Date: 14/08/2024

Experiment 2.6

AIM

To check syntax of for loop in C

ALGORITHM

1. Start
2. Create a lex file with following lexical rules:
    1. Include y.tab.h generated by the YACC program.
    2. If input is a sequence of digits:
        1. Return token INTEGER.
    3. If input is the keyword for:
        1. Return token FOR.
    4. If input is a type keyword (int, float, char):
        1. Return token TYPE.
    5. If input is a valid identifier (starts with a letter or _, followed by letters, digits, or _):
        1. Return token IDENTIFIER.
    6. If input matches relational operators (==, <=, >=, !=, <, >):
        1. Return token RELATIONAL_OPERATOR.
    7. If input matches assignment operators (=, +=, -=, *=, /=, etc.):
        1. Return token ASSIGN.
    8. If input matches arithmetic operators (+, -, *, /, %):
        1. Return token ARITHMETIC_OPERATOR.
    9. If input matches logical operators (||, &&):
        1. Return token LOGICAL_OPERATOR.
    10. If input matches bitwise operators (<<, >>, &, |, ^):
        1. Return token BITWISE_OPERATOR.
    11. If input matches parentheses, curly braces, semicolon, comma, or !:

    1. Return the corresponding token (LPAREN, RPAREN, LCURLY, RCURLY, SEMICOLON, COMMA, NOT).
12. For any other symbol, skip.

3. Create YACC to parse input as follows:
    1. Define tokens for all operators, keywords, and symbols as per the LEX file.
    2. Define grammar rules to recognize a valid C-style for loop:
        1. The input consists of zero or more for loops.
        2. Each for loop must match the pattern:

            for (initialization ; condition ; update) statement_block

            On successful recognition of a for loop, print Valid.
        3. Initialization can be a comma seperated series of variable declarations or assignments. Initialization can be NULL as well.
        4. Condition can be any logical or relational expression.
        5. Update can be assignments or arithmetic expressions.
        6. Statement block can be a single statement or a block enclosed in {}.
    3. In the user code section:
        1. Define error handling function.
        2. Define main function to call yyparse().
4. Use lex command to generate C program.
5. Use yacc to create y.tab,h and y.tab.c
6. Compile and run y.tab.c along with lex program using gcc.
7. Stop

RESULT

Successfully verified for loop syntax