The Lucas Lexer

Group-3

1. JATIN TARACHANDANI
2. PRASHANTH SRIRAM S
3. S GOUTHAM SAI
4. SHANTANU PANDEY
5. ARAVINDA KUMAR REDDY T
6. ANIRUDH SRINIVASAN
7. T SRIVATSAN
8. JULAKUNTLA MADHURI

Design Overview of the Implementation

- From the language specification, lex rules were formulated [Keywords, Identifiers, Literals, Operators, Comments and Whitespaces]
- Antlr was setup as per the documentation.
- The formulated lex rules were implemented in Antlrv4.
- Makefile was made to generate the lexer using Antlr from the grammar file and to run lexer on test cases.

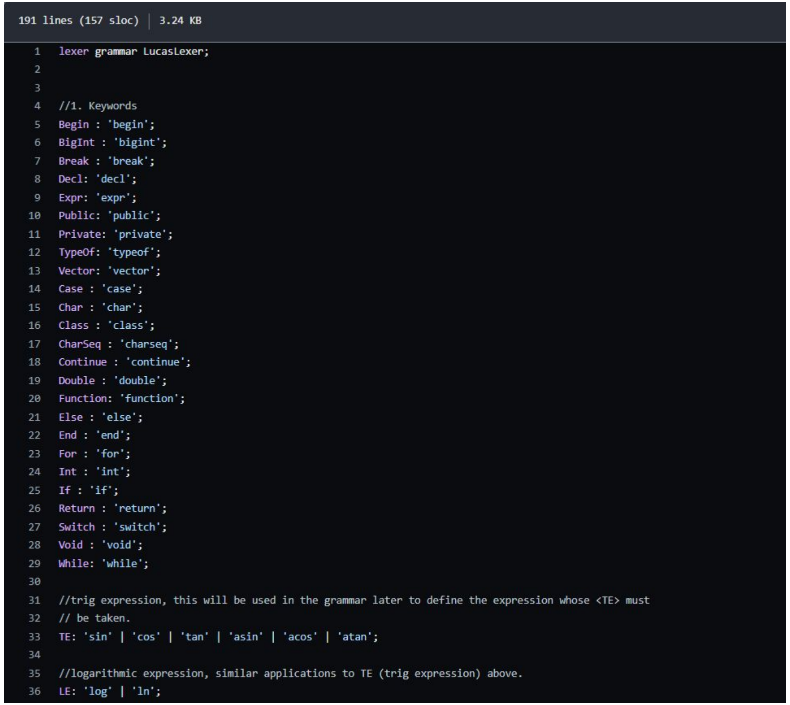
ANTLR V4

- ANTLR - ANother Tool for Language Recognition.
- Antlr can be used for both lexing and parsing.
- We need to provide a target language in which the Lexer should be generated.
  [For example: C++, C#, Java, Go, Swift, PHP]
- We have used Antlr to generate the Lexer in Java.

ANTLR

- ANTLR uses LL(k) parsing to analyse the grammar.
- Parsers, lexers, and tree-parsers are accepted grammar specifications.
- Commands: (‘LucasLexer.g4’ be the grammar file)
    - antlr4 LucasLexer.g4
    - javac LucasLexer*.java
    - grun LucasLexer tokens -tokens < test.txt

Grammar File



191 lines (157 sloc) | 3.24 KB

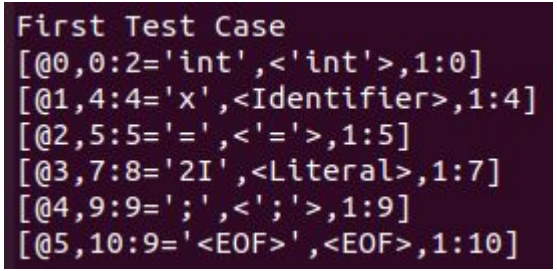
```
lexer grammar LucasLexer;


//1. Keywords
Begin : 'begin';
BigInt : 'bigint';
Break : 'break';
Decl: 'decl';
Expr: 'expr';
Public: 'public';
Private: 'private';
TypeOf: 'typeof';
Vector: 'vector';
Case : 'case';
Char : 'char';
Class : 'class';
CharSeq : 'charseq';
Continue : 'continue';
Double : 'double';
Function: 'function';
Else : 'else';
End : 'end';
For : 'for';
Int : 'int';
If : 'if';
Return : 'return';
Switch : 'switch';
Void : 'void';
While: 'while';

//trig expression, this will be used in the grammar later to define the expression whose <TE> must
// be taken.
TE: 'sin' | 'cos' | 'tan' | 'asin' | 'acos' | 'atan';

//logarithmic expression, similar applications to TE (trig expression) above.
LE: 'log' | 'ln';
```

