Compiler Theory

The 3rd Program Assignment – Scanner Construction

21900628 – Sechang Jang

Index

Theoretical Foundations

BNF (Backus-Naur Form)

BNF is a notation technique used to express the grammar of a language in a formal way. It is widely used in the field of computer science to describe the syntax of programming languages, data structures, and document formats. BNF provides a set of derivation rules composed of a sequence of symbols, where each rule describes one of the constructs of the language. It uses ::= to denote definition, | to indicate alternatives, and terminal and non-terminal symbols to represent the basic elements and constructs of the language, respectively.

Left Recursion Elimination

Left recursion occurs in grammars when a non-terminal symbol in a production rule refers to itself as its first element. This can be problematic for certain types of parsers, like recursive descent parsers, because it can lead to infinite recursion. To handle this, left recursion needs to be eliminated from grammar rules. This is done by transforming recursive production rules into equivalent non-recursive ones, often by introducing new non-terminal symbols and rearranging the order of production rules to ensure that recursive calls are no longer made as the first action, thus allowing parsers to handle the grammar correctly without infinite loops. Left recursion elimination is a crucial step in preparing a grammar for many parsing techniques.

LL Parser

An LL parser is a type of top-down, predictive parser used for syntactic analysis in compilers, characterized by scanning input from left to right and producing a leftmost derivation. The most common form, LL(1), uses one lookahead token to make parsing decisions. It builds the parse tree from the start symbol down to the leaves, using a stack to maintain the parsing state. The stack initially contains the grammar's start symbol, and the input buffer holds the input string. The parser uses a parsing table to predict which production to apply based on the current lookahead token. During parsing, it matches and advances tokens, reporting a syntax error if there's a mismatch. The parsing process is successful if both the stack and input buffer are empty at the end. LL parsers require the grammar to be left-factored and free of left recursion, making them efficient for suitable grammars but limited to those that meet specific structural requirements.

BNF

PROGRAM -> program IDENTIFIER BLOCK
IDENTIFIER. -> STARTCHAR RESTCHAR
STARTCHAR -> LETTER | $
LETTER -> a | b | c | ... | z | A | B | C | ... | Z
DIGIT -> 0 | 1 | ... | 9
RESTCHAR -> POSSIBLECHAR RESTCHAR
POSSIBLECHAR -> LETTER | DIGIT | $ | . | _
BLOCK -> begin STATEMENT end
STATEMENT -> IF_BLOCK STATEMENT | ASSIGNMENT STATEMENT | PRINT STATEMENT | DISPLAY STATEMENT | FOR_STMT STATEMENT | WHILE_STMT STATEMENT | break; | IF_BLOCK | ASSIGNMENT | PRINT | FOR_STMT | WHILE_STMT
IF_BLOCK -> IF_STMT | IF_STMT ELSE_IF_STMT | IF_STMT ELSE_STMT | IF_STMT ELSE_IF_STMT ELSE_STMT
IF_STMT -> if (COMPARISON_STMT) BLOCK
ELSE_IF_STMT -> else_if (COMPARISON_STMT) BLOCK | else_if (COMPARISON_STMT) BLOCK ELSE_IF_STMT
ELSE_STMT -> else BLOCK
TYPE -> int | integer | /eps
COMPARISON_STMT -> NUMORID COMPARISON_OP NUMORID
ASSIGNMENT -> TYPE ASSIGN ;
ASSIGN -> IDENTIFIER EQUAL | ASSIGN , IDENTIFIER EQUAL
EQUAL -> = ASSIGNED
ASSIGNED -> NUMORID ARITHOP ASSIGNED | NUMORID
NUMORID -> NUMBER | IDENTIDIFER
PRINT -> print_line (“ STRING “) ;
STRING -> LETTER | LETTER STRING
NUMBER -> DIGIT | DIGIT NUMBER
COMPARISON_OP -> > | < | == | != | <= | >=
ARITHOP -> + | - | / | *
INCREMENTALOP -> ++ | /eps
DISPLAY -> display (“ string “) ;
FOR_STMT -> for (ASSIGNMENT ; COMPARISON_STMT ; IDENTIFIER INCREMENTALOP) BLOCK
WHILE_STMT -> while (COMPARSION_STMT) BLOCK

