COMPILER PROJECT I: Lexical Analyzer

Documentation

20184690 장성현

# 1. Definition of tokens and their regular expressions

Some regular expressions in advance:

*letter* = A|B|C…|Y|Z|a|b|c|..|y|z *natural* = 1|2|3|4|5|6|7|8|9 *whitespace*= \n| |\t

* 1. **VTYPE**

Definition) variable types.

Regular Expression) VTYPE = int | char | bool | float

* 1. **INTEGER**

Definition) integer variable.

Regular Expression) INTEGER = (-|ε)natural(0|natural)\* | 0

* 1. **LITERAL**

Definition) literal variable.

Regular Expression) LITERAL = “(letter| |0|natural)\* ”

* 1. **BOOLEAN**

Definition) Boolean variable.

Regular Expression) BOOLEAN = true | false

* 1. **REAL**

Definition) floating-point number variable.

Regular Expression)

REAL = (-|ε) (0|natural(0|natural)\*). (0|(0|natural)\* natural)

* 1. **ID**

Definition) an identifier.

Regular Expression) ID = (letter|\_)(letter|0|natural\_)\*

* 1. KEYWORDS

|  |  |  |
| --- | --- | --- |
| Token Name | Definition | Regular Expression |
| **IF** | if for if statement | IF = if |
| **ELSE** | else for else statement | ELSE = else |
| **WHILE** | while for while-loop statement | WHILE = while |
| **FOR** | for for for-loop statement | FOR = for |
| **RETURN** | return for return statement | RETURN = return |

* 1. OPERATORS

|  |  |  |
| --- | --- | --- |
| Token Name | Definition | Regular Expression |
| **ARITHM** | Arithmetic operators | ARITHM = +|-|\*|/ |
| **BITWISE** | Bitwise operators | BITWISE = <<|>>|&|| |
| **COMP** | Comparison operators | COMP = <|>|==|!=|<=|>= |
| **ASSIGN** | Assignment operator | ASSIGN = = |

* 1. SYMBOLS

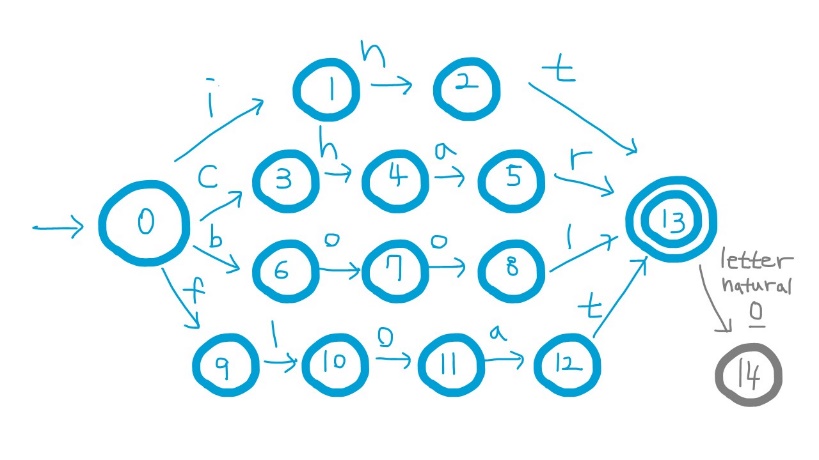
|  |  |  |
| --- | --- | --- |
| Token Name | Definition | Regular Expression |
| **SEMI** | A terminating symbol of statements | SEMI = ; |
| **LBRKT** | A pair of symbols for defining area/scope of variables and functions | LBRKT = { |
| **RBRKT** | RBRKT = } |
| **LPAREN** | A pair of symbols for indicating a function/statement | LPAREN = ( |
| **RPAREN** | RPAREN = ) |
| **COMMA** | A symbol for separating input arguments in functions | COMMA = , |

# 2. DFA transition graph or table for recognizing the regular expressions

DFA transition table에서 노란색 표시는 Final State를 의미합니다.

