Project Report: Parse Tree Generation

Compiler Design

CSE375

Fall’2020

SECTION:2

Submitted To:

DR. SHAMIM HASNAT RIPON

Professor

Department of CSE

East West University

Submitted by:

Maleha Israt Chowdhury

Submitted date:

25th Sept,2020.

Short description of Grammar:

There will be two part coming out from root. One is declaration and another one is function. Declaration will have the header file of the language and function will have the rest part. Structure used in grammar are mentioned below,

- Declaration
- Function
    - Main function
    - Sub function
- Conditional Statement
    - If-else(nested)
        - If expression
    - Switch Case
        - Switch expression
- Loop Statement
    - For loop
        - For expression
    - While loop
        - While expression
- Break Statement
- Return Statement
- Output Statement

- Variable statement
- Expression statement
    - Binary operation
    - Relation operation
    - Logical operation
    - Increment operation
- Method call statement

GRAMMAR STATEMENT(g4)

- grammar com ;
- root : declaration function ;

- declaration : ('*' declarationlist '"' declarationtype '"')+ ;
- declarationlist : 'include' ;
- declarationtype : term ('.' term)? ;

- function: main_func sub_func ;

- main_func : typeSpecifier 'main' '[' ']' block ;
- sub_func : typeSpecifier 'subfunc' '[' (typeSpecifier term (',')?)+ ']' block ;

- block : '(' statement ')' ;

statement :

```
    (var_stat
    |loop_stat
    |condition_stat
    |output_stat
    |break_stat
    |ifswitch
    |return_stat
    |method_call)+
    ;
```

- var_stat :

  typeSpecifier expr (','expr)* ':' | expr ':' ;

- loop_stat :

for_stat| while_stat ;

- for_stat :

    'for' '[' forexpr ']' block ;

- forexpr :

    var '=' term ',' expr ',' var incr_op ;

- while_stat :

    'while''['whilexpr']' block ;

- whilexpr :

    expr ',' expr ;

- condition_stat :

    'IF' '[' expr ']' block ('ELIF' '[' expr ']' block)* ('ELSE' block)?

    | 'SWITCH' '[' expr ']' block ;

- ifswitch :

    '(' expr ')'

    | (term ('('expr')')* '~' (output_stat|expr|break_stat))+ ;

- expr : expr binop expr | expr relop expr | expr logical_op expr

    | '(' expr ')'| expr term | term ;

- binop :

    'plus' | 'minus' | 'multiply' | 'divide' ;

- relop :

    'greater than equal' | 'less than' | 'greater than' |

    'less than equal' | '='|'!=' ;

- logical_op :

    '&&' | '||' ;

- incr_op:

    'increment'|'decrement' ;

- method_call:

    'subfunc' '[' (term (',')?)* ']' ':' ;

- break_stat:

    'break' ':' ;

- output_stat:

    'print' term ':' ;

- return_stat:

