|  |
| --- |
| Team 10 |
| **Compiler Term-Project #2** |
| The implementation of a bottom-up syntax analyzer |

|  |  |
| --- | --- |
| Date | June 20, 2020 |
| Instructor | Hyosu Kim |
| Team | #10 |
| Members | Heesang Ro (20145001)  Junhyuck Woo (20145337) |



# INDEX

[INDEX 1](#_Toc44101338)

[SPECIFICATIONS 2](#_Toc44101339)

[Modified part in specifications 2](#_Toc44101340)

[FIRST SET 3](#_Toc44101341)

[FOLLOW SET 4](#_Toc44101342)

[NFA (Non-deterministic Finite Automata) 5](#_Toc44101343)

[1. Overall Graph 5](#_Toc44101344)

[DFA (Deterministic Finite Automata) 6](#_Toc44101345)

[1. NFA with numbering 6](#_Toc44101346)

[2. Subset Construction 7](#_Toc44101347)

[3. Graph 9](#_Toc44101348)

[SLR PARSING TABLE 10](#_Toc44101349)

[CODE ALGORITHM 11](#_Toc44101350)

[IMPLEMENTATION 12](#_Toc44101351)

[1. Definition of Tokens, Alphabet 12](#_Toc44101352)

[TEST CASES & RESULT 13](#_Toc44101353)

[1. Correct Test Code 13](#_Toc44101354)

[2. Error Test Code 14](#_Toc44101355)

[APPENDIX 15](#_Toc44101356)

[1. First Set 15](#_Toc44101357)

[2. Follow Set 16](#_Toc44101358)

[3. NFA to DFA 17](#_Toc44101359)

[4. Transition Table 20](#_Toc44101360)

# SPECIFICATIONS

|  |
| --- |
| **CFG G:**   1. CODE → VDECL CODE | FDECL CODE | ε 2. VDECL → vtype id semi | vtype ASSIGN semi 3. ASSIGN → id assign RHS 4. FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace 5. ARG → vtype id MOREARGS | ε 6. MOREARGS → comma vtype id MOREARGS | ε 7. BLOCK → STMT BLOCK | ε 8. STMT → VDECL | ASSIGN semi 9. STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE 10. STMT → while lparen COND rparen lbrace BLOCK rbrace 11. STMT → for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 12. ELSE → else lbrace BLOCK rbrace | ε 13. RHS → EXPR | literal 14. EXPR → TERM addsub EXPR | TERM 15. TERM → FACTOR multdiv TERM | FACTOR 16. FACTOR → lparen EXPR rparen | id | num | float 17. COND → FACTOR comp FACTOR 18. RETURN → return FACTOR semi   **Terminals**   1. **vtype** for the types of variables and functions 2. **num** for signed integers 3. **float** for floating-point numbers 4. **literal** for literal strings 5. **id** for the identifiers of variables and functions 6. **if**, **else**, **while**, **for** and **return** for if, else, while, for and return statements respectively 7. **addsub** for + and - arithmetic operators 8. **multdiv** for \* and / arithmetic operators 9. **assign** for assignment operators 10. **comp** for comparison operators 11. **semi** and **comma** for semicolons and commas respectively 12. **lparen**, **rparen**, **lbrace**, and **rbrace** for (, ), {, and } respectively   **Non-terminals**  CODE, VDECL, FDECL, ARG, MOREARGS, BLOCK, STMT, ASSIGN, RHS, EXPR, TERM, FACTOR, COND, RETURN, ELSE  **Start symbol**  CODE |

## Modified part in specifications

In the CFG, we add one more line.