A small snippet of our grammar file

Sample Input and Output

Input-1: `int x= 2I;`

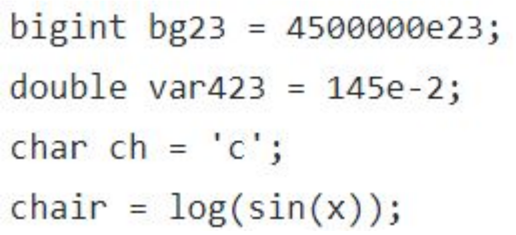
Output:



```
First Test Case
[@0,0:2='int',<'int'>,1:0]
[@1,4:4='x',<Identifier>,1:4]
[@2,5:5='=',<'='>,1:5]
[@3,7:8='2I',<Literal>,1:7]
[@4,9:9=';',<';'>,1:9]
[@5,10:9='<EOF>',<EOF>,1:10]
```

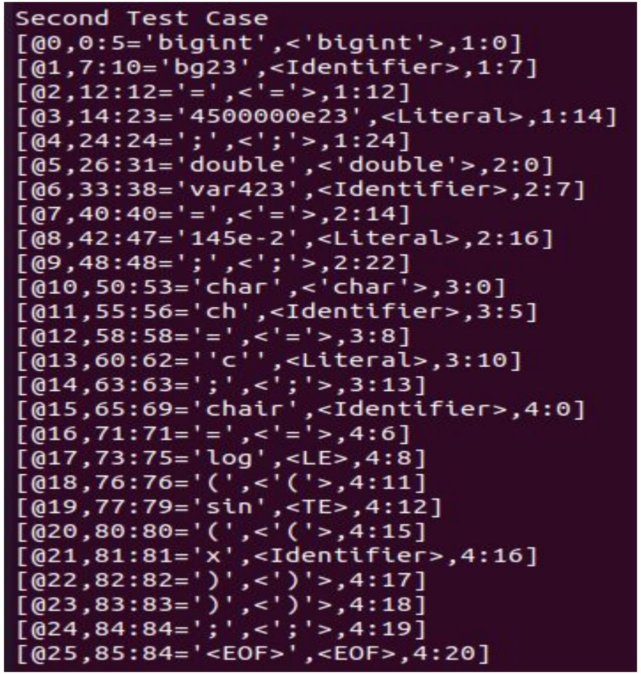
Sample Input and Output

Input-2:



```
bigint bg23 = 4500000e23;
double var423 = 145e-2;
char ch = 'c';
chair = log(sin(x));
```

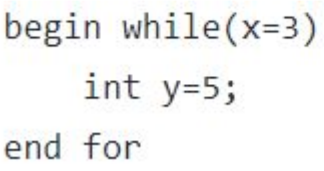
Output:



```
Second Test Case
[@0,0:5='bigint',<'bigint'>,1:0]
[@1,7:10='bg23',<Identifier>,1:7]
[@2,12:12='=',<'='>,1:12]
[@3,14:23='4500000e23',<Literal>,1:14]
[@4,24:24=';',<';'>,1:24]
[@5,26:31='double',<'double'>,2:0]
[@6,33:38='var423',<Identifier>,2:7]
[@7,40:40='=',<'='>,2:14]
[@8,42:47='145e-2',<Literal>,2:16]
[@9,48:48=';',<';'>,2:22]
[@10,50:53='char',<'char'>,3:0]
[@11,55:56='ch',<Identifier>,3:5]
[@12,58:58='=',<'='>,3:8]
[@13,60:62=''c'',<Literal>,3:10]
[@14,63:63=';',<';'>,3:13]
[@15,65:69='chair',<Identifier>,4:0]
[@16,71:71='=',<'='>,4:6]
[@17,73:75='log',<LE>,4:8]
[@18,76:76='(',<'('>,4:11]
[@19,77:79='sin',<TE>,4:12]
[@20,80:80='(',<'('>,4:15]
[@21,81:81='x',<Identifier>,4:16]
[@22,82:82=')',<')'>,4:17]
[@23,83:83=')',<')'>,4:18]
[@24,84:84=';',<';'>,4:19]
[@25,85:84='<EOF>',<EOF>,4:20]
```

