First and Follow Sets for LL(1) Parser

Compiler Theory - Programming Project 3 : LL(1) Parser

21400646 Lim Chae Eon

Grammar Rule Choices from BNF

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

stmt-sequence	→	*statement* **;** *stmt-sequence*
stmt-sequence	→	*comment* **nextLine**
stmt-sequence	→	ε
statement	→	*if-stmt*
statement	→	*declaration-stmt*
statement	→	*loop-stmt*
statement	→	*assign-stmt*
statement	→	*function-stmt*
statement	→	*switch-stmt*
statement	→	*increment-stmt*
statement	→	**break**
simple-stmt	→	*stmt-sequence*
simple-stmt	→	**{** *stmt-sequence* **}**
if-stmt	→	**if** *function-parameter simple-stmt else-part*
else-part	→	**else** *simple-stmt*
else-part	→	ε
declaration-stmt	→	**var** *id*
id	→	*assign-stmt* id$_2$
id	→	**IDENTIFIER** id$_2$
id$_2$	→	**,** id id$_2$
id$_2$	→	ε
assign-stmt	→	**IDENTIFIER** *assign-op exp*
assign-op	→	**=**
assign-op	→	-=
assign-op	→	**+=**
exp	→	*simple-exp logic-exp*
logic-exp	→	logic-op *simple-exp*
logic-exp	→	ε
logic-op	→	**<**
logic-op	→	**>**
logic-op	→	**<=**
logic-op	→	**>=**
logic-op	→	**==**
logic-op	→	**!=**
simple-exp	→	term simple-exp$_2$
simple-exp$_2$	→	add-op term simple-exp$_2$
simple-exp$_2$	→	ε
add-op	→	**+**
add-op	→	−
term	→	*factor term*$_2$

term$_2$	→	*mul-op factor term*$_2$
term$_2$	→	ε
mul-op	→	*****
mul-op	→	**/**
factor	→	**(** *exp* **)**
factor	→	**NUMBER**
factor	→	**IDENTIFIER**
loop-stmt	→	**for** *function-parameter simple-stmt*
loop-stmt	→	**while** *function-parameter simple-stmt*
function-stmt	→	*function-keyword function-parameter*
function-keyword	→	**window.prompt**
function-keyword	→	**window**
function-keyword	→	**parseFloat**
function-keyword	→	**document.writeln**
function-keyword	→	**document.write**
function-keyword	→	**document**
increment-stmt	→	*increment-op* **IDENTIFIER**
increment-stmt	→	**IDENTIFIER** *increment-op*
increment-op	→	**++**
increment-op	→	−−
switch-stmt	→	**switch** *function-parameter* **{** *case-part default-block* **}**
case-part	→	*case-block case-part*
case-part	→	ε
case-block	→	*case-condition* **:** *stmt-sequence*
case-condition	→	**case** *case-parameter*
case-parameter	→	**NUMBER**
case-parameter	→	**LITERAL**
default-block	→	**default :** *stmt-sequence*
default-block	→	ε
function-parameter	→	**(** *parameter-part* **)**
paramter-part	→	*exp* condition-part
paramter-part	→	**LITERAL**
condition-part	→	**;** *exp condition-part*
condition-part	→	ε
comment	→	**// anything nextLine**

First Sets

First(Program) = {**<script_start>**}

First(stmt-sequence) = {ε**, break, if, var**, **for, while**, **IDENTIFIER**, **window.prompt, window, parseFloat, document.writeln, document.write, document**, **switch**, **//, ++, --**}

First(statement) = {**break, if, var**, **for, while**, **IDENTIFIER**, **window.prompt, window, parseFloat, document.writeln, document.write, document**, **switch, ++, --**}

First(*simple-stmt*) = {**{, break, if, var**, **for, while**, **IDENTIFIER**, **window.prompt, window, parseFloat, document.writeln, document.write, document**, **switch**, **//, ++, --**}

