SER-502 Project Team-9

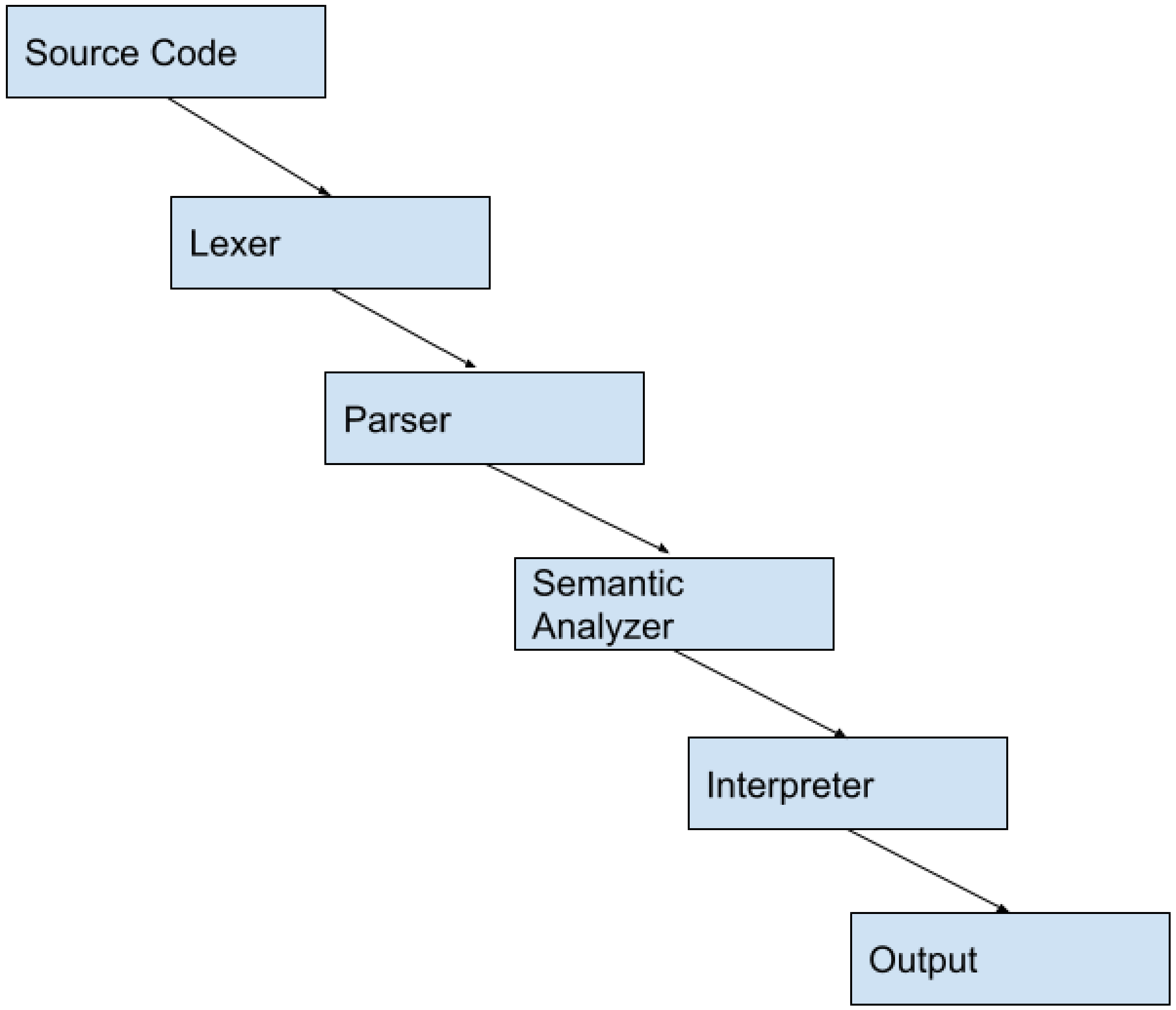
Milestone-1

Members:
Vinesh Reddy Naga
Venkata Vamsikrishna Iytha
Sai Kumar Chunchu
Yogaleena Mandalapu

Name of the Language: **SIMPLE**

Tools/Languages Used:
SWI Prolog is used for all components of the project, namely Lexer, Parser, Evaluator & Interpreter.

Design:



Lexer:
We are using **Prolog** to perform lexical analysis. User provides the source code in a file and the prolog will open the file and stream tokens from the source code. The end goal of this lexer is to generate a list of tokens/lexemes from the source code.

Parser:
We are using **Prolog** to generate a parse tree from the list of tokens generated by lexer. From our experience with Prolog, writing a parser in Prolog was an easy choice. Also we want to implement our learning from this course. We are employing the top down parsing technique in our parser.

Evaluator/Interpreter:
We are using **Prolog** to assign semantics to the above generated parse tree. Then we will use eval predicate to execute the intermediate code and display an output.

Data Structures used in SIMPLE :
List in Prolog.

Features and usage of the language:

Data Types:
- SIMPLE supports Integer (int), String (string), Boolean(bool) data types.
- Definitions of Int, String and Boolean are similar to that of C++.

Operators:
- SIMPLE supports assignment (=), addition, subtraction, multiplication and division operators.
- SIMPLE supports '<','>','<=','>=','!=','==' comparison operators.
- SIMPLE supports '++', '--' operators.
- SIMPLE supports 'and', 'or', 'not' for Boolean identifiers.
- SIMPLE supports ternary operator '?:'.

Control Structures:
- SIMPLE supports if-then-else conditional structure.
- SIMPLE supports traditional for loop, while loop and a for in range(number,number) loop.

Miscellaneous:
- SIMPLE supports Print functionality i.e users can print a string to output.
- SIMPLE supports comments at the top of the program, i.e before the 'begin' keyword.
- SIMPLE expects a semicolon at the end of all commands and declarations.
- SIMPLE allows data type declarations without initialization.
- Identifier should be an alphabetic word starting with a small letter.
- String in SIMPLE should be in “ ” double quoted.

Grammar:
P → Program
K → Block
D → Declarations
C → Commands
B → Boolean Expression
E → Expressions
UO → Unary Operators
BO → Boolean Operations
SC → Singleline Comment
EXP → Expression
TO → Ternary Operator
I → Identifier
N → Number
S → String
T → Set of Terminals

Grammar for SIMPLE:

```
P ::= begin K end | SC begin K end

K ::= D;C;| C;

D::= D; D | int I = N | string I = S | int I | string I | bool I | bool I = B

C::= C; C | if (B) then {C} else {C}
         | for (int I = E; B; UO ) {C}
         | for I in range(E,E) {C}
         | while (B) {C}
         | I = EXP
         | I = TO
         | D
         | print T

B ::= true | false | E == E | E != E | E < E | E <= E | E > E | E >= E | BO

E ::= E + E | E - E | E * E | E / E | (E) | I | N

UO ::= I++ | I-- | I = E

BO ::= B and B | B or B | not B | (B)
SC ::= /*S*/

EXP ::= E | BO

TO ::= B ? E : E
```

I ::= Identifier
N ::= number
S ::= “ string ”
T::= I | N | S | B

Sample Source codes:
1)

```
/*@author: Batman*/
begin
int x = 9;
bool yacc;
string s=”my string”;
if(true) then {print “true” ;}
else {x= 5>=3 ? x++ : yAcc = false}
end
```

2)

```
begin
string s=”Hello world”;
for (int x=0; x < 1;x++){
print s ;}
print “bye”;
end
```

Github Link:- https://github.com/saikumarchunchu/SER502-Spring2021-Team9