Compiler Design Lab
Lab Mini Project

A C++ compiler for “NESTED WHILE and FOR” & “IF - ELSE -Ladder”

Context Free Grammar + Tokens

Name	SRN
R Siva Girish	PES1201700159
Parshva B Jain	PES1201701336
Mayank Agarwal	PES1201701349

G -> {V,T,P,S}

V = Variables **T = Terminals** **P = Productions**
S = Start Symbol

V = { Start , binop , unary , Relop , Logic_op , bitwise_op , Assignment , expr , stmt , dt }

T = { +, -, *, /, %, --, ++, <, <=, >=, >, ==, !=, &&, ||, ^, |, !, &, Name, Numeric Constants, String Constants, NULL, int , short , byte ,long , double , float , string , bool , char}

Tokens

Keyword -> <Keyword,pattern>
Pattern : if , else , for , while , cout , main ,
int, float, short, byte, long, double, string, bool, char

Identifier -> <Identifier, pattern, value, line_no>
Pattern: Name

RelationalOp -> <RelationalOp, pattern>
Pattern: >, <, >=, <=, ==, !=

Punctuation -> <Punctuation, Pattern>
Pattern: (,),',",{,}

AssignmentOp -> <AssignmentOp, Pattern>
Pattern: =

ArithmeticOp -> <ArithmeticOp, Pattern>
Pattern: +,-,*,/,%

BitwiseOp -> <BitwiseOp, Pattern>
Pattern: &,|

UnaryOp -> <UnaryOp, Pattern>
Pattern: ++, --

CFG Productions

Notes
+ Square brackets indicate optionality.
+ Curly brackets indicate *

Start -> **int main'('')' stmt**

binop -> +
| −
| *
| /
| %

unary -> --
| ++

Relop -> ==
| !=
| <=
| <
| >=
| >

Logic_op-> &&
| ||

bitwise_op -> &
| |
| ^

Assignment -> dt [{Name','}]Name ';'
| dt Name '=' expr ';'
| Name '=' expr ';'

```
expr        ->    '~' expr
                | '!' expr
                | unary expr
                | expr unary
                | expr Logic_op expr
                |  expr binop expr
                |  expr relop expr
                |  expr bitwise_op expr
                |  '(' expr ')'
                | Name
                | Numeric_Constants
                | String_Constants

stmt        ->          if '(' expr ')' stmt [else if '(' expr ')' stmt ][ else stmt ]
                |  while '(' expr ')' stmt
                | for '(' [ assg ] ';' [ expr ] ';' [ assg ] ')' stmt
                |  return [ expr ] ';'
                |  Assignment
                |  Name'(' [expr { ',' expr } ] ')' ';'
                |  '{' { stmt } '}'
                |  ';'
                | expr ';'
                | NULL
                | cout {{'<<' expr} [<< endl]}';'

dt      ->  int | short | byte | long | double | float | string | bool | char
```