First(*if-stmt*) = {**if**}

First(*else-part*) = {**else**, ε}

First(*declaration-stmt*) = {**var**}

First(*id*) = {**IDENTIFIER**}

First(id_2) = {**,**, ε}

First(*assign-stmt*) = {**IDENTIFIER**}

First(*assign-op*) = {**=, −=, +=**}

First(*exp*) = {**(, NUMBER, IDENTIFIER**}

First(logic-exp) = {ε**, <, >, <=, >=, ==, !=**}

First(*logic-op*) = {**<, >, <=, >=, ==, !=**}

First(*simple-exp*) = {**(, NUMBER, IDENTIFIER**}

First($simple\text{-}exp_2$) = {ε**, +, −**}

First(*add-op*) = {**+, −**}

First(*term*) = {**(, NUMBER, IDENTIFIER**}

First($term_2$) = {ε**, *, /**}

First(*mul-op*) = {***, /**}

First(*factor*) = {**(, NUMBER, IDENTIFIER**}

First(*loop-stmt*) = {**for, while**}

First(*function-stmt*) = {**window.prompt, window, parseFloat, document.writeln, document.write, document**}

First(*function-keyword*) = {**window.prompt, window, parseFloat, document.writeln, document.write, document**}

First(*increment-stmt*) = {**++, −−, IDENTIFIER**}

First(*increment-op*) = {**++, −−**}

First(*switch-stmt*) = {**switch**}

First(*case-part*) = {ε**, case**}

First(*case-block*) = {**case**}

First(*case-condition*) = {**case**}

First(*case-parameter*) = {**NUMBER, LITERAL**}

First(*default-block*) = {**default,** ε}

First(*function-parameter*) = {**(**}

First(*paramter-part*) = {**(, NUMBER, IDENTIFIER** , **LITERAL**}

First(*condition-part*) = {**;**, ε}

First(*comment*) = {**//**}

Follow Sets

Follow(Program) = {$}

Follow(stmt-seq) = {<script_end>, case, }}

Follow(comment) = {nextLine}

Follow(statement) = {;}

Follow(if-stmt) = {Follow(statement)} = {;}

Follow(*declaration-stmt*) = {Follow(statement)} = {;}

Follow(*loop-stmt*) = {Follow(statement)} = {;}

Follow(*assign-stmt*) = {Follow(statement)} = {;, ,}

Follow(*function-stmt*) = {Follow(statement)} = {;}

Follow(*switch-stmt*) = {Follow(statement)} = {;}

Follow(*increment-stmt*) = {Follow(statement)} = {;}

Follow(function-parameter) = {{, break, if, var, for, while, IDENTIFIER, window.prompt, window, parseFloat, document.writeln, document.write, document, switch, IDENTIFIER, //, ;}

Follow(simple-stmt) = {else, ;, }}

Follow(else-part) = {;}

Follow(id) = {; }

Follow(id_2) = { ; }

Follow(assign-op) = {(, NUMBER, IDENTIFIER}

Follow(exp) = {;,), ,}

Follow(simple-exp) = {<, >, <=, >=, ==, !=,), ;, ,}

Follow(logic-exp) = {;,), ,}

Follow(logic-op) = {(, NUMBER, IDENTIFIER}

Follow(term) = {<, >, <=, >=, ==, !=,), ;, +, −, ,}

Follow($simpe\text{-}exp_2$) = {<, >, <=, >=, ==, !=,), ;, ,}

Follow(add-op) = {(, NUMBER, IDENTIFIER}

Follow(factor) = { *, /, ,}

Follow($term_2$) = {<, >, <=, >=, ==, !=,), ;, +, −, ,}

Follow(mul-op) = {(, NUMBER, IDENTIFIER}

Follow(factor) = { *, / }

Follow(function-keyword) = {(}

Follow(increment-op) = {IDENTIFIER, ;}

Follow(case-part) = {default}

Follow(default-block) = {}}

Follow(case-block) = {case}

Follow(case-condition) = {:}

Follow(case-parameter) = {:}

Follow(parameter-part) = {) }

Follow(condition-part) = {) }