00: S’ → CODE

# FIRST SET

This is the result of First Set. You can check our handwriting version at the [appendix 1](#_1._First_Set).

|  |
| --- |
| First(RETURN) = {return}  First(FACTOR) = {lparen, id, num, float}  First(COND) = {lparen, id, num, float}  Fist(TERM) = {lparen, id, num, float}  First(EXPR) = {lparen, id, num, float}  First(RHS) = {lparen, id, num, float, literal}  First(ELSE) = {else, ε}  First (VDECL) = {vtype}  Fist (ASSIGN) = {id}  First (STMT) = {for, while, if, vtype, id}  First (BLOCK) = {for, while, if, vtype, id, ε}  First (MOREARGS) = {comma, ε}  First (ARG) = {vtype, ε}  First (FDECL) = {vtype}  First (CODE) = {vtype, ε}  First (S’) = {vtype, ε} |

# FOLLOW SET

This is the result of Follow Set. You can check our handwriting version at the [appendix 2](#_2._Follow_Set).

|  |
| --- |
| Follow (S’) = {$}  Follow (CODE) = {$}  Follow (VDECL) = {vtype, rbrace, return, for, while, if, id, $}  Follow (ASSIGN) = {semi, rparen}  Follow (FDECL) = {$, vtype}  Follow (ARG) = {rparen}  Follow (MOREARGS) = {rparen}  Follow (BLOCK) = {rbrace, return}  Follow (STMT) = {rbrace, return, if, while, for, vtype, id}  Follow (ELSE) = {rbrace, return, if, while, for, vtype, id}  Follow (RHS) = {semi, rparen}  Follow (EXPR) = {semi, rparen}  Follow (TERM) = {addsub, rparen, semi}  Follow (FACTOR) = {semi, comp, multdiv, rparen, addsub}  Follow (COND) = {semi, rparen}  Follow (RETURN) = {rbrace} |

# NFA (Non-deterministic Finite Automata)

Our team drew the NAF (Non-deterministic Finite Automata) using flow chart drawing tool[[1]](#footnote-1). The graph is too big, so it is hard to see the detail, so we included the drawing file in the “Handwriting” folder, if you want please check the directory.

## 1. Overall Graph

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

# DFA (Deterministic Finite Automata)

DFA is generated using the NFA which we built. At first, we attached the number on each node and derived the DFA using subset (powerset) construction algorithm. You can check the handwriting version at the [appendix 3](#_3._NFA_to).

## 1. NFA with numbering

This image also can be found at the “Handwriting” directory.

텍스트, 지도, 그리기이(가) 표시된 사진

자동 생성된 설명

## 2. Subset Construction

|  |  |
| --- | --- |
| = T0  = T1  = T2  = T3  = T4  = T5  = T2  = T3  = T4  = T6  = T7  = T8  = T4  = T3  = T2  = T9  = T10  = T11  = T12  = T13  = T14  = T15  = T16  = T17  = T18  = T19  = T20  = T21  = T22  = T23  = T24  = T25  = T26  = T27  = T28  = T18  = T19  = T20  = T21  = T22  = T23  = T29  = T30  = T31  = T32  = T18  = T19  = T20  = T21  = T23  = T58  = T57  = T20  = T21  = T22  = T23  = T59  = T43  = T11  = T10  = T60  = T31  = T61  = T62  = T63  = T64  = T65  = T66  = T67  = T20  = T21  = T22  = T23  = T68  = T69  = T57  = T20  = T21  = T22  = T23  = T70  = T36  = T37  = T38  = T39  = T40  = T41  = T42  = T43  = T71  = T36  = T37  = T38  = T39  = T40  = T41  = T42  = T43  = T72 | = T22  = T23  = T33  = T19  = T20  = T21  = T22  = T23  = T34  = T35  = T36  = T37  = T38  = T39  = T40  = T41  = T42  = T43  = T44  = T45  = T46  = T47  = T36  = T37  = T38  = T39  = T40  = T41  = T42  = T43  = T48  = T49  = T50  = T51  = T52  = T7  = T10  = T53  = T54  = T55  = T20  = T21  = T22  = T23  = T56  = T57  = T20  = T21  = T22  = T73  = T74  = T75  = T43  = T76  = T77  = T78  = T79  = T80  = T81  = T36  = T37  = T38  = T39  = T40  = T41  = T42  = T43  = T82  = T36  = T37  = T38  = T39  = T40  = T41  = T42  = T43  = T83  = T84 |

## 3. Graph

This image also can be found at the “Handwriting” directory.