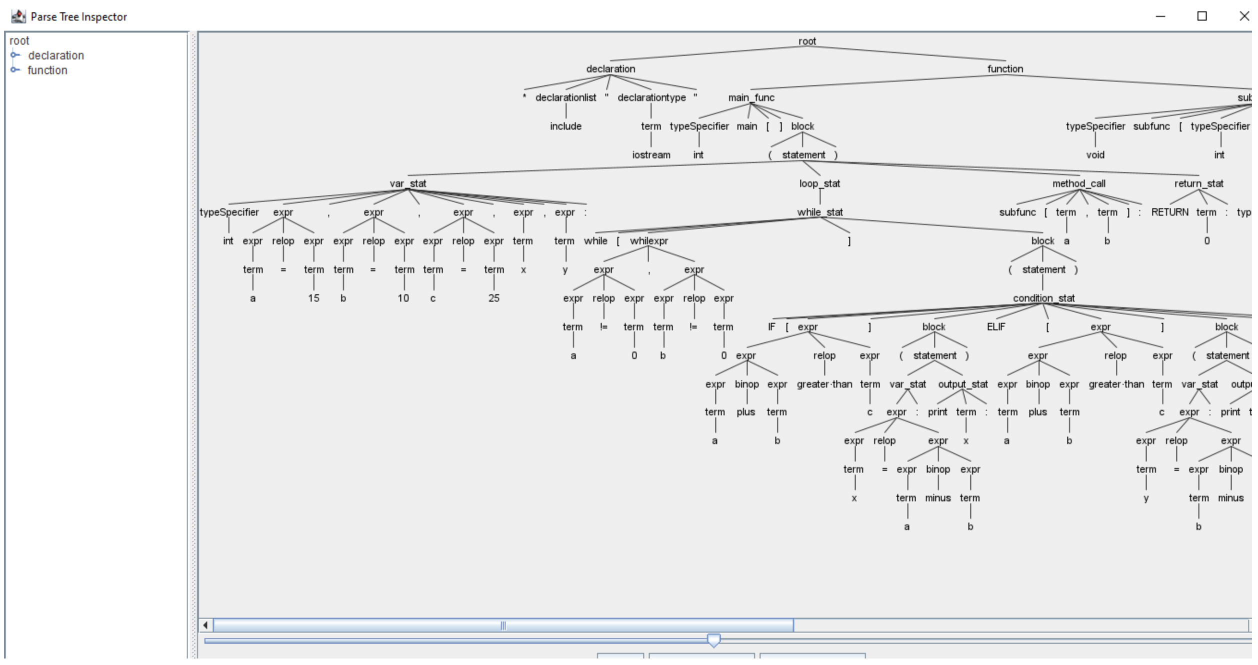
    'RETURN' term ':' | 'RETURN' ':' ;

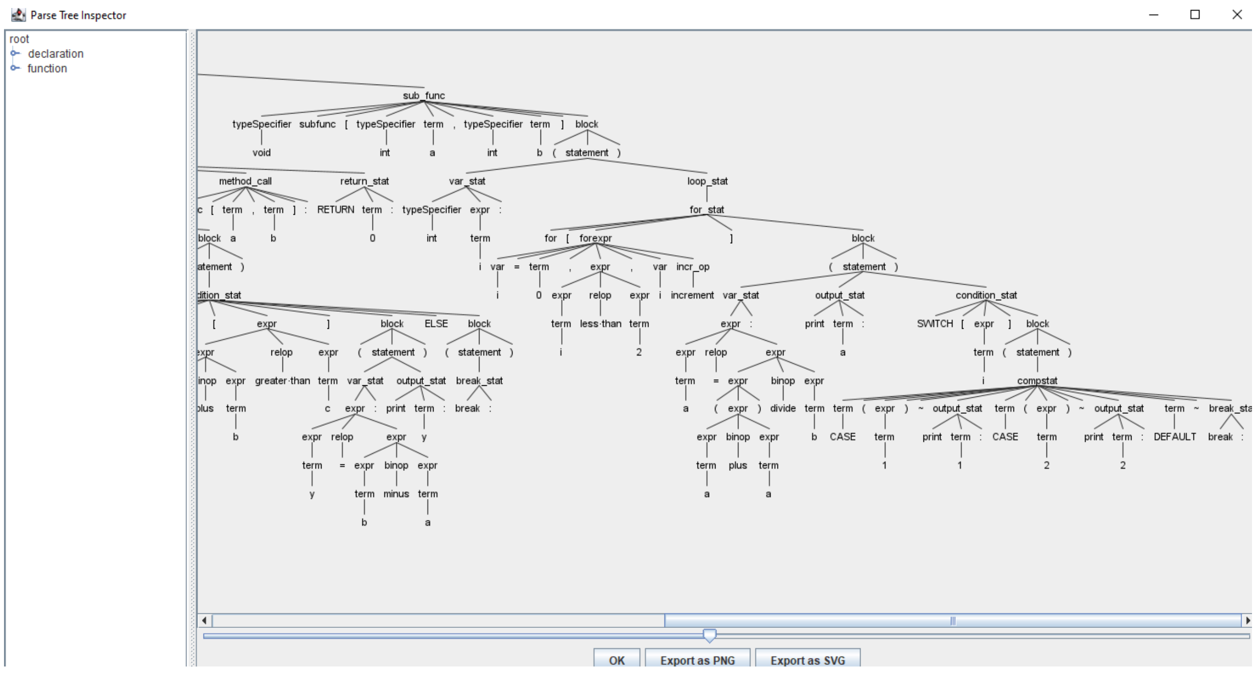
- typeSpecifier : 'int' | 'void' | 'float' ;
- var : ID ;
- term :ID |LIT ;
- ID : [a-zA-Z]+ ;
- LIT :[0-9]+ ;
- WS : [\t\r\n]+ -> skip ;

