|  |
| --- |
| 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](#_Toc42959048)

[SPECIFICATIONS 1](#_Toc42959049)

[1. Overall Graph 2](#_Toc42959050)

[NFA (Non-deterministic Finite Automata) 3](#_Toc42959051)

[1. Overall Graph 3](#_Toc42959052)

[DFA (Deterministic Finite Automata) 4](#_Toc42959053)

[1. ID 4](#_Toc42959054)

[2. INTEGER 6](#_Toc42959055)

[3. FLOAT 7](#_Toc42959056)

[4. LITERAL 9](#_Toc42959057)

[TROUBLE & SOLUTION 11](#_Toc42959058)

[1. Longest matching 11](#_Toc42959059)

[2. Period 12](#_Toc42959060)

[3. Error 13](#_Toc42959061)

[IMPLEMENTATION 14](#_Toc42959062)

[1. Definition of Tokens, Alphabet 14](#_Toc42959063)

[2. ID-DFA 15](#_Toc42959064)

[3. INTEGER - DFA 16](#_Toc42959065)

[4. FLOAT - DFA 17](#_Toc42959066)

[5. LITERAL - DFA 18](#_Toc42959067)

[6. Other Tokens 19](#_Toc42959068)

[7. Other - File I/O 20](#_Toc42959069)

[TEST CASES & RESULT 21](#_Toc42959070)

[1. Correct Test Code 21](#_Toc42959071)

[2. Error Test Code 22](#_Toc42959072)

[APPENDIX 24](#_Toc42959073)

[1. NFA to DFA with transition table 24](#_Toc42959074)

[2. Git-Hub 29](#_Toc42959075)

# 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  **Start symbol**  CODE |

## Modified part in specifications

In the CFG, we add one more line.

00: GOAL -> CODE

# FIRST SET

This is the result of First Set. You can check our handwriting version at the appendix 1.

|  |
| --- |
| 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 (GOAL) = {vtype, ε} |

# FOLLOW SET

This is the result of Follow Set. You can check our handwriting version at the appendix 2.

|  |
| --- |
| Follow (GOAL) = {$}  Follow (CODE) = {$}  Follow (VDECL) = {ε, vtype, if, while, for, id}  Follow (ASSIGN) = {semi, rparen}  Follow (FDECL) = {ε, vtype}  Follow (ARG) = {rparen}  Follow (MOREARGS) = {rparen}  Follow (BLOCK) = {rbrace, return}  Follow (STMT) = {ε, if, while, ofr, vtype, id}  Follow (ELSE) = {ε, if, while, ofr, 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 attached the drawing file in the <이름 나중에 정할것> folder, if you want please check the directory.

## 1. Overall Graph

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

## DFA (Deterministic Finite Automata)

We built DFA graph and transition table using subset (powerset) construction algorithm.

If you want to see the hand-writing paper, please check the appendix 1.

## 1. ID

모니터, 검은색, 화면, 텔레비전이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {A, B, C} = T0 |
| = = {D, F, G, H, I, J, K, L, R} = T1  =  =  = = {E, F, G, H, I, J, K, L, R} = T2 |
| = = {M, Q, R, H, I, J, K, L} = T3  = = {N, Q, R, H, I, J, K, L} = T4  = = {O, Q, R, H, I, J, K, L} = T5  = = {P, Q, R, H, I, J, K, L} = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |

* Transition table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | letter | zero | non-zero | \_ |
| * T0 | T1 |  |  | T2 |
| *T1* | T3 | T4 | T5 | T6 |
| *T2* | T3 | T4 | T5 | T6 |
| *T3* | T3 | T4 | T5 | T6 |
| *T4* | T3 | T4 | T5 | T6 |
| *T5* | T3 | T4 | T5 | T6 |
| *T6* | T3 | T4 | T5 | T6 |

