C++ GRAMMAR : (for and if constructs)

TEAM MEMBERS :

ARPIT SINGH (01FB16ECS073)
ASHISH SANU (01FB16ECS075)
BILAL SHAKIL (01FB16ECS091)


```
start:
    INCLUDE start | function start |  class start | nam start | declaration start |
    ;

nam : USING NAMESPACE obj ';' ;

class :
                CLASS ID '{' classbdy '}'  classobj ';'
                 ;

classbdy :
                ACCESS ':' classbdy | declaration classbdy | function classbdy | ;

classobj : ID ',' classobj | ID | ;

function:
          type ID '(' arg_list ')'  compound_statement  |
          type ID '(' ')'  compound_statement |
          type ID '(' type_list ')' ';' |
          type ID ':'':' ID '(' arg_list ')'  compound_statement |
          type ID ':'':' ID '(' ')'  compound_statement
          ;

arg_list : arg ',' arg_list | arg ;

type_list : type ',' type_list | type ;

arg:
          type ID |
          ;

compound_statement:
           '{' statement_list '}' |
           '{' '}'
```

```
        ;

statement_list:
                statement |
                statement statement_list
                ;

statement:
                declaration |
                assignment |
                array |
                for |
                if_else|
                function_call  |
                RETURN expression ';'

                ;


declaration:
                type identifier_list ';' ;

identifier_list:
                ID ',' identifier_list | ID
                ;

assignment:
                ID '=' expression ';' |
                type ID '=' expression ';'
                ;


for:
                FOR '(' assignment  expression  ';' expression ')' compound_statement |

                FOR '(' assignment  expression  ';' expression ')' statement |

                FOR '(' ';' expression ';' ')' compound_statement |

                FOR '(' ';' ';' ')' compound_statement |

                FOR '(' assignment ';' ')' compound_statement |

                FOR '(' assignment expression ';' ')' compound_statement |

                FOR '(' ';' expression ';' expression ')' compound_statement |

                FOR '(' assignment ';'expression ')' compound_statement
                ;
```

```
if_else:
                IF '(' expression ')' compound_statement |
                if_else ELSE IF '('expression ')' compound_statement |
                if_else ELSE compound_statement
                ;


expression : expression AND rel_exp | expression OR rel_exp | NOT rel_exp | rel_exp ;

rel_exp : rel_exp relop add_expression | add_expression ;

add_expression : add_expression '+' mul_expression | add_expression '-' mul_expression |
                  mul_expression ;

mul_expression : mul_expression '*' cast_exp |  mul_expression '/' cast_exp | cast_exp ;

cast_exp : unary_exp | '(' type ')' cast_exp ;

unary_exp : exp | INCR exp | DECR exp | exp INCR | exp DECR | unary_op exp ;

unary_op : '-' | '+' | '&' | '!' ;

exp : base | exp '(' ')' | exp '(' identifier_list ')' ;

base : ID | NUM | FNUM | STRING | '(' expression')' ;

relop : LE | GE | GT | LT | EE  | NE ;

array:
                type ID '[' NUM ']' ';'   |
                type ID '[' NUM ']' '=' STRING ';'   |
                ID '[' NUM ']' '=' STRING ';' |
                type ID '[' NUM ']' '=' NUM ';'  |
                ID '[' NUM ']' '=' NUM ';'
                ;

function_call:
                ID '(' identifier_list ')' ';'
                ID '(' ')' ';'
                ;
obj :    STD;

type:
                INT | VOID | CHAR | FLOAT  | DOUBLE | BOOL
                ;
```