Context-free Grammar Expression

BNF and EBNF for Recursive Descent Parser

Compiler Theory - Programming Project 2 : Recursive Descent Parser
21400646 Lim Chae Eon

BNF Expression

Italic word : Non-Terminal, **blue colored word : Terminal**

```
# Script Definition part
Program          →    <script_start> stmt-sequence <script_end>
stmt-sequence    →    statement ; stmt-sequence | ε
statement        →    if-stmt | declaration-stmt | loop-stmt | assign-stmt
                      | function-stmt | switch-stmt | increment-stmt
                      | comment | break
simple-stmt      →    statement ; | { stmt-sequence }

# If-Conditional Statement Part
if-stmt          →    if function-parameter simple-stmt
                      | if function-parameter simple-stmt else simple-stmt

# Variable Declaration Statement Part
declaration-stmt →    var WS* id
                      *WS : white space
id               →    assign-stmt | id , id | IDENTIFIER

# Assignment Definition Statement Part
assign-stmt      →    IDENTIFIER assign-op exp
assign-op        →    = | -= | +=
```

```
# Operation Expression Part
exp          →    simple-exp logic-op simple-exp | simple-exp
logic-op     →    < | > | <= | >= | == | !=
simple-exp   →    simple-exp add-op term | term
add-op       →    + | −
term         →    term mul-op factor | factor
mul-op       →    * | /
factor       →    ( exp ) | NUMBER | IDENTIFIER

# Loop Statement Part
loop-stmt    →    for function-parameter simple-stmt
                | while function-parameter simple-stmt

# Function Statement Part
function-stmt  →    function-keyword function-parameter
function-keyword  →    window.prompt | window | parseFloat
                     | document.writeln | document.write | document

# Increment/Decrement Operation Part
increment-stmt  →    increment-op IDENTIFIER | IDENTIFIER increment-op
increment-op    →    ++ | −−

# Switch-Conditional Statement Part
switch-stmt     →    switch function-parameter { case-part }
case-part       →    case-block case-part default-block
case-block      →    case-condition : stmt-sequence | ε
```

42 *case-condition* → **case** *case-parameter*

43 *case-parameter* → **NUMBER** | **LITERAL**

44 *default-block* → **default :** *stmt-sequence* | ε

45

46 **# Function Parameter Handling Part**

47 *function-parameter* → **(** *exp* **)** | **(** *exp* **;** *exp* **;** *exp* **)** | **(LITERAL)**

48

49 **# Comment Part**

50 *comment* → **// anything EOL***

*EOL : End of Line

EBNF Expression

Italic word : Non-Terminal, **blue colored word : Terminal,** **bold-face word : EBNF expression**

```
1  # Script Definition part
2  Program          →    <script_start> stmt-sequence <script_end>
3  stmt-sequence    →    statement [ ; stmt-sequence ]
4  statement        →    if-stmt | declaration-stmt | loop-stmt | assign-stmt
                         | function-stmt | switch-stmt | increment-stmt
                         | comment | break
5  simple-stmt      →    statement | { stmt-sequence }
6
7  # If-Conditional Statement Part
8  if-stmt          →    if function-parameter simple-stmt [ else simple-stmt ]
9
10 # Variable Declaration Statement Part
11 declaration-stmt →    var WS id { , id }
12 id               →    assign-stmt | IDENTIFIER
13
14 # Assignment Definition Statement Part
15 assign-stmt      →    IDENTIFIER assign-op exp
16 assign-op        →    = | -= | +=
17
18 # Operation Expression Part
19 exp              →    simple-exp [ logic-op simple-exp ]
20 logic-op         →    < | > | <= | >= | == | !=
21 simple-exp       →    term { add-op term }
```

```
add-op          →   + | −

term            →   factor { mul-op factor }

mul-op          →   * | /

factor          →   ( exp ) | NUMBER | IDENTIFIER


# Loop Statement Part

loop-stmt       →   for function-parameter simple-stmt
                    | while function-parameter simple-stmt


# Function Statement Part

function-stmt   →   function-keyword function-parameter

function-keyword    →   window.prompt | window | parseFloat
                        | document.writeln | document.write | document


# Increment/Decrement Operation Part

increment-stmt      →   increment-op IDENTIFIER | IDENTIFIER increment-op

increment-op        →   ++ | −−


# Switch-Conditional Statement Part

switch-stmt         →   switch function-parameter { case-part }

case-part           →   { case case-parameter : stmt-sequence } default-block

case-parameter      →   NUMBER | LITERAL

default-block       →   [ default : stmt-sequence ]


# Function Parameter Handling Part

function-parameter  →   ( exp [ ; exp ; exp ] ) | ( LITERAL )
```

46

47 # Comment Part

48 *comment* → **// anything EOL***

*EOL : End of Line