PROGRAM -> program IDENTIFIER BLOCK
IDENTIFIER. -> STARTCHAR RESTCHAR
STARTCHAR -> LETTER | $
LETTER -> a | b | c | ... | z | A | B | C | ... | Z
DIGIT -> 0 | 1 | ... | 9
RESTCHAR -> POSSIBLECHAR RESTCHAR'
RESTCHAR' -> POSSIBLECHAR RESTCHAR' | /eps
POSSIBLECHAR -> LETTER | DIGIT | $ | . | _
BLOCK -> begin STATEMENT end
STATEMENT -> IF_BLOCK STMT | ASSIGNMENT STMT | PRINT STMT | DISPLAY STMT | FOR_STMT STMT | WHILE_STMT STMT
STMT -> IF_BLOCK STMT | ASSIGNMENT STMT | PRINT STMT | DISPLAY STMT | FOR_STMT STMT | WHILE_STMT STMT | break; | /eps
IF_BLOCK -> IF_STMT | IF_STMT ELSE_IF_STMT | IF_STMT ELSE_STMT | IF_STMT ELSE_IF_STMT ELSE_STMT
IF_STMT -> if (COMPARISON_STMT) BLOCK
ELSE_IF_STMT -> else_if (COMPARISON_STMT) BLOCK NEW_ELSE_IF
NEW_ELSE_IF -> ELSE_IF_STMT | /eps
ELSE_STMT -> else BLOCK
TYPE -> int | integer | /eps
COMPARISON_STMT -> NUMORID COMPARISON_OP NUMORID
ASSIGNMENT -> TYPE ASSIGN ;
ASSIGN -> IDENTIFIER EQUAL ASSIGN'
ASSIGN' -> , IDENTIFIER EQUAL ASSIGN' | /eps
EQUAL -> = ASSIGNED ASSIGN'
ASSIGNED -> NUMERIC ASSIGNED'
ASSIGNED' -> ARITHOP NUMORID ASSIGNED' | /eps
NUMORID -> NUMBER | IDENTIDIFER
PRINT -> print_line (" STRING ") ;
STRING -> LETTER STRING'
STRING' -> LETTER STRING' | /eps
NUMBER -> DIGIT NUMBER'
NUMBER' -> DIGIT NUMBER' | /eps
COMPARISON_OP -> > | < | == | != | <= | >=
ARITHOP -> + | - | / | *
INCREMENTALOP -> ++ | /eps
DISPLAY -> display (" string ") ;
FOR_STMT -> for (ASSIGNMENT COMPARISON_STMT ; IDENTIFIER INCREMENTALOP) BLOCK
WHILE_STMT -> while (COMPARSION_STMT) BLOCK

First Set

First Set

First(DIGIT) = { 0, ⋯, 9 }

First(PROGRAM) = { Program } First(IDENTIFIER) = { a, ⋯, z, $ } First(LETTER) = { a, ⋯, z, A ⋯, Z }

First(STARTCHAR) = { a, ⋯, z, $ } First(RESTCHAR) = { a, ⋯, z, 0, ⋯, 9, $, ., _, A, ⋯, Z }

First(POSSIBLECHAR) = { a, ⋯, z, 0, ⋯, 9, $, ., _, A, ⋯, Z } First(BLOCK) = { begin }

First(RESTCHAR') = { a, ⋯, z, 0, ⋯, 9, $, ., _, A, ⋯, Z, ε } First(TYPE) = { integer, int, ε }

First(STATEMENT) = { if, a, ⋯, z, $, print_line, display, for, while, int, integer }

First(STMT) = { if, a, ⋯, z, $, print_line, display, for, while, break, int, integer, ε }

First(IF_BLOCK) = { if } First(IF_STMT) = { if } First(ELSE_IF_STMT) = { else_if }

First(NEW-ELSE-IF-STMT) = { else_if, ε } First(ELSE_STMT) = { else }

First(COMPARISON_STMT) = { a, ⋯, z, A ⋯, Z, 0 ⋯ 9, $ } First(ASSIGNMENT) = { a, ⋯, Z, $, int, integer }

First(ASSIGN) = { a, ⋯, z, $, A, ⋯, Z } First(ASSIGN') = { , }

First(ASSIGNED) = { 0, ⋯, 9, a ⋯ z, $, A, ⋯, Z } First(ASSIGNED') = { +, −, *, /, ε }

First(PRINT) = { print_line } First(NUMBER) = { 0, ⋯, 9 } First(NUMBER') = { 0, ⋯, 9, ε }

