East West University

CSE375 Compiler Design Project

Section - 1

Submitted by-

Mizanur Rahman

ID: 2018-1-60-249

Submitted to-

Dr. Shamim H Ripon

Professor Department of Computer Science and Engineering
East West University

Description:

The grammar is designed to support the language type which includes:

1. A declaration function with include and define feature

2. A main function including types of expressions, conditional statements, iteration statements, reading and writing variables, break statement.

The syntax of the grammar, the input and its corresponding parse tree is given below.

Grammar:

```
grammar prog ;

root : declaration function ;

declaration : (declare_include | declare_define)+ ;

declare_include : '[' 'include' '(' declarationtype ')' ']';

declare_define : '[' 'define' LIT 'as' ID ']';

declarationtype : ID '.' ID ;

function : 'main' '[' ']' ':' block ;

block : '{' statement '}' ;

statement :(

 expressionstmt

| selectionstmt

| iterationstmt

| statement_return

| outputstmt
| inputstmt
| breakstmt
)+
;
expressionstmt : expr ',' typeSpecifier ';' ;
expr : expr binop expr | expr relop expr | expr logical_op expr | '(' expr ')' | term ;
statement_return : 'return' expr ';' | 'return' term ';' ;
binop : '+' | '-' | '*' | '/' ;
relop : '==' | '!=' | '<=' | '<' | '>' | '>=' | '=' ;
logical_op : 'and' | 'or' | 'not' ;
selectionstmt : 'if' '[' expr ']' block | 'if' '[' expr ']' block ('elif' '[' expr ']' block)*
'else' block ;
breakstmt : 'break' ;
iterationstmt : whilestmt | loopstmt ;
whilestmt : 'while' '[' expr ']' block ;
loopstmt : 'loop''['loopexpr']' block ;
loopexpr : var '='term 'to' var '=' term ',' 'increment' 'by' term ;
outputstmt : 'write' ':' expr ';' ;
inputstmt : 'read' ':' var ';' ;
var : ID ;
incr_op : '++'| '--' ;
term : ID | LIT ;
typeSpecifier : 'integer' | 'character' | 'float' ;
ID : [a-zA-Z]+ ;
LIT : [0-9]+ ;
```