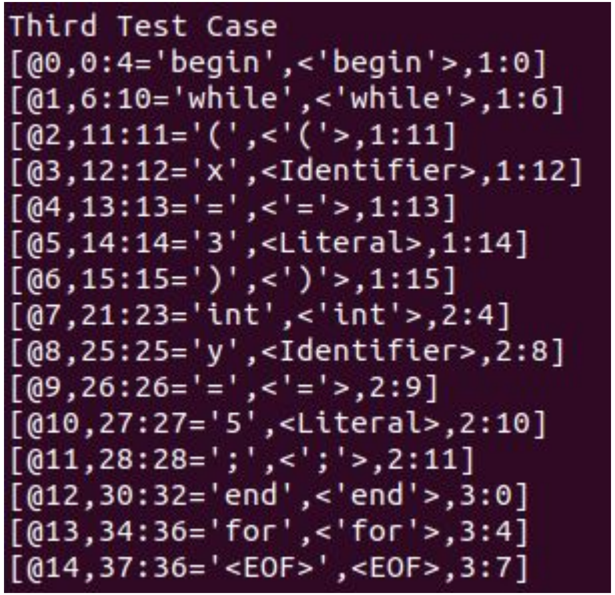
Sample Input and Output

Input-3:



```
begin while(x=3)
    int y=5;
end for
```

Output:

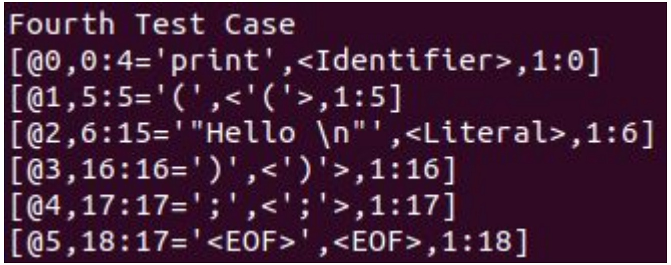


```
Third Test Case
[@0,0:4='begin',<'begin'>,1:0]
[@1,6:10='while',<'while'>,1:6]
[@2,11:11='(',<'('>,1:11]
[@3,12:12='x',<Identifier>,1:12]
[@4,13:13='=',<'='>,1:13]
[@5,14:14='3',<Literal>,1:14]
[@6,15:15=')',<')'>,1:15]
[@7,21:23='int',<'int'>,2:4]
[@8,25:25='y',<Identifier>,2:8]
[@9,26:26='=',<'='>,2:9]
[@10,27:27='5',<Literal>,2:10]
[@11,28:28=';',<';'>,2:11]
[@12,30:32='end',<'end'>,3:0]
[@13,34:36='for',<'for'>,3:4]
[@14,37:36='<EOF>',<EOF>,3:7]
```

Sample Input and Output

Input-4: `print("Hello \n");`

Output:



```
Fourth Test Case
[@0,0:4='print',<Identifier>,1:0]
[@1,5:5='(',<'('>,1:5]
[@2,6:15='"Hello \n"',<Literal>,1:6]
[@3,16:16=')',<')'>,1:16]
[@4,17:17=';',<';'>,1:17]
[@5,18:17='<EOF>',<EOF>,1:18]
```