First(STRING) = { a, ⋯, z, A, ⋯, Z } First(STRING') = { a, ⋯, z, A, ⋯ Z, ε }

First(COMPARISONOP) = { <, >, ==, !=, <=, >= } First(ARITHOP) = { +, −, *, / }

First(INCREMENTALOP) = { ++, ε } First(DISPLAY) = { display } First(FOR_STMT) = for First(WHILE_STMT) = while

First(Equal) = { = }

Follow Set

Follow Set

Follow (PROGRAM) = { $ }

Follow (IDENTIFIER) = { begin, $, a .. z, ;, =, ,, >, <, ==, !=, >=, <=, +, -, *, / }

Follow (STARTCHAR) = { $, a .. z, 0 .. 9, ., _)

Follow (LETTER) = { $, a ... z, 0 .. 9, ., _, begin,), ;, =, ,, ", >, <, ==, !=, >=, <=, +, -, *, /, ++ }

Follow (DIGIT) = { $, a ... z, 0 .. 9, ., _, begin,), ;, =, ,, >, <, ==, !=, >=, <=, +, -, *, /, ++ }

Follow (RESTCHAR) = { $, a .. z, begin,), ;, =, ,, >, <, ==, !=, >=, <= , +, -, *, /, ++ }

Follow (POSSIBLECHAR) = { $, a ... z , 0 ... 9, ., _ , begin ,) , ;, = , , <, >, ==, !=, >=, <=, +, -, *, /, ++ }

Follow (RESTCHAR') = { $, a .. Z, begin,), ;, =, ,, }, <, ==, !=, >=, <=, +, -, *, /, tt }

Follow (BLOCK) = { $, a..Z, end, display, for, while, break;, if, else_if, else, int, integer, print_line, $ }

Follow (STATEMENT) = { end } Follow(STMT) = { end }

Follow (IF_BLOCK) = { $, a ... Z, end, display, for, while, break;, if, int, integer, print_line }

Follow (IF_STMT) = { $, a ... Z, end, display, for, while, break;, if, int, integer, print_line, else_if, else }

Follow (ELSE_IF_STMT) = { $, a ... Z, end, display, for, while, break;, if, int, integer, print_line, else }

Follow (NEW_ELSE_IF) = { $, a ... Z, end, display, for, while, break;, if, int, integer, print_line, else }

Follow (ELSE_STMT) = { $, a ... Z, end, display, for, while, break;, if, int, integer, print_line }

Follow (COMPARISON_STMT) = {), ; } Follow (TYPE) = { $, a .. Z }

Follow (ASSIGNMENT) = { $, a ... Z, end, display, for, while, break;, int, integer, ;, print_line }

Follow (ASSIGN) = { ; } Follow (ASSIGN') = { $, a ... Z, }

Follow (ASSIGNED) = { ;, , } Follow (ASSIGNED') = { ;, , }

Follow (PRINT) = { $, a ... Z, end, display, for, while, break;, int, print_line }

Follow (NUMORID) = { $, a ... Z,), ;, , , <, >, <=, >=, ==, !=, +, -, *, / }

Follow (NUMBER) = { $, a ... Z,), ;, , , <, >, <=, >=, ==, !=, +, -, *, / }

Follow (NUMBER') = { $, a ... Z,), ;, , , <, >, <=, >=, ==, !=, +, -, *, / }

Follow (STRING) = { " } Follow (STRING') = { " } Follow (COMPARISON_OP) = { $, a .. Z, 0 .. 9 }

Follow (ARITH OP) = { $, a .. Z, 0 .. 9 }

Follow (INCREMENTAL OP) = {) }

Follow (DISPLAY) = { $, a..Z, end, display, for, while, break;, if, int, integer, print_line }

Follow (FOR_STMT) = { $, a..Z, end, display, for, while, break;, if, int, integer, print_line }

Follow (WHILE_STMT) = { $, a..Z, end, display, for, while, break;, if, int, integer, print_line }