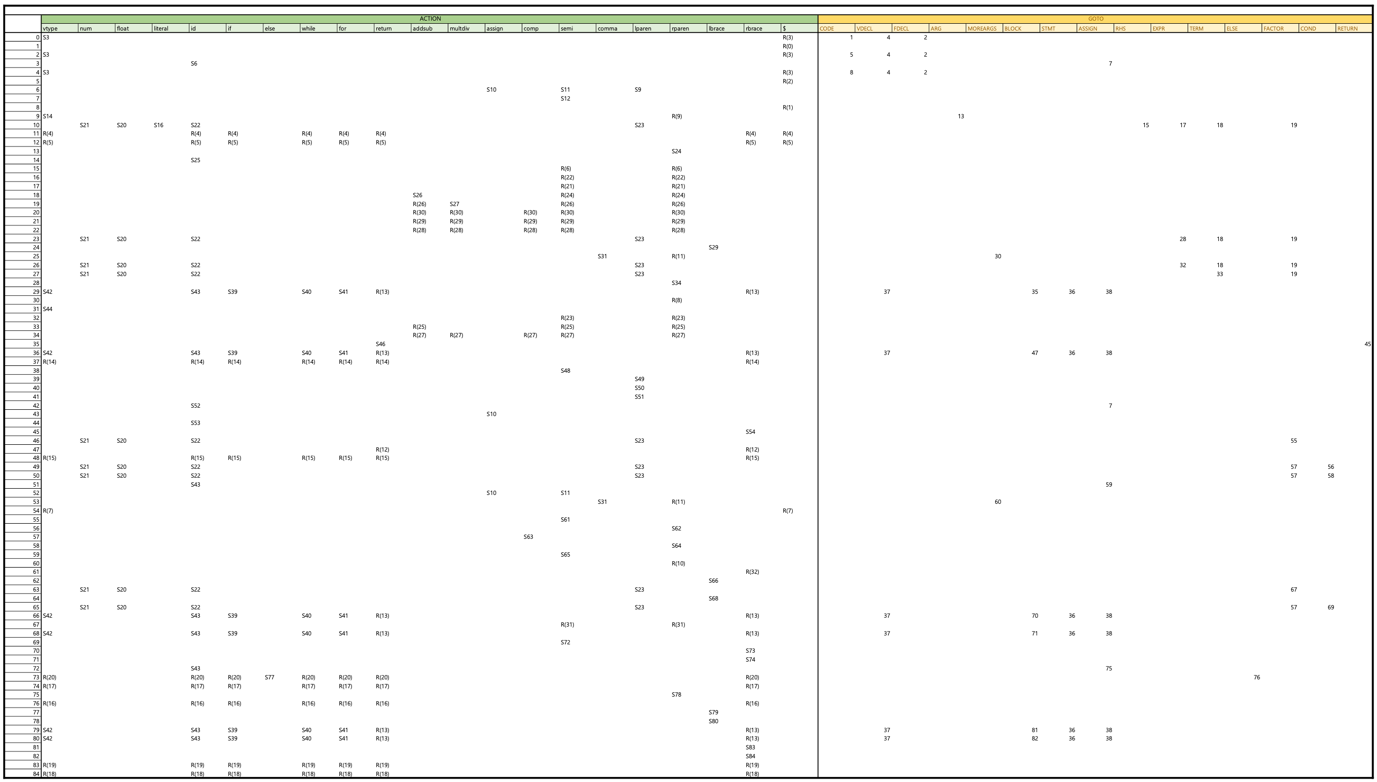
To generate the parsing table, graph node should include the CFG. However, there are too many things to include at the single node, so instead to do that we make the reference list. You can check the list at [appendix 4](#_4._Transition_Table).

텍스트이(가) 표시된 사진

자동 생성된 설명

# SLR PARSING TABLE

The number of terminals is 20, the number of non-terminal is 15, and the number of DFA node is 85. It means that the matrix is huge. When filling in the “reduce” operation, we refer to the rules and follow. You already read the follow set in appendix 2, and you can see the rules in [appendix 5](#_5._Rules). The table is huge to render at the limited space, so we include the SLR parsing table at the “Handwriting” directory.