* DFA Graph

검은색, 옅은, 빨간색, 얼굴이(가) 표시된 사진

자동 생성된 설명

## 2. INTEGER

텔레비전, 화면, 모니터, 검은색이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {A, B, C, D, E, G, H, I} = T0 |
| = = {F, H, I} = T1  = = {T, S} =T2  = = {J, K, L, M, N, R, S} =T3 |
| =  =  = = T3 |
| =  =  =  = |
| = {O, Q, L, M, N, R, S} = T4  = = {P, Q, L, M, N, R, S} = T5 |
| =  = T4  = = T5 |
| =  = T4  = = T5 |

* DFA Graph
* 실외, 잔디, 빨간색, 평야이(가) 표시된 사진

  자동 생성된 설명Transition table

|  |  |  |  |
| --- | --- | --- | --- |
|  | - | zero | non-zero |
| * T0 | T1 | T2 | T3 |
| T1 |  |  | T3 |
| *T2* |  |  |  |
| *T3* |  | T4 | T5 |
| *T4* |  | T4 | T5 |
| *T5* |  | T4 | T5 |

## 3. FLOAT

컴퓨터, 검은색, 옅은, 노트북이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {1, 2, 3, 4, 6, 7, 8, 10} = T0 |
| = = {5, 6, 7, 8, 10} = T1  = = {9, 20, 21} =T2  = = {11, 12, 13, 14, 16, 19, 20, 21} =T3  = |
| =  = = T2  = = T3  = |
| =  =  =  = = {22, 23, 24, 30, 31, 32, 34, 37, 38} = T4 |
| =  = {13, 14, 15, 16, 18, 19, 20, 21} = T5  = {13, 14, 16, 17, 18, 19, 20, 21} = T6  = = T4 |
| =  = {25, 31, 32, 33, 34, 36, 37, 38, 40} = T7  = = {31, 32, 34, 35, 36, 37, 38, 39, 40} = T8  = |
| =  = T5  = = T6  = = T4 |
| =  = T5  = = T6  = = T4  = |

= {31, 32, 33, 34, 36, 37, 38} = T9

|  |
| --- |
| = = T8  = |
| =  = T9  = = T8  = |
| =  = T9  = = T8  = |

* Transition table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | - | zero | . | non-zero |
| * T0 | T1 | T2 |  | T3 |
| T1 |  | T2 |  | T3 |
| T2 |  |  | T4 |  |
| T3 |  | T5 | T4 | T6 |
| T4 |  | T7 |  | T8 |
| T5 |  | T5 | T4 | T6 |
| T6 |  | T5 | T4 | T6 |
| T7 |  | T9 |  | T8 |
| T8 |  | T9 |  | T8 |
| T9 |  | T9 |  | T8 |

* DFA Graph

검은색, 샷, 어두운, 잔디이(가) 표시된 사진

자동 생성된 설명

## 4. LITERAL

텔레비전, 화면, 검은색, 모니터이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {A} = T0 |
| = = {B, C, D, E, F, G, H, N, O} = T1  =  =  =  = |
| = = {P} = T2  = = {I, M, N, O, D, E, F, G, H} = T3  = = {J, M, N, O, D, E, F, G, H} = T4  = = {K, M, N, O, D, E, F, G, H} = T5  = = {L, M, N, O, D, E, F, G, H} = T6 |
| =  =  =  =  = |
| = = T2  = = T3  = T4  = = T5  = = T6 |
| = = T2  = = T3  = T4  = = T5  = = T6 |
| = = T2  = = T3  = T4  = = T5  = = T6 |
| = = T2  = = T3  = T4  = = T5  = = T6 |

* Transition table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | “ | letter | zero | non-zero |  |
| * T0 | T1 |  |  |  |  |
| T1 | T2 | T3 | T4 | T5 | T6 |
| *T2* |  |  |  |  |  |
| T3 | T2 | T3 | T4 | T5 | T6 |
| T4 | T2 | T3 | T4 | T5 | T6 |
| T5 | T2 | T3 | T4 | T5 | T6 |
| T6 | T2 | T3 | T4 | T5 | T6 |