Follow (EQVAL) = { $, a .. Z, ; }

Parsing Table

	program	Identifer	Number	String	begin	end	break	if	(	)	else_if	else	int	integer	;	=	,	print_line	"	Compariso	Arithemati	++	break;	Display	for	while
PROGRAM	PROGRAM -> program																									
IDENTIFIER		IDENTIFIER																								
BLOCK					BLOCK -> begin																					
NUMBER			NUMBER ->																							
STRING				STRING ->																						
STATEMENT		STATEMENT ->						STATEMENT ->					STATEMENT ->	STATEMENT ->				STATEMENT -> PRINT						STATEMENT ->	STATEMENT ->	STATEMENT ->
STMT		STMT -> ASSIGNME				STMT -> ε		STMT -> BLOCK					STMT -> ASSIGNME	STMT -> ASSIGNME				STMT -> PRINT					STMT -> break;	STMT -> DISPLAY	STMT -> FOR_STMT	STMT -> WHILE_STM
IF_BLOCK								IF STMT -> IF_STMT ELSE_IF_ST																		
IF_STMT								if -> if (COMPARISON_STMT)																		
ELSE_IF_STMT		ELSE_IF_STMT -> ε				ELSE_IF_STMT -> ε	ELSE_IF_STMT -> ε				ELSE_IF_STMT ->	ELSE_IF_STMT -> ε	ELSE_IF_STMT -> ε	ELSE_IF_STMT -> ε				ELSE_IF_STMT -> ε						ELSE_IF_STMT -> ε	ELSE IF_STMT ->	ELSE IF_STMT ->
NEW_ELSE_IF		NEW_NEW_ELSE_IF_ST				NEW_NEW_ELSE_IF_ST	NEW_NEW_ELSE_IF_ST				NEW_ELSE_IF_STMT ->	NEW_NEW_ELSE_IF_ST	NEW_NEW_ELSE_IF_ST	NEW_NEW_ELSE_IF_ST				NEW_NEW_ELSE_IF_ST						NEW_NEW_ELSE_IF_ST	NEW_NEW_ELSE_IF_ST	NEW_NEW_ELSE_IF_ST
ELSE_STMT		ELSE_STMT				ELSE_STMT	ELSE_STMT				ELSE_STMT	ELSE_STMT	ELSE_STMT	ELSE_STMT				ELSE_STMT						ELSE_STMT	ELSE_STMT	ELSE_STMT
COMPARISON_STMT		COMPARISON_STMT -	COMPARISON_STMT -																							
ASSGINMENT		ASSIGNMENT -> TYPE											ASSIGNMENT -> TYPE	ASSIGNMENT -> TYPE												
ASSIGN		ASSIGN -> IDENTIFIER																								
EQUAL															EQUAL -> ε	EQUAL ->	EQUAL -> ε									
ASSIGN'															ASSIGN' -> ε		ASSIGN' -> , IDENTIFIER									
ASSIGNED		ASSIGNED ->	ASSIGNED ->																							
ASSIGNED'															ASSIGNED' -> ε		ASSIGNED' -> ε				ASSIGNED' ->				
PRINT																		PRINT -> print_line (								
NUMBEROR		NUMBEROR	NUMBEROR																NUMBEROR							
DISPLAY																								DISPLAY -> display (		
FOR_STMT																									FOR -> for (ASSIGNMENT	
WHILE_STMT																										WHILE -> while (COMPARIS
TYPE		Type -> ε											TYPE -> int	TYPE ->												
COMPOP																				COMPOP -						
INCREMENT																						INCREMENT				

Excel File is included for detail.

Output

Test 1

```
PS D:\Handong\4-1\Compiler Theory\HW3> java -cp bin LLparser .\testcase\test1.txt
[GENERATE-stack] [$, BLOCK, IDENTIFIER, program]
[MATCHED] program - program
[GENERATE-stack] [$, BLOCK, IDENTIFIER]
[MATCHED] IDENTIFIER - Test1
[GENERATE-stack] [$, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, TYPE]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, int]
[MATCHED] int - int
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 2206
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED] , - ,
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 0
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[GENERATE-stack] [$, end, STMT, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, end, STMT, PRINT]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), NUMORID, (, print_line]
[MATCHED] print_line - print_line
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), NUMORID, (]
[MATCHED] ( - (
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), NUMORID]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), STRING]
[MATCHED] STRING - "Good evening."
[MATCHED] ) - )
[GENERATE-stack] [$, end, STMT, end, STMT, ;]
[MATCHED] ; - ;
[GENERATE-stack] [$, end, STMT, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$]
Parsing OK
```

```
PS D:\Handong\4-1\Compiler Theory\HW3> java -cp bin LLparser .\testcase\test1error.txt
[GENERATE-stack] [$, BLOCK, IDENTIFIER, program]
[MATCHED] program - program
[GENERATE-stack] [$, BLOCK, IDENTIFIER]
[MATCHED] IDENTIFIER - Test1error
[GENERATE-stack] [$, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, TYPE]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, int]
[MATCHED] int - int
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 2206
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED] , - ,
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 0
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[GENERATE-stack] [$, end, STMT, ELSE_STMT, NEW_ELSE_IF]
[GENERATE-stack] [$, end, STMT, BLOCK, else]
[MATCHED] else - else
[GENERATE-stack] [$, end, STMT, BLOCK]
[GENERATE-stack] [$, end, STMT, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STMT, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, end, STMT, PRINT]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), NUMORID, (, print_line]
[MATCHED] print_line - print_line
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), NUMORID, (]
[MATCHED] ( - (
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), NUMORID]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ), STRING]
[MATCHED] STRING - "Good evening."
[MATCHED] ) - )
[GENERATE-stack] [$, end, STMT, end, STMT, ;]
[MATCHED] ; - ;
[GENERATE-stack] [$, end, STMT, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$, end, STMT]
Keyword end not matched
Parsing failed
```