# CODE ALGORITHM

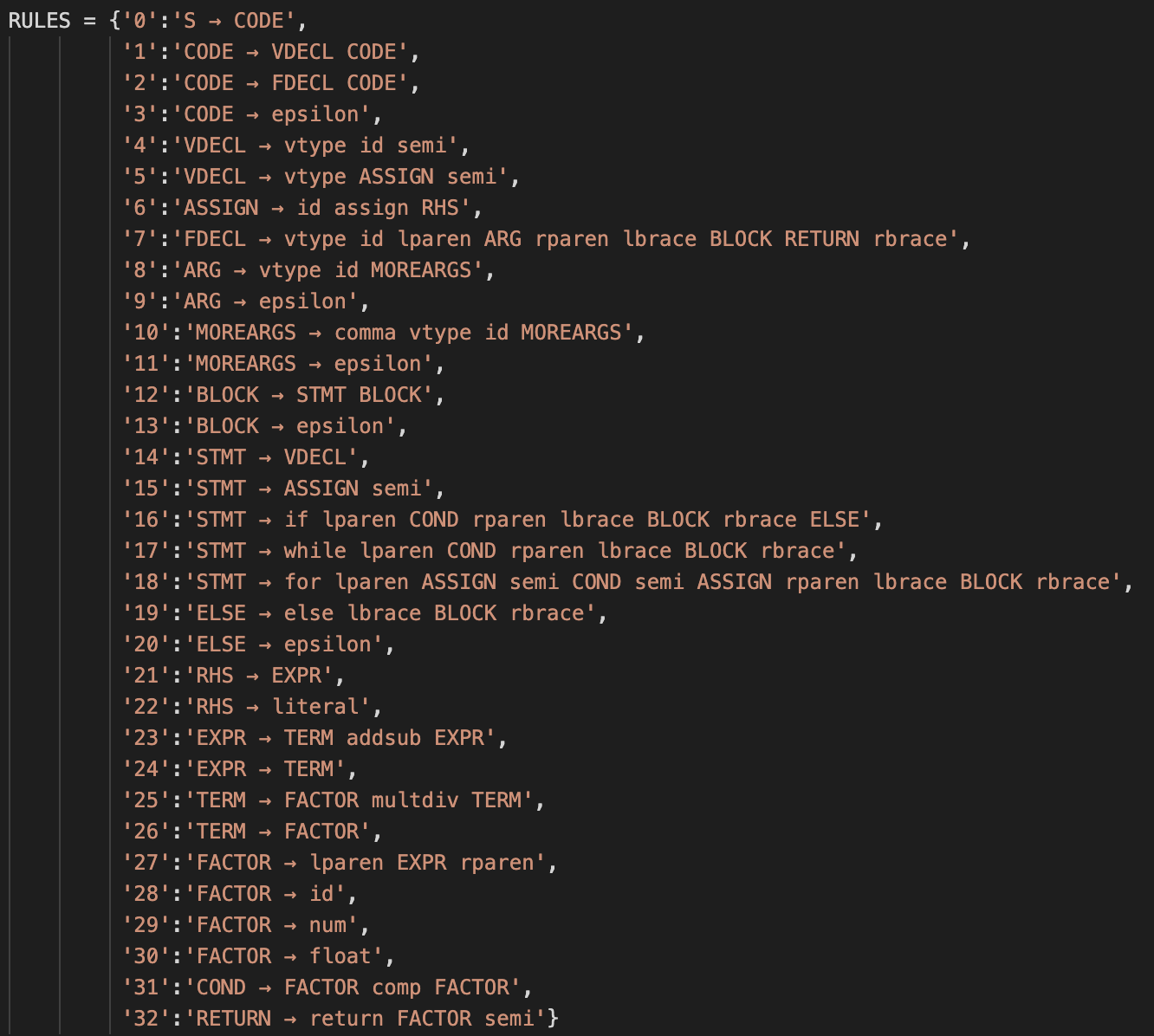
# IMPLEMENTATION

Before explaining our works, we introduce the developing environment.

|  |
| --- |
| **Language**   * Python3 (version: 3.7.4)   **Operating System**   * macOS Catalina * Windows 10   **IDE (Integrated Development Environment)**   * Visual Studio Code (version: 1.45.0) * PyCharm (version: 3.9.4)   **Project Management**   * Git (version: 2.24.2) * Git-Hub |

## 1. Definition of Rules

When calculating the reduce operation, we need to refer to the rules. And there is a relationship between numbers from reduce process and rules. To use this relationship, we implement this using the dictionary structure.



