[BNF]

START -> PROGRAM

PROGRAM -> program identifier COMPOUND\_STMT

COMPOUND\_STMT -> begin STMTS end

STMTS -> STMT STMTS | STMT

STMT -> CONDITIONAL\_STMT | WHILE\_STMT | FOR\_STMT | SIMPLE\_STMT ;

CONDITIONAL\_STMT -> if EXPRESSION COMPOUND\_STMT ELSE\_IF\_STMT | if EXPRESSION COMPOUND\_STMT ELSE\_IF\_STMT else COMPOUND\_STMT

ELSE\_IF\_STMT -> else\_if EXPRESSION COMPOUND\_STMT ELSE\_IF\_STMT | ''

WHILE\_STMT -> while EXPRESSION COMPOUND\_STMT

FOR\_STMT -> for ( DECLARATION\_STMT ; EXPRESSION ; EXPRESSION ) COMPOUND\_STMT

SIMPLE\_STMT -> ASSIGNMENT\_STMT | PRINT\_STMT | DECLARATION\_STMT | BREAK\_STMT | DISPLAY\_STMT

ASSIGNMENT\_STMT -> identifier = EXPRESSION

PRINT\_STMT -> print\_line ( string\_literal ) | print\_line ( identifier )

DECLARATION\_STMT -> TYPE VARIABLE\_DECLARATION VARIABLE\_DECLARATIONS

VARIABLE\_DECLARATIONS -> , VARIABLE\_DECLARATION VARIABLE\_DECLARATIONS | ''

VARIABLE\_DECLARATION -> identifier | identifier = EXPRESSION

BREAK\_STMT -> break

DISPLAY\_STMT -> display ( string\_literal )

EXPRESSION -> SIMPLE\_EXPRESSION | SIMPLE\_EXPRESSION RELATIONAL\_OPERATOR SIMPLE\_EXPRESSION

SIMPLE\_EXPRESSION -> SIMPLE\_EXPRESSION ADDING\_OPERATOR TERM | TERM

TERM -> TERM MULTIPLYING\_OPERATOR FACTOR | FACTOR

FACTOR -> identifier | number\_literal | ( EXPRESSION ) | identifier ++

RELATIONAL\_OPERATOR -> < | > | = | ==

ADDING\_OPERATOR -> + | -

ADDING\_OPERATOR -> \* | /

MULTIPLYING\_OPERATOR -> \* | /

TYPE -> int | integer

[BNF]

START -> PROGRAM

PROGRAM -> program identifier COMPOUND\_STMT

COMPOUND\_STMT -> begin STMTS end

STMTS -> STMT STMTS

STMTS -> STMT

STMT -> CONDITIONAL\_STMT

STMT -> WHILE\_STMT

STMT -> FOR\_STMT

STMT -> SIMPLE\_STMT ;

CONDITIONAL\_STMT -> if EXPRESSION COMPOUND\_STMT ELSE\_IF\_STMT

CONDITIONAL\_STMT -> if EXPRESSION COMPOUND\_STMT ELSE\_IF\_STMT else COMPOUND\_STMT

ELSE\_IF\_STMT -> else\_if EXPRESSION COMPOUND\_STMT ELSE\_IF\_STMT

ELSE\_IF\_STMT -> ‘’

WHILE\_STMT -> while EXPRESSION COMPOUND\_STMT

FOR\_STMT -> for ( DECLARATION\_STMT ; EXPRESSION ; EXPRESSION ) COMPOUND\_STMT

SIMPLE\_STMT -> ASSIGNMENT\_STMT

SIMPLE\_STMT -> PRINT\_STMT

SIMPLE\_STMT -> DECLARATION\_STMT

SIMPLE\_STMT -> BREAK\_STMT

SIMPLE\_STMT -> DISPLAY\_STMT

ASSIGNMENT\_STMT -> identifier = EXPRESSION

PRINT\_STMT -> print\_line ( string\_literal )

PRINT\_STMT -> print\_line ( identifier )

DECLARATION\_STMT -> TYPE VARIABLE\_DECLARATION VARIABLE\_DECLARATIONS

VARIABLE\_DECLARATIONS -> , VARIABLE\_DECLARATION VARIABLE\_DECLARATIONS

VARIABLE\_DECLARATIONS -> ‘’

VARIABLE\_DECLARATION -> identifier

VARIABLE\_DECLARATION -> identifier = EXPRESSION

BREAK\_STMT -> break

DISPLAY\_STMT -> display ( string\_literal )

EXPRESSION -> SIMPLE\_EXPRESSION

EXPRESSION -> SIMPLE\_EXPRESSION RELATIONAL\_OPERATOR SIMPLE\_EXPRESSION

SIMPLE\_EXPRESSION -> SIMPLE\_EXPRESSION ADDING\_OPERATOR TERM

SIMPLE\_EXPRESSION -> TERM

TERM -> TERM MULTIPLYING\_OPERATOR FACTOR

TERM -> FACTOR

FACTOR -> identifier

FACTOR -> number\_literal

FACTOR -> ( EXPRESSION )

FACTOR -> identifier ++

RELATIONAL\_OPERATOR -> <

RELATIONAL\_OPERATOR -> >

RELATIONAL\_OPERATOR -> =

RELATIONAL\_OPERATOR -> ==

ADDING\_OPERATOR -> +

ADDING\_OPERATOR -> -

MULTIPLYING\_OPERATOR -> \*

MULTIPLYING\_OPERATOR -> /

TYPE -> int

TYPE -> integer