* 1. **VTYPE**

VTYPE = int | char | bool | float

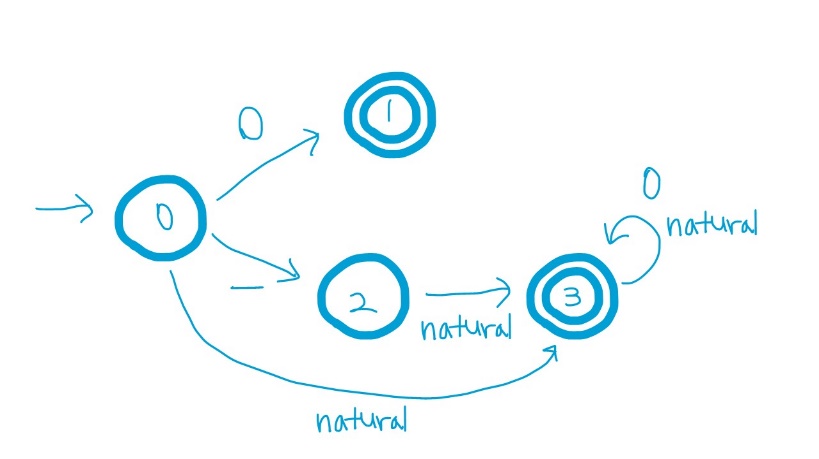


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i | n | t | c | h | a | r | b | o | l | **f** | **letter** | **0** | **natural** | **\_** |
| **T0** | T1 |  |  | T3 |  |  |  | T6 |  |  | T9 |  |  |  |  |
| **T1** |  | T2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **T2** |  |  | T13 |  |  |  |  |  |  |  |  |  |  |  |  |
| **T3** |  |  |  |  | T4 |  |  |  |  |  |  |  |  |  |  |
| **T4** |  |  |  |  |  | T5 |  |  |  |  |  |  |  |  |  |
| **T5** |  |  |  |  |  |  | T13 |  |  |  |  |  |  |  |  |
| **T6** |  |  |  |  |  |  |  |  | T7 |  |  |  |  |  |  |
| **T7** |  |  |  |  |  |  |  |  | T8 |  |  |  |  |  |  |
| **T8** |  |  |  |  |  |  |  |  |  | T13 |  |  |  |  |  |
| **T9** |  |  |  |  |  |  |  |  |  | T10 |  |  |  |  |  |
| **T10** |  |  |  |  |  |  |  |  | T11 |  |  |  |  |  |  |
| **T11** |  |  |  |  |  | T12 |  |  |  |  |  |  |  |  |  |
| **T12** |  |  | T13 |  |  |  |  |  |  |  |  |  |  |  |  |
| **T13** |  |  |  |  |  |  |  |  |  |  |  | T14 | T14 | T14 | T14 |
| **T14** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

(빈칸은 모두 Φ)

* 1. **INTEGER**

INTEGER = (-|ε)natural(0|natural)\* | 0

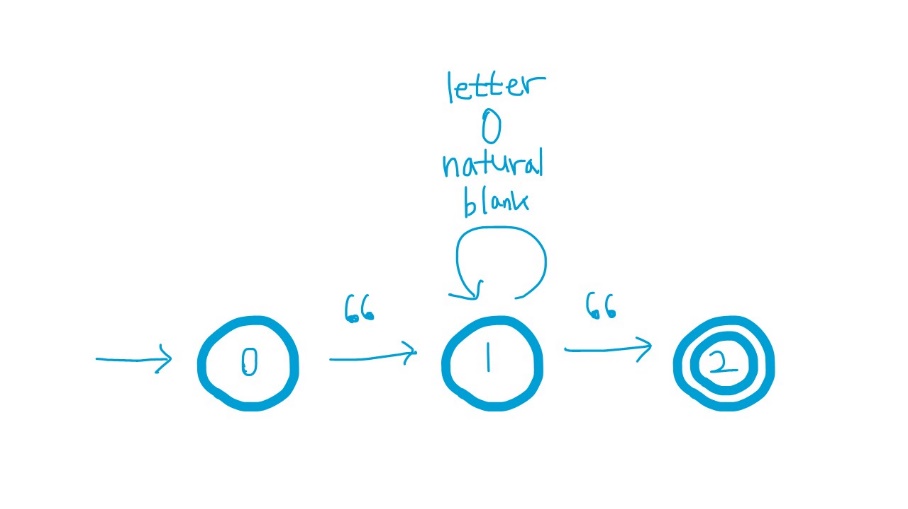


|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **-** | **natural** |
| **T0** | T1 | T2 | T3 |
| **T1** |  |  |  |
| **T2** |  |  | T3 |
| **T3** | T3 |  | T3 |