* DFA Graph

검은색, 빨간색, 잔디, 화면이(가) 표시된 사진

자동 생성된 설명

# TROUBLE & SOLUTION

## 1. Longest matching

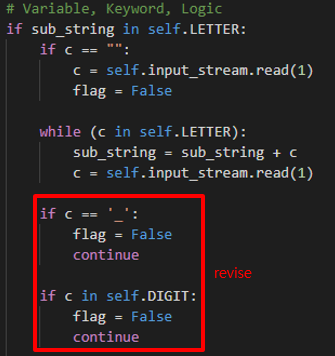
* Trouble

When we tested identifier that includes keyword and variable (e.g. if, while, int, char), longest matching didn’t work well.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Result |
| 검은색, 앉아있는, 화면, 전화이(가) 표시된 사진  자동 생성된 설명 | ……  VARIABLE int  ID int\_ABC  ASSIGN =  …… | ……    …… |

* Solution

We add more if condition in variable, keyword, logic part so that our program can do longest matching.



* Result

|  |  |  |
| --- | --- | --- |
| Input | Previous result | Revised result |
|  | ……    …… |  |

## 2. Period

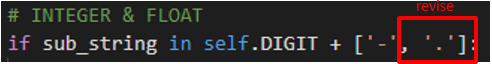
* Trouble

When we tested . (Period) without any prefix, our program is finished that line.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Result |
|  | Line2 error! | Program is finished unexpected. |

* Solution

At first, our program didn’t include the condition that what we did if the first substring is . (Period). So, we add . (Period) condition in our program.



* Result

|  |  |  |
| --- | --- | --- |
| Input | Previous result | Revised result |
|  | Program is finished unexpected. |  |

## 3. Error

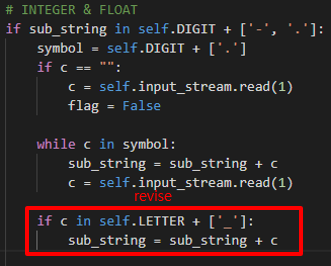
* Trouble

When we tested wrong inputs like 33abc, 3.3test, our program didn’t get the error.

|  |  |  |
| --- | --- | --- |
| Input | Expected result | Result |
| 어두운, 검은색, 시계, 오렌지이(가) 표시된 사진  자동 생성된 설명 | Line2 error! | ……    …… |

* Solution

We add more if condition in int, float part so that our program can do longest matching.



* Result

|  |  |  |
| --- | --- | --- |
| Input | Previous result | Revised result |
|  | ……    …… |  |

# IMPLEMENTATION

Before explaining our works, we will introduce the developing environment and how to manage our project, please check the appendix-2.

|  |
| --- |
| **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 |

We defined the four tokens: ID, INTEGER, FLOAT, LITERAL as function, and the others are defined as a list type variable. In addition, we defined five symbols; LETTER, SYMBOL, ZERO, NON\_ZERO and DIGIT to improve the productivity and readability.

## 1. Definition of Tokens, Alphabet

We defined the four tokens: ID, INTEGER, FLOAT, LITERAL as function, and the others are defined as a list type variable. In addition, we defined five symbols; LETTER, SYMBOL, ZERO, NON\_ZERO and DIGIT to improve the productivity and readability.

스크린샷이(가) 표시된 사진

자동 생성된 설명

## 2. ID-DFA

스크린샷이(가) 표시된 사진

자동 생성된 설명

This part of code is for ID DFA that decides whether the input\_string is ID or not. This method has 7 states, and the start state is 0. In a for-loop, it works exactly the same as the ID DFA that was designed in the previous page. This method reads input\_string in order, and a recent state is changed by input\_string. If it is finished in the final states, which are 1 to 6, the input\_string will be accepted. Otherwise, the input\_string will be denied, and it means the input\_string isn’t ID.

## 3. INTEGER - DFA

스크린샷, 앉아있는, 테이블, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

This part of code is for INT DFA that decides whether the input\_string is int or not. This method has 6 states, and the start state is T0. In a for-loop, it works exactly the same as the INT DFA that was designed in the previous page. This method reads input\_string in order, and a recent state is changed by input\_string. If it is finished in the final states, which are state2, state3, state4, and state5, the input\_string will be accepted. Otherwise, the input\_string will be denied, and it means input\_string isn’t int.