## 2. Definition of SLR Table

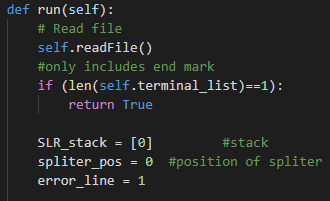
The SLR table has a lot of empty space. Such a matrix is called a sparse matrix. To save memory waste and better readability, we decide to use a list and dictionary. It is the structure in which the list holds a dictionary. A mechanism is simple, using the current state number, get the dictionary, and refer the operation using the next symbol as key.

앉아있는, 테이블, 컴퓨터이(가) 표시된 사진

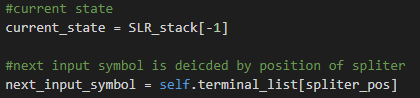
자동 생성된 설명

## 3. Splitter and Stack, Shift and Reduce

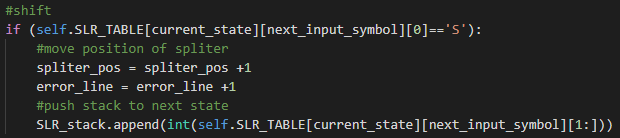
We need to decide position of splitter and define state stack. Stack only has 0, position of splitter is 0 at first.



Current state is decided by top of the stack. And next input symbol is decided by position of splitter.

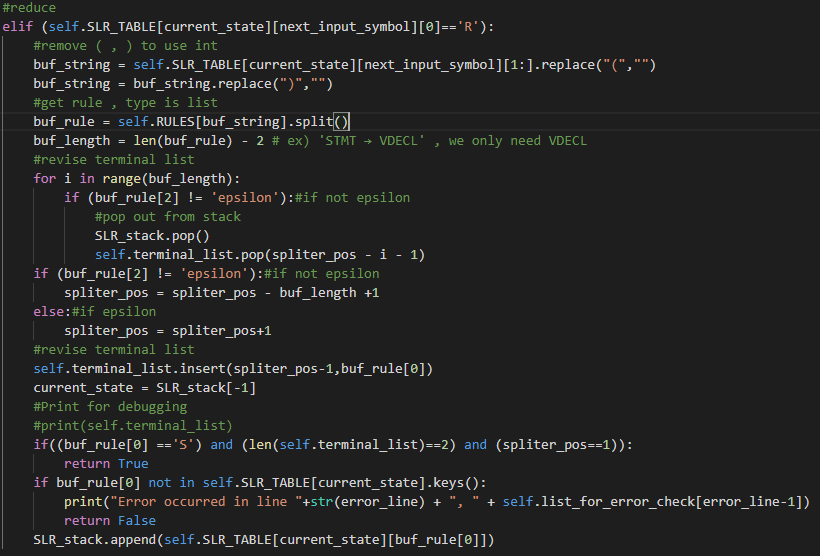


Shift is occurred when the value of SLR TABLE is like S73. It means shift and GOTO 73. 73 is added in our stack and splitter moves forward.



Reduce is occurred when the value of SLR TABLE is like R(19). It means reduce by the rule number 19. The number of right hand side of rule is popped out from our stack. If right hand side is epsilon, popped out nothing. And revise our total input list, and also revise position of splitter because the length of total input list is changed.

Let’s move on next step. We need to update current state that is decided by top of the stack. And, we check it is acceptable. If it includes start dummy symbol ‘S’, then it is accepted. If not, do next step GOTO. Add next state into our stack.