(빈칸은 모두 Φ)

* 1. **LITERAL**

LITERAL = “(letter| |0|natural)\* ”

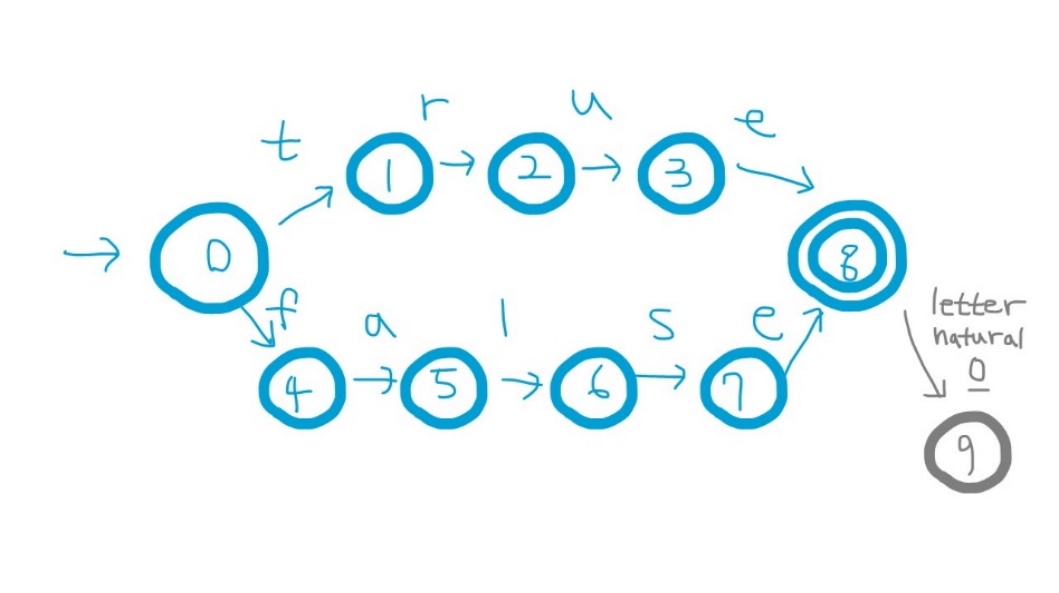
****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **“** | **letter** | **0** | **natural** | **blank** |
| **T0** | T1 |  |  |  |  |
| **T1** | T2 | T1 | T1 | T1 | T1 |
| **T2** |  |  |  |  |  |

(빈칸은 모두 Φ)