## 4. FLOAT - DFA

스크린샷, 앉아있는, 테이블, 노트북이(가) 표시된 사진

자동 생성된 설명전화이(가) 표시된 사진

자동 생성된 설명

This part of the code is for FLOAT DFA that decides whether the input\_string is float or not. This method has 10 states, and the start state is T0. In a for-loop, it works exactly the same as the FLOAT DFA that was designed in the previous page. This method reads input\_string in order, and a recent state is changed by input\_string. If it is finished in the final states, which are state7 and state8, the input\_string will be accepted. Otherwise, the input\_string will be denied, and it means input\_string isn’t a float.

## 5. LITERAL - DFA

스크린샷이(가) 표시된 사진

자동 생성된 설명

This part of code is for LITERAL DFA that decides whether the input\_string is a string or not. This method has 7 states, and the start state is 0. In a for-loop, it works exactly the same as the LITERAL DFA that was designed in the previous page. This method reads input\_string in order, and a recent state is changed by input\_string. If it is finished in the final state, which is 2, the input\_string will be accepted. Otherwise, the input\_string will be denied, and it means input\_string isn’t a string.

## 6. Other Tokens

The software checks the other tokens following matching processing.

|  |  |
| --- | --- |
| * Variable, Keyword, Logic   텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명   * ASSIGN, COMPARISON   스크린샷이(가) 표시된 사진  자동 생성된 설명 | * BRACE, PAREN, TERM, COMMA, OPERATOR, COMPARISON   텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |

* Subtract

그리기이(가) 표시된 사진

자동 생성된 설명

## 7. Other - File I/O

* Read File

|  |  |
| --- | --- |
| 쥐고있는, 남자이(가) 표시된 사진  자동 생성된 설명 | 시계, 측정기, 검은색, 표지판이(가) 표시된 사진  자동 생성된 설명 |

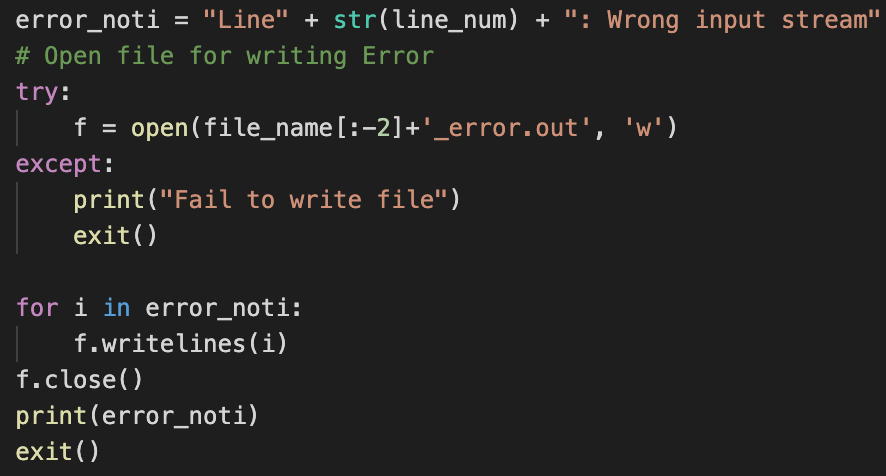
* Write File

텍스트이(가) 표시된 사진

자동 생성된 설명

* Error Case

If the code is written as out of correct grammar, the software returns the error message and creates the error message file as followed format FIEL\_NAME\_error.out. There are 5 points where check the error; each of them use the same code, so we attached a single picture.



# 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 |  |

* ‘a.a’ isn’t correct input.

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

* ‘12people’ isn’t correct ID. ID should be started with English letter or ‘\_’.

|  |  |
| --- | --- |
| **Input** | **Result** |
| 어두운, 사진, 오렌지, 모니터이(가) 표시된 사진  자동 생성된 설명 |  |

* ‘013’ isn’t corret int.

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

* ‘-0’ isn’t correct int.

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

* Literal string should be terminated with a symbol “.

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

# APPENDIX

## 1. First Set

텍스트이(가) 표시된 사진

자동 생성된 설명

## 2. Follow Set

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