Input used for this grammar:

```
*include "iostream"

int main [ ](

        int a = 15,b=10,c=25,x,y:

                while[a!=0,b!=0](

                        IF[a plus b greater than c ]

                                (       x=a minus b :

                                        print x:

                                )

                        ELIF [a plus b greater than c ]

                                (       y=b minus a  :

                                        print y:

                                )

                        ELSE    (               break :

                                )

                )

                subfunc[a,b] :

                RETURN 0:

        )

void subfunc [int a,int b](

        int i:

        for[i = 0 ,i less than 2,i increment]

        (       a = (a plus a) divide b:

                print a:

                SWITCH[i] (

                CASE(1)~ print 1:

                CASE(2)~ print 2:

                DEFAULT~ break: )

                 ) )
```

Image of input :

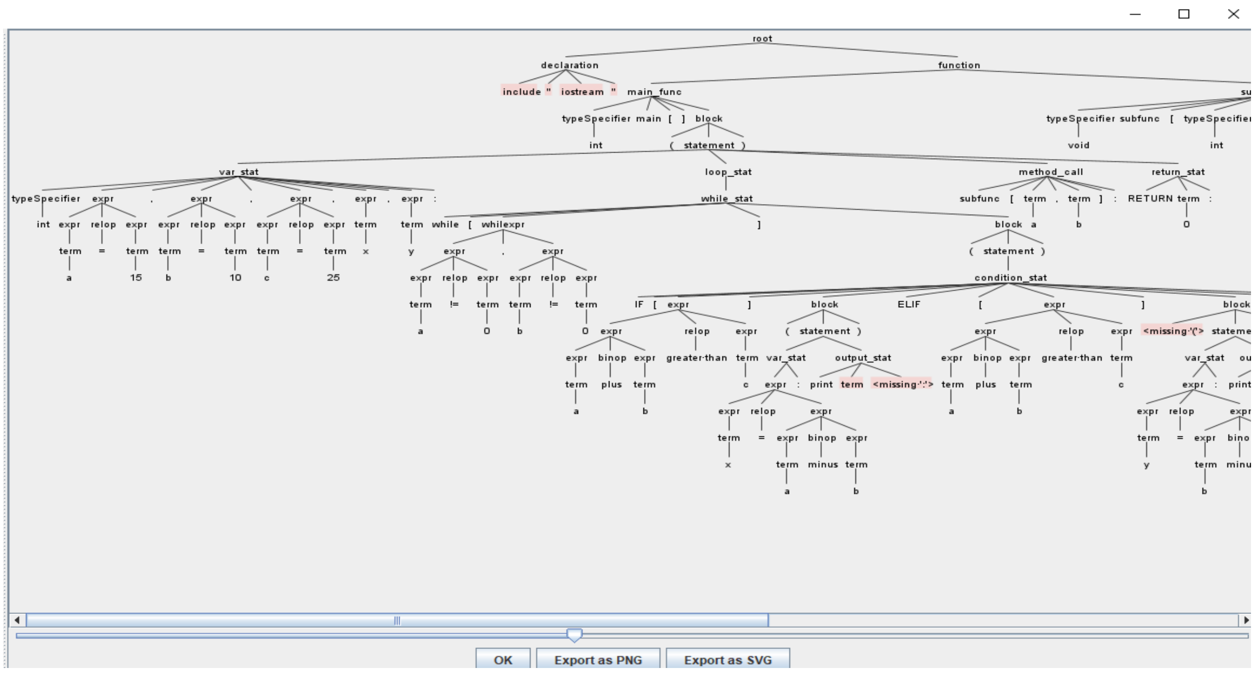