* 1. **BOOLEAN**

BOOLEAN = true | false

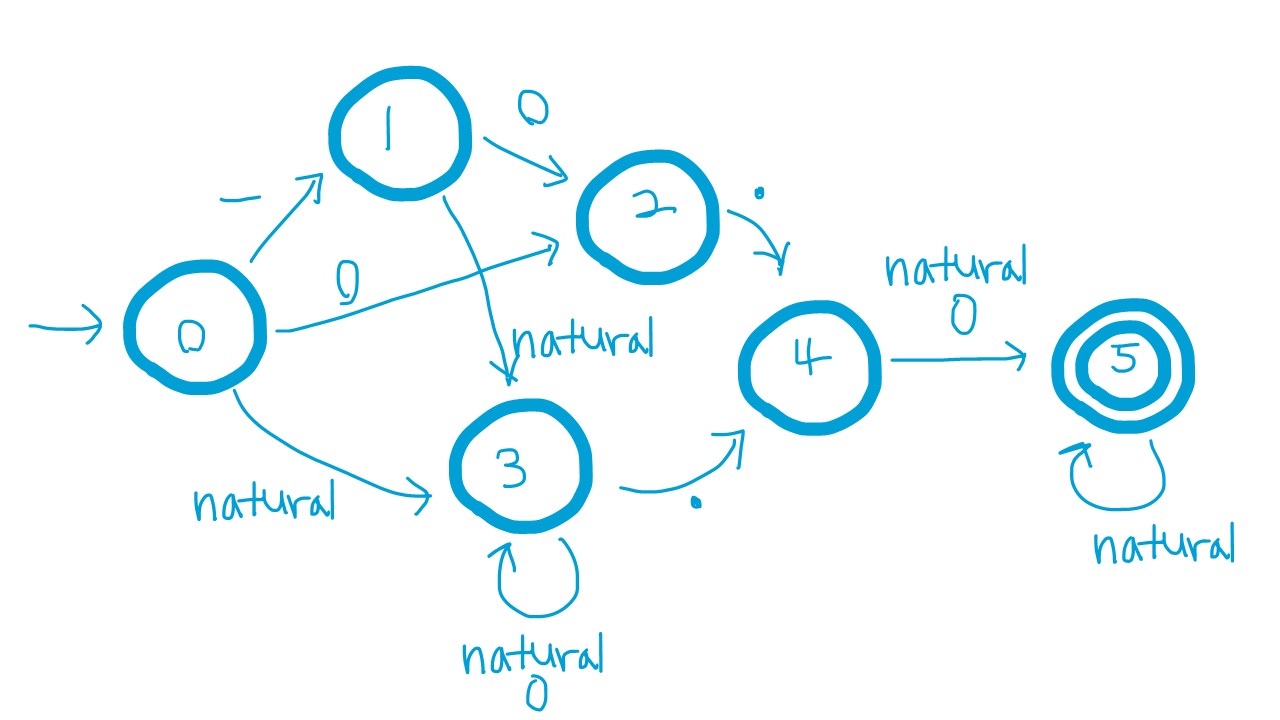
****

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **t** | **r** | **u** | **e** | **f** | **a** | **l** | **s** | **letter** | **0** | **natural** | **\_** |
| **T0** | T1 |  |  |  | T4 |  |  |  |  |  |  |  |
| **T1** |  | T2 |  |  |  |  |  |  |  |  |  |  |
| **T2** |  |  | T3 |  |  |  |  |  |  |  |  |  |
| **T3** |  |  |  | T8 |  |  |  |  |  |  |  |  |
| **T4** |  |  |  |  |  | T5 |  |  |  |  |  |  |
| **T5** |  |  |  |  |  |  | T6 |  |  |  |  |  |
| **T6** |  |  |  |  |  |  |  | T7 |  |  |  |  |
| **T7** |  |  |  | T8 |  |  |  |  |  |  |  |  |
| **T8** |  |  |  |  |  |  |  |  | T9 | T9 | T9 | T9 |
| **T9** |  |  |  |  |  |  |  |  |  |  |  |  |

(빈칸은 모두 Φ)

* 1. **REAL**

REAL = (-|ε) (0|natural(0|natural)\*). (0|(0|natural)\* natural)

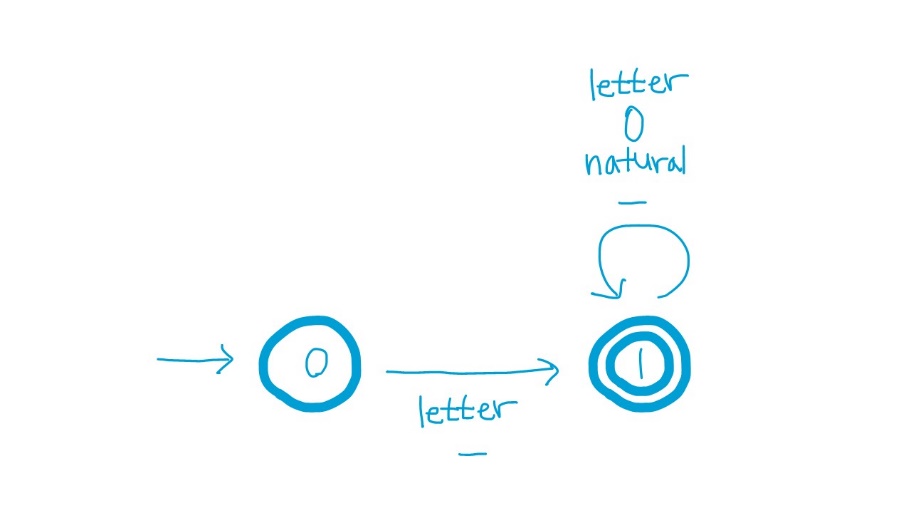


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **-** | **0** | **natural** | **.** |
| **T0** | T1 | T2 | T3 |  |
| **T1** |  | T2 | T3 |  |
| **T2** |  |  |  | T4 |
| **T3** |  | T3 | T3 | T4 |
| **T4** |  | T5 | T5 |  |
| **T5** |  | T6 | T5 |  |
| **T6** |  | T6 | T5 |  |

