated by semicolons, supporting counting or iterative loops.

Other Statements

- Return statement: return EXP ;
- Break statement: break ;