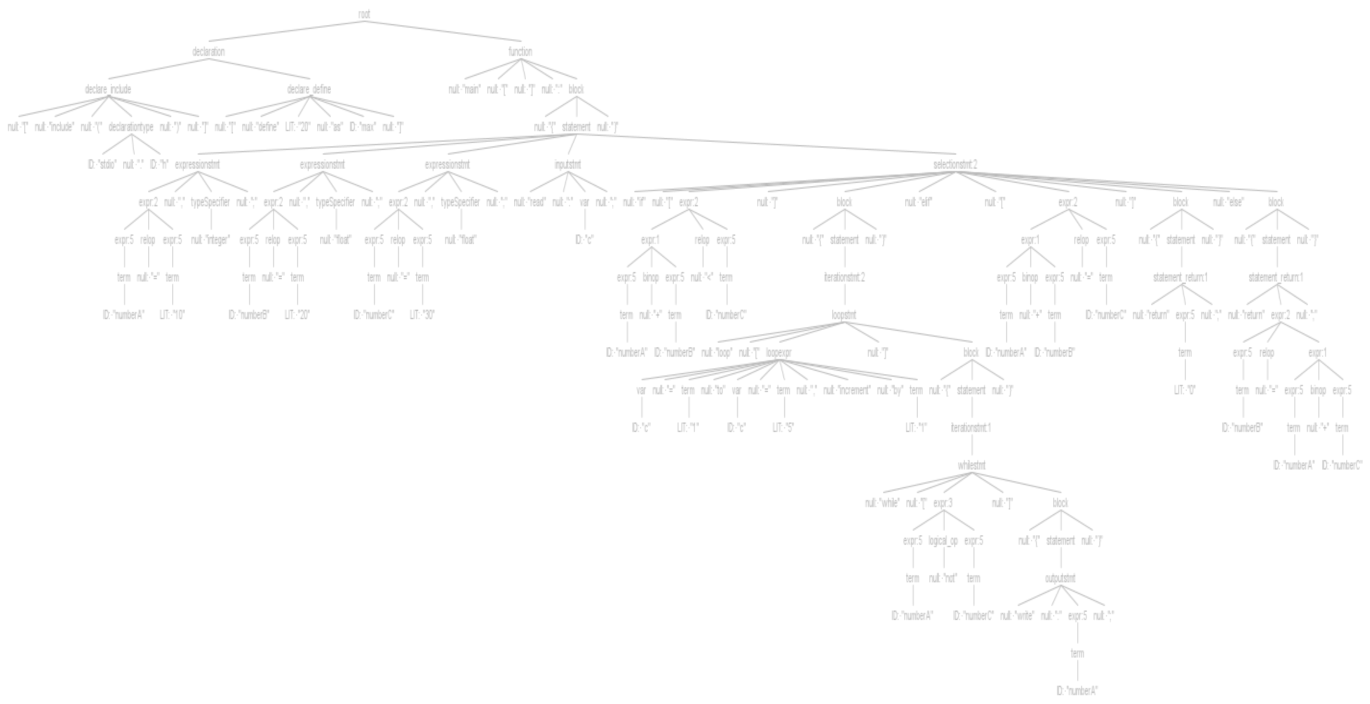
```
WS : [ \t\r\n]+ -> skip ;
```

Correct input;

```
[include (stdio.h)]
[define 20 as max]
main[]:
{
 numberA  = 10, integer;
 numberB  = 20, float;
 numberC=30, float;
 read: c;
if[numberA  + numberB  < numberC]
{
                    loop[c=1 to c = 5,increment by 1]
                              {
                                                  while[numberA  not numberC]
                                        {
                                                  write: numberA ;
                                        }
                              }
}
elif[numberA  + numberB  = numberC ]
          {
```

```
                    return 0;

          }

else

          {

                    return numberB =numberA  + numberC;

          }

}
```

Write parse tree:

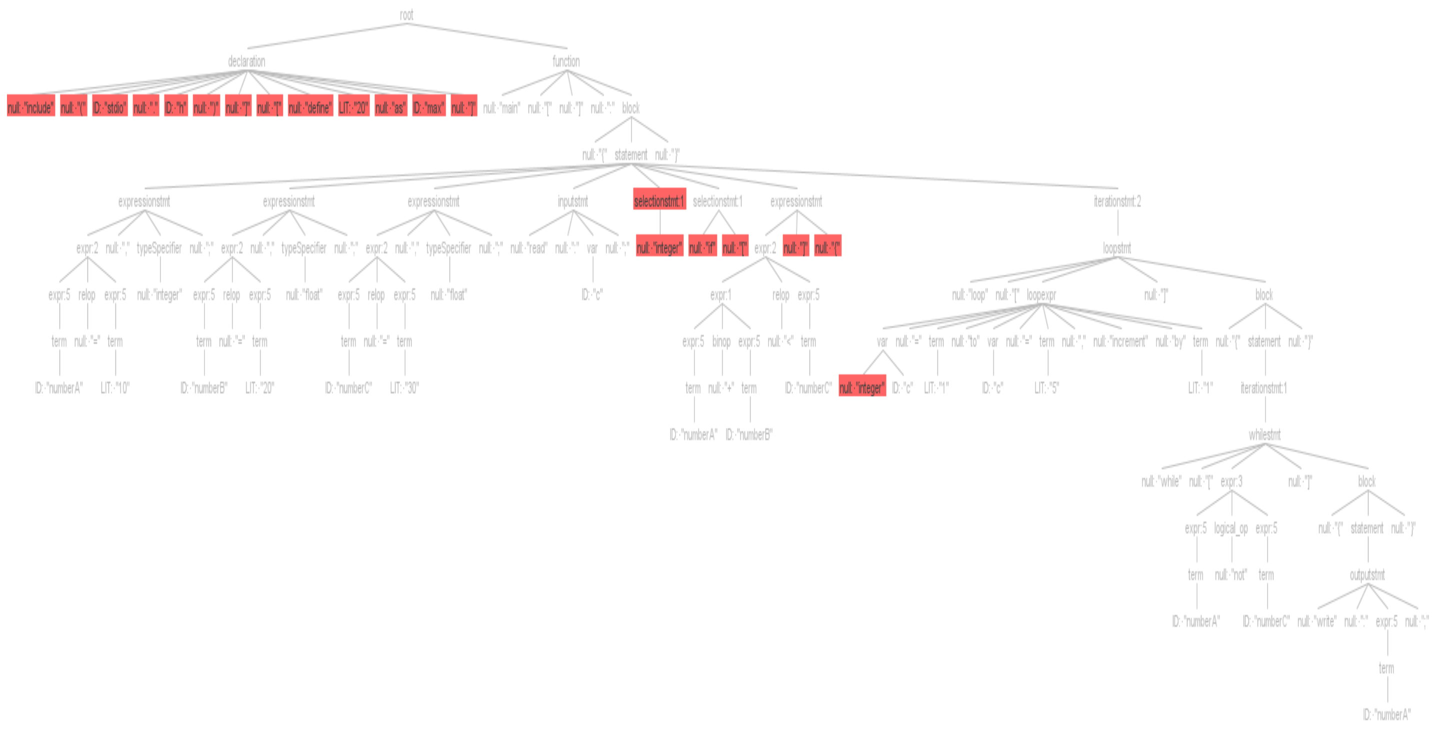


Wrong input:

```
#include (stdio.h)]
[define 20 as max]
main[]:
{
 numberA  = 10, integer;
 numberB  = 20, float;
 numberC=30, float;
 read: c;
if[numberA  + numberB  < numberC]
{
                    loop[integer c=1 to c = 5,increment by 1]
                              {
                                                  while[numberA  not numberC]
                                        {
                                                  write: numberA ;
                                        }
                              }
}
elif[numberA  + numberB  = numberC ]
           {
                     return 0;
```

```
        }
else
      {
              return numberB =numberA  + numberC;
      }
}
```

Wrong parse tree:


root
declaration
function
null: "include"
null: "("
ID: "stdio"
null: "."
ID: "h"
null: ")"
null: "]"
null: "["
null: "define"
LIT: "20"
null: "as"
ID: "max"
null: "]"
null: "main"
null: "["
null: "]"
null: ":"
block
null: "{"
statement
null: "}"
expressionstmt
expressionstmt
expressionstmt
inputstmt
selectionstmt:1
selectionstmt:1
expressionstmt
iterationstmt:2
expr:2
null: ","
typeSpecifier
null: ";"
expr:2
null: ","
typeSpecifier
null: ";"
expr:2
null: ","
typeSpecifier
null: ";"
null: "read"
null: ":"
var
null: ";"
null: "integer"
null: "if"
null: "["
expr:2
null: "]"
null: "{"
loopstmt
expr:5
relop
expr:5
null: "integer"
expr:5
relop
expr:5
null: "float"
expr:5
relop
expr:5
null: "float"
ID: "c"
expr:1
relop
expr:5
null: "loop"
null: "["
loopexpr
null: "]"
block
term
null: "="
term
term
null: "="
term
term
null: "="
term
expr:5
binop
expr:5
null: "<"
term
var
null: "="
term
null: "to"
var
null: "="
term
null: ","
null: "increment"
null: "by"
term
null: "{"
statement
null: "}"
ID: "numberA"
LIT: "10"
ID: "numberB"
LIT: "20"
ID: "numberC"
LIT: "30"
term
null: "+"
term
ID: "numberC"
null: "integer"
ID: "c"
LIT: "1"
ID: "c"
LIT: "5"
LIT: "1"
iterationstmt:1
ID: "numberA"
ID: "numberB"
whilestmt
null: "while"
null: "["
expr:3
null: "]"
block
expr:5
logical_op
expr:5
null: "{"
statement
null: "}"
term
null: "not"
term
outputstmt
ID: "numberA"
ID: "numberC"
null: "write"
null: ":"
expr:5
null: ";"
term
ID: "numberA"