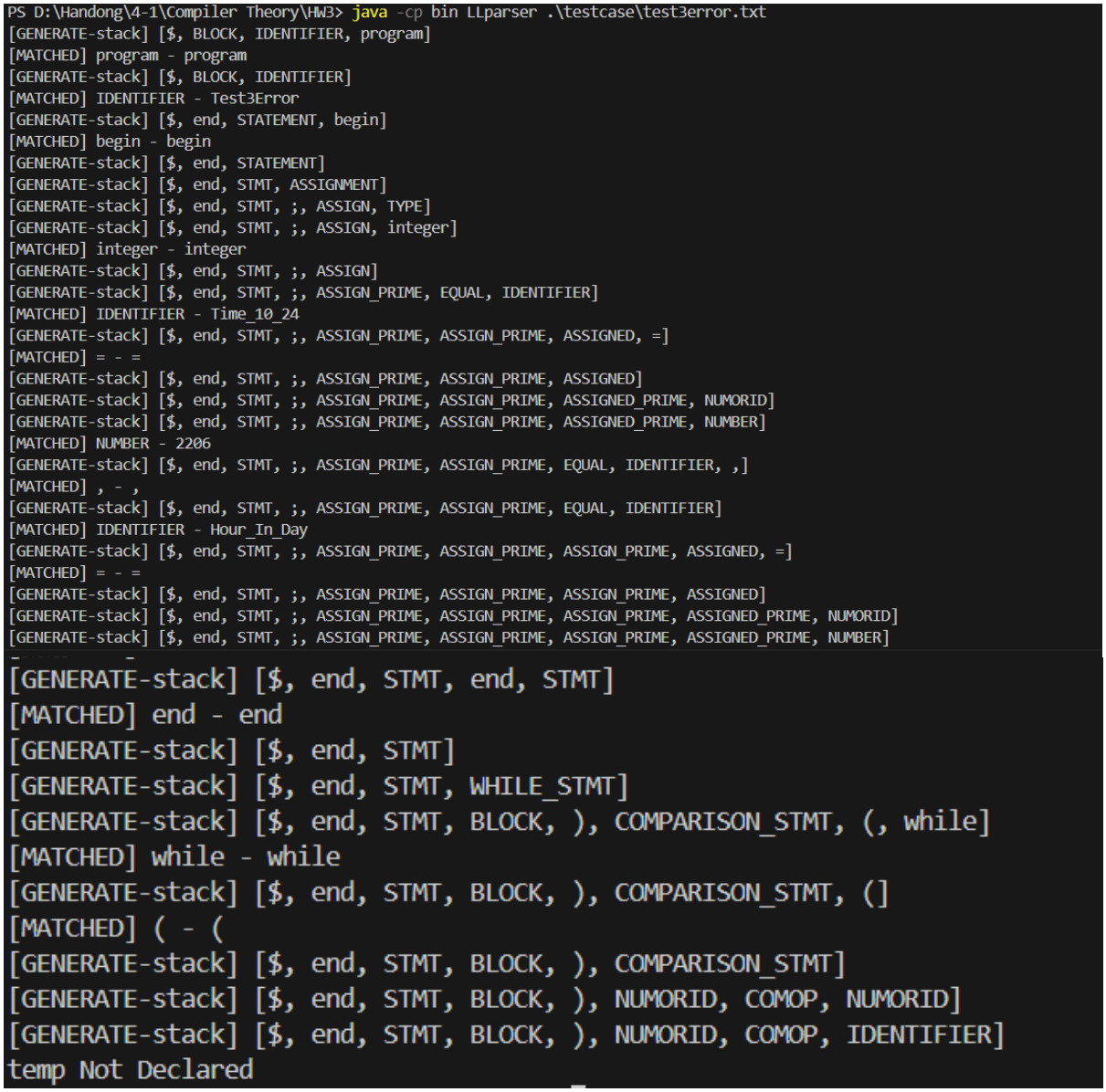
Test 2

```
PS D:\Handong\4-1\Compiler Theory\HW3> java -cp bin LLparser .\testcase\test2.txt
[GENERATE-stack] [$, BLOCK, IDENTIFIER, program]
[MATCHED] program - program
[GENERATE-stack] [$, BLOCK, IDENTIFIER]
[MATCHED] IDENTIFIER - Test2
[GENERATE-stack] [$, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, TYPE]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, int]
[MATCHED] int - int
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 0
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, IDENTIFIER]
[MATCHED] IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID, ARITH_OP]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID, +]
[MATCHED] + - +
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 10
[MATCHED] ; - ;
[GENERATE-stack] [$, end, STMT, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$]
Parsing OK
```

```
PS D:\Handong\4-1\Compiler Theory\HW3> java -cp bin LLparser .\testcase\test2error.txt
[GENERATE-stack] [$, BLOCK, IDENTIFIER, program]
[MATCHED] program - program
[GENERATE-stack] [$, BLOCK, IDENTIFIER]
[MATCHED] IDENTIFIER - Test2error
[GENERATE-stack] [$, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, TYPE]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, int]
[MATCHED] int - int
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 0
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED] , - ,
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 1
[MATCHED] ; - ;
[GENERATE-stack] [$, end, STMT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
Keyword Spelling Error
```