(빈칸은 모두 Φ)

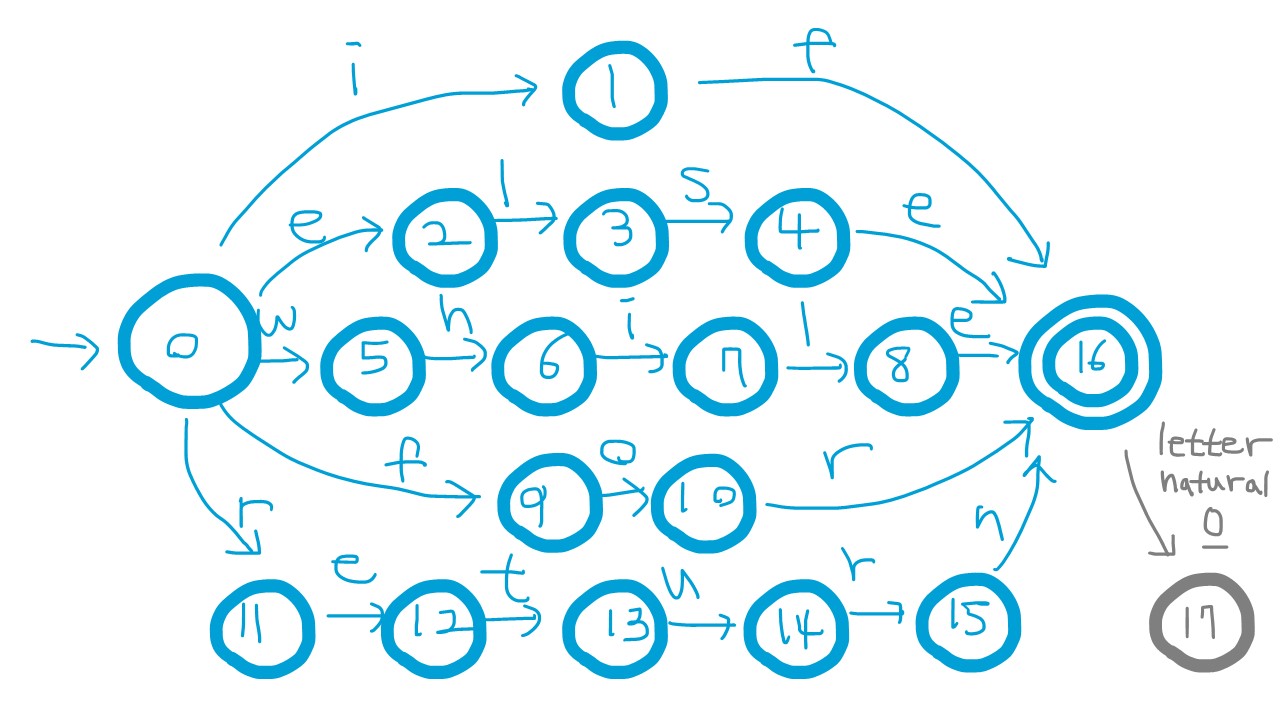
* 1. **ID**

ID = (letter|\_)(letter|0|natural\_)\*

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **letter** | **\_** | **0** | **natural** |
| **T0** | T1 | T1 |  |  |
| **T1** | T1 | T1 | T1 | T1 |