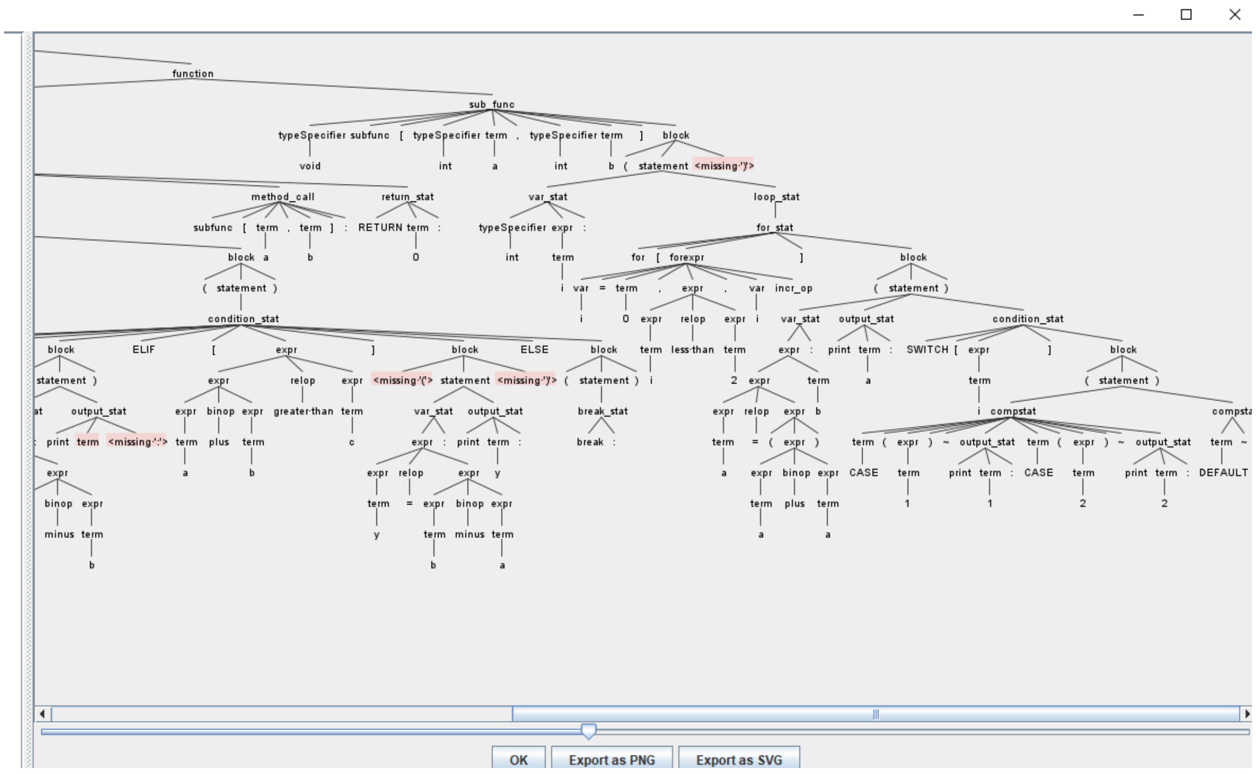


ERROR INPUT USED FOR THE GRAMMAR:

```
include "iostream"
int main [ ](
int a = 15,b=10,c=25,x,y:
                while[a!=0,b!=0](
                        IF[a plus b greater than c ]
                        (       x=a minus b :
                                Print
                        )
                        ELIF [a plus b greater than c ]
                                y=b minus a  :
                                print y:
                        ELSE   (
                                break :
                        )
                        )
                        subfunc[a,b] :
                        RETURN 0:
                        )
void subfunc [int a,int b](
        int i:
        for[i = 0 ,i less than 2,i ]
        (       a = (a plus a)  b:
                print a:
                SWITCH[i](
                CASE(1)~ print 1:
                CASE(2)~ print 2:
                DEFAULT~
                ) )
```

Image of Error file:


OK
Export as PNG
Export as SVG


OK
Export as PNG
Export as SVG