Test3

```
PS D:\Handong\4-1\Compiler Theory\HW3> java -cp bin LLparser .\testcase\test3.txt
[GENERATE-stack] [$, BLOCK, IDENTIFIER, program]
[MATCHED] program - program
[GENERATE-stack] [$, BLOCK, IDENTIFIER]
[MATCHED] IDENTIFIER - Test3
[GENERATE-stack] [$, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, TYPE]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, integer]
[MATCHED] integer - integer
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Time_10_24
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 2206
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED] , - ,
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Hour_In_Day
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 0
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED] , - ,
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] ; - ;
[GENERATE-stack] [$, end, STMT, end, STMT, ELSE_STMT, ELSE_IF_STMT, end]
[MATCHED] end - end
[GENERATE-stack] [$, end, STMT, end, STMT, ELSE_STMT, ELSE_IF_STMT]
[MATCHED] end - end
[GENERATE-stack] [$, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$]
Parsing OK
```


PS D:\Handong\4-1\Compiler Theory\HW3> java -cp bin LLparser .\testcase\test3error.txt
[GENERATE-stack] [$, BLOCK, IDENTIFIER, program]
[MATCHED] program - program
[GENERATE-stack] [$, BLOCK, IDENTIFIER]
[MATCHED] IDENTIFIER - Test3Error
[GENERATE-stack] [$, end, STATEMENT, begin]
[MATCHED] begin - begin
[GENERATE-stack] [$, end, STATEMENT]
[GENERATE-stack] [$, end, STMT, ASSIGNMENT]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, TYPE]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN, integer]
[MATCHED] integer - integer
[GENERATE-stack] [$, end, STMT, ;, ASSIGN]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Time_10_24
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[MATCHED] NUMBER - 2206
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER, ,]
[MATCHED] , - ,
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, EQUAL, IDENTIFIER]
[MATCHED] IDENTIFIER - Hour_In_Day
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED, =]
[MATCHED] = - =
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMORID]
[GENERATE-stack] [$, end, STMT, ;, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGN_PRIME, ASSIGNED_PRIME, NUMBER]
[GENERATE-stack] [$, end, STMT, end, STMT]
[MATCHED] end - end
[GENERATE-stack] [$, end, STMT]
[GENERATE-stack] [$, end, STMT, WHILE_STMT]
[GENERATE-stack] [$, end, STMT, BLOCK,), COMPARISON_STMT, (, while]
[MATCHED] while - while
[GENERATE-stack] [$, end, STMT, BLOCK,), COMPARISON_STMT, (]
[MATCHED] (- (
[GENERATE-stack] [$, end, STMT, BLOCK,), COMPARISON_STMT]
[GENERATE-stack] [$, end, STMT, BLOCK,), NUMORID, COMOP, NUMORID]
[GENERATE-stack] [$, end, STMT, BLOCK,), NUMORID, COMOP, IDENTIFIER]
temp Not Declared