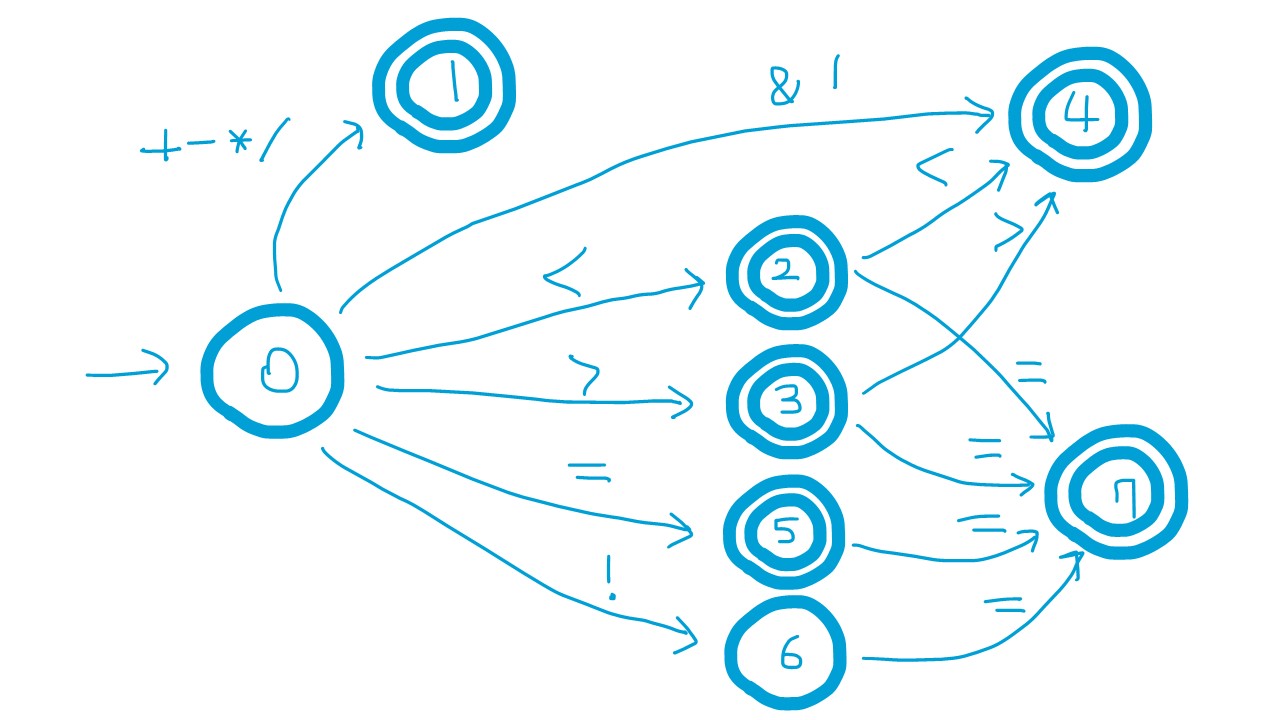
(빈칸은 모두 Φ)

* 1. KEYWORDS: **IF ELSE WHILE FOR RETURN**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **i** | **f** | **e** | **l** | **s** | **w** | **h** | **o** | **r** | **t** | **u** | **n** | **letter** | **0** | **natural** | **\_** |
| **T0** | T1 | T9 | T2 |  |  | T5 |  |  | T11 |  |  |  |  |  |  |  |
| **T1** |  | T16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **T2** |  |  |  | T3 |  |  |  |  |  |  |  |  |  |  |  |  |
| **T3** |  |  |  |  | T4 |  |  |  |  |  |  |  |  |  |  |  |
| **T4** |  |  | T16 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **T5** |  |  |  |  |  |  | T6 |  |  |  |  |  |  |  |  |  |
| **T6** | T7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **T7** |  |  |  | T8 |  |  |  |  |  |  |  |  |  |  |  |  |
| **T8** |  |  | T16 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **T9** |  |  |  |  |  |  |  | T10 |  |  |  |  |  |  |  |  |
| **T10** |  |  |  |  |  |  |  |  | T16 |  |  |  |  |  |  |  |
| **T11** |  |  | T12 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **T12** |  |  |  |  |  |  |  |  |  | T13 |  |  |  |  |  |  |
| **T13** |  |  |  |  |  |  |  |  |  |  | T14 |  |  |  |  |  |
| **T14** |  |  |  |  |  |  |  |  | T15 |  |  |  |  |  |  |  |
| **T15** |  |  |  |  |  |  |  |  |  |  |  | T16 |  |  |  |  