Context-free Grammar Expression

BNF and BNF w/o Left Recursion for LL(1) Parser

Compiler Theory - Programming Project 3 : Recursive Descent Parser

21400646 Lim Chae Eon

BNF Expression

Italic word : Non-Terminal, **blue colored word : Terminal,** Yellow box : left recursion occured or left factored

# Script Definition part		
Program	→	**<script_start>** *stmt-sequence* **<script_end>**
stmt-sequence	→	*statement* **;** *stmt-sequence* | comment **nextLine** | ε
statement	→	*if-stmt* | *declaration-stmt* | *loop-stmt* | *assign-stmt* | *function-stmt* | *switch-stmt* | *increment-stmt* | **break**
simple-stmt	→	*stmt-sequence* | **{** *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** *id*
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 default-block }
case-part        →   case-block case-part | ε
case-block       →   case-condition : stmt-sequence
```

```
case-condition        →     case case-parameter
case-parameter        →     NUMBER | LITERAL
default-block         →     default : stmt-sequence |  ε

# Function Parameter Handling Part
function-parameter    →     ( exp ) | ( exp ; exp ; exp ) | ( LITERAL )

# Comment Part
comment        →     // anything EOL*
```

*EOL : End of Line

BNF Expression without Left-Recursion

Italic word : Non-Terminal, **blue colored word : Terminal,** Green box : Resolved left recursion or factored

---

# Script Definition part

Program → **<script_start>** *stmt-sequence* **<script_end>**

stmt-sequence → *statement* **;** *stmt-sequence* | *comment* **nextLine** | ε

statement → *if-stmt* | *declaration-stmt* | *loop-stmt* | *assign-stmt* | *function-stmt* | *switch-stmt* | *increment-stmt* | **break**

simple-stmt → *stmt-sequence* | **{** *stmt-sequence* **}**

# If-Conditional Statement Part

if-stmt → **if** *function-parameter simple-stmt else-part*

else-part → **else** *simple-stmt* | ε

# Variable Declaration Statement Part

declaration-stmt → **var** *id*

id → *assign-stmt* id_2 | **IDENTIFIER** id_2

id_2 → ε | **,** id id_2

# Assignment Definition Statement Part

assign-stmt → **IDENTIFIER** *assign-op exp*

assign-op → **=** | **−=** | **+=**

# Operation Expression Part

exp → *simple-exp logic-exp*

logic-exp → *logic-op simple-exp* | ε

logic-op → **<** | **>** | **<=** | **>=** | **==** | **!=**

simple-exp → term simple-exp$_2$

simple-exp$_2$ → add-op term simple-exp$_2$ | ε

add-op → **+** | −

term → *factor term*$_2$

term$_2$ → *mul-op factor term*$_2$ | ε

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 default-block* **}**

case-part → *case-block case-part* | case-block

```
case-block            →    case-condition : stmt-sequence | ε
case-condition        →    case case-parameter
case-parameter        →    NUMBER | LITERAL
default-block         →    default : stmt-sequence |  ε

# Function Parameter Handling Part
function-parameter    →    ( parameter-part )
paramter-part         →    exp condition-part | LITERAL
condition-part        →    ; exp condition-part | ε

# Comment Part
comment         →    // anything nextLine
```