# TEST CASES & RESULT

## 1. Correct Test Code

|  |  |
| --- | --- |
| **Input** | **Result** |
| **텍스트, 화면, 테이블이(가) 표시된 사진  자동 생성된 설명** | 텍스트이(가) 표시된 사진  자동 생성된 설명텍스트이(가) 표시된 사진  자동 생성된 설명 |

## 2. Error Test Code

* In our lexical analyzer, we don’t allow to use the character ‘$’.

|  |  |
| --- | --- |
| **Input** | **Result** |
| 검은색이(가) 표시된 사진  자동 생성된 설명 |  |

* ‘=!’ isn’t correct comparison.

|  |  |
| --- | --- |
| **Input** | **Result** |
|  |  |

* ‘.0’ isn’t correct float.

|  |  |
| --- | --- |
| **Input** | **Result** |
| 검은색, 어두운, 화면, 오렌지이(가) 표시된 사진  자동 생성된 설명 |  |

* ‘1.10’ isn’t correct float. In our lexical analyzer, the right side of a decimal point must be a single digit 0 or a non-empty sequence terminating with a non-zero digit.

|  |  |
| --- | --- |
| **Input** | **Result** |
| 검은색, 어두운, 앉아있는, 측정기이(가) 표시된 사진  자동 생성된 설명 |  |

# APPENDIX

## 1. First Set

텍스트이(가) 표시된 사진

자동 생성된 설명

## 2. Follow Set

텍스트이(가) 표시된 사진

자동 생성된 설명

## 3. NFA to DFA

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

## 4. DFA Node Name List

|  |  |
| --- | --- |
| T0 | 1 S' -> .CODE  3 CODE ->.  4 CODE -> .FDECL CODE  5 CODE -> .VDECL CODE  10 FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi |
| T1 | 2 S' -> CODE. |
| T2 | 3 CODE ->.  4 CODE -> .FDECL CODE  5 CODE -> .VDECL CODE  8 CODE -> FDECL. CODE  10 FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi |
| T3 | 11 FDECL -> vtype. id lparen ARG rparen lbrace BLOCK RETURN rbrace  85 VDECL -> vtype. id semi  88 VDECL -> vtype. ASSIGN semi  91 ASSIGN -> .id assign RHS |
| T4 | 3 CODE ->.  4 CODE -> .FDECL CODE  5 CODE -> .VDECL CODE  6 CODE -> VDECL. CODE  10 FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi |
| T5 | 9 CODE -> FDECL CODE. |
| T6 | 12 FDECL -> vtype id. lparen ARG rparen lbrace BLOCK RETURN rbrace  86 VDECL -> vtype id. semi  92 ASSIGN -> id. assign RHS |
| T7 | 89 VDECL -> vtype ASSIGN. semi |
| T8 | 7 CODE -> VDECL CODE. |
| T9 | 13 FDECL -> vtype id lparen. ARG rparen lbrace BLOCK RETURN rbrace  28 ARG -> .  29 ARG -> .vtype id MOREARGS |
| T10 | 93 ASSIGN -> id assign. RHS  96 RHS -> .literal  97 RHS -> .EXPR  100 EXPR -> .TERM  101 EXPR -> .TERM addsub EXPR  106 TERM -> .FACTOR  107 TERM -> .FACTOR multdiv TERM  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T11 | 87 VDECL -> vtype id semi. |
| T12 | 90 VDECL -> vtype ASSIGN semi. |
| T13 | 14 FDECL -> vtype id lparen ARG. rparen lbrace BLOCK RETURN rbrace |
| T14 | 30 ARG -> vtype. id MOREARGS |
| T15 | 94 ASSIGN -> id assign RHS. |
| T16 | 95 RHS -> literal. |
| T17 | 98 RHS -> EXPR . |
| T18 | 99 EXPR -> TERM.  102 EXPR -> TERM. addsub EXPR |
| T19 | 105 TERM -> FACTOR.  108 TERM -> FACTOR. multdiv TERM |
| T20 | 112 FACTOR -> float. |
| T21 | 114 FACTOR -> num. |
| T22 | 116 FACTOR -> id. |
| T23 | 100 EXPR -> .TERM  101 EXPR -> .TERM addsub EXPR  106 TERM -> .FACTOR  107 TERM -> .FACTOR multdiv TERM  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen  118 FACTOR -> lparen. EXPR rparen |
| T24 | 15 FDECL -> vtype id lparen ARG rparen. lbrace BLOCK RETURN rbrace |
| T25 | 31 ARG -> vtype id. MOREARGS  33 MOREARGS -> .  34 MOREARGS -> .comma vtype id MOREARGS |
| T26 | 100 EXPR -> .TERM  101 EXPR -> .TERM addsub EXPR  103 EXPR -> TERM addsub. EXPR  106 TERM -> .FACTOR  107 TERM -> .FACTOR multdiv TERM  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T27 | 106 TERM -> .FACTOR  107 TERM -> .FACTOR multdiv TERM  109 TERM -> FACTOR multdiv. TERM  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T28 | 119 FACTOR -> lparen EXPR. rparen |
| T29 | 16 FDECL -> vtype id lparen ARG rparen lbrace. BLOCK RETURN rbrace  24 BLOCK -> .  25 BLOCK -> .STMT BLOCK  39 STMT -> .VDECL  40 STMT -> .ASSIGN semi  41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE  42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace  43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi  91 ASSIGN -> .id assign RHS |
| T30 | 32 ARG -> vtype id MOREARGS. |
| T31 | 35 MOREARGS -> comma. vtype id MOREARGS |
| T32 | 104 EXPR -> TERM addsub EXPR. |
| T33 | 110 TERM -> FACTOR multdiv TERM. |
| T34 | 120 FACTOR -> lparen EXPR rparen. |
| T35 | 17 FDECL -> vtype id lparen ARG rparen lbrace BLOCK. RETURN rbrace  20 RETURN -> .return FACTOR semi |
| T36 | 24 BLOCK -> .  25 BLOCK -> .STMT BLOCK  26 BLOCK -> STMT. BLOCK  39 STMT -> .VDECL  40 STMT -> .ASSIGN semi  41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE  42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace  43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi  91 ASSIGN -> .id assign RHS |
| T37 | 44 STMT -> VDECL. |
| T38 | 45 STMT -> ASSIGN. Semi |
| T39 | 47 STMT -> if. lparen COND rparen lbrace BLOCK rbrace ELSE |
| T40 | 61 STMT -> while. lparen COND rparen lbrace BLOCK rbrace |
| T41 | 72 STMT -> for. lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace |
| T42 | 85 VDECL -> vtype. id semi  88 VDECL -> vtype. ASSIGN semi  91 ASSIGN -> .id assign RHS |
| T43 | 92 ASSIGN -> id. assign RHS |
| T44 | 36 MOREARGS -> comma vtype. id MOREARGS |
| T45 | 18 FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN. rbrace |
| T46 | 21 RETURN -> return. FACTOR semi  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T47 | 27 BLOCK -> STMT BLOCK . |
| T48 | 46 STMT -> ASSIGN semi. |
| T49 | 48 STMT -> if lparen. COND rparen lbrace BLOCK rbrace ELSE  68 COND -> .FACTOR comp FACTOR  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T50 | 62 STMT -> while lparen. COND rparen lbrace BLOCK rbrace  68 COND -> .FACTOR comp FACTOR  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T51 | 73 STMT -> for lparen. ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  91 ASSIGN -> .id assign RHS |
| T52 | 86 VDECL -> vtype id. semi  92 ASSIGN -> id. assign RHS |
| T53 | 33 MOREARGS -> .  34 MOREARGS -> .comma vtype id MOREARGS  37 MOREARGS -> comma vtype id. MOREARGS |
| T54 | 19 FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace. |
| T55 | 22 RETURN -> return FACTOR. semi |
| T56 | 49 STMT -> if lparen COND. rparen lbrace BLOCK rbrace ELSE |
| T57 | 69 COND -> FACTOR. comp FACTOR |
| T58 | 63 STMT -> while lparen COND. rparen lbrace BLOCK rbrace |
| T59 | 74 STMT -> for lparen ASSIGN. semi COND semi ASSIGN rparen lbrace BLOCK rbrace |
| T60 | 38 MOREARGS -> comma vtype id MOREARGS. |
| T61 | 23 RETURN -> return FACTOR semi. |
| T62 | 50 STMT -> if lparen COND rparen. lbrace BLOCK rbrace ELSE |
| T63 | 70 COND -> FACTOR comp. FACTOR  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T64 | 64 STMT -> while lparen COND rparen. lbrace BLOCK rbrace |
| T65 | 68 COND -> .FACTOR comp FACTOR  75 STMT -> for lparen ASSIGN semi. COND semi ASSIGN rparen lbrace BLOCK rbrace  111 FACTOR -> .float  113 FACTOR -> .num  115 FACTOR -> .id  117 FACTOR -> .lparen EXPR rparen |
| T66 | 24 BLOCK -> .  25 BLOCK -> .STMT BLOCK  39 STMT -> .VDECL  40 STMT -> .ASSIGN semi  41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE  42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace  43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi  91 ASSIGN -> .id assign RHS |
| T67 | 71 COND -> FACTOR comp FACTOR. |
| T68 | 24 BLOCK -> .  25 BLOCK -> .STMT BLOCK  39 STMT -> .VDECL  40 STMT -> .ASSIGN semi  41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE  42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace  43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi  91 ASSIGN -> .id assign RHS |
| T69 | 76 STMT -> for lparen ASSIGN semi COND. semi ASSIGN rparen lbrace BLOCK rbrace |
| T70 | 52 STMT -> if lparen COND rparen lbrace BLOCK. rbrace ELSE |
| T71 | 66 STMT -> while lparen COND rparen lbrace BLOCK. Rbrace |
| T72 | 77 STMT -> for lparen ASSIGN semi COND semi. ASSIGN rparen lbrace BLOCK rbrace  91 ASSIGN -> .id assign RHS |
| T73 | 53 STMT -> if lparen COND rparen lbrace BLOCK rbrace. ELSE  55 ELSE -> .  56 ELSE -> .else lbrace BLOCK rbrace |
| T74 | 67 STMT -> while lparen COND rparen lbrace BLOCK rbrace. |
| T75 | 78 STMT -> for lparen ASSIGN semi COND semi ASSIGN. rparen lbrace BLOCK rbrace |
| T76 | 54 STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE. |
| T77 | 57 ELSE -> else. lbrace BLOCK rbrace |
| T78 | 79 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen. lbrace BLOCK rbrace |
| T79 | 24 BLOCK -> .  25 BLOCK -> .STMT BLOCK  39 STMT -> .VDECL  40 STMT -> .ASSIGN semi  41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE  42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace  43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  58 ELSE -> else lbrace. BLOCK rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi  91 ASSIGN -> .id assign RHS |
| T80 | 24 BLOCK -> .  25 BLOCK -> .STMT BLOCK  39 STMT -> .VDECL  40 STMT -> .ASSIGN semi  41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE  42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace  43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace  80 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen lbrace. BLOCK rbrace  83 VDECL -> .vtype ASSIGN semi  84 VDECL -> .vtype id semi  91 ASSIGN -> .id assign RHS |
| T81 | 59 ELSE -> else lbrace BLOCK. rbrace |
| T82 | 81 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK. rbrace |
| T83 | 60 ELSE -> else lbrace BLOCK rbrace. |
| T84 | 82 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace. |

## 5. Rules

|  |
| --- |
| 1. S’ → CODE 2. CODE → VDECL CODE 3. CODE → FDECL CODE 4. CODE → ε 5. VDECL → vtype id semi 6. VDECL → vtype ASSIGN semi 7. ASSIGN → id assign RHS 8. FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace 9. ARG → vtype id MOREARGS 10. ARG → ε 11. MOREARGS → comma vtype id MOREARGS 12. MOREARGS → ε 13. BLOCK → STMT BLOCK 14. BLOCK → ε 15. STMT → VDECL 16. STMT → ASSIGN semi 17. STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE 18. STMT → while lparen COND rparen lbrace BLOCK rbrace 19. STMT → for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 20. ELSE → else lbrace BLOCK rbrace 21. ELSE → ε 22. RHS → EXPR 23. RHS → literal 24. EXPR → TERM addsub EXPR 25. EXPR → TERM 26. TERM → FACTOR multdiv TERM 27. TERM → FACTOR 28. FACTOR → lparen EXPR rparen 29. FACTOR → id 30. FACTOR → num 31. FACTOR → float 32. COND → FACTOR comp FACTOR 33. RETURN → return FACTOR semi |

1. https://app.diagrams.net [↑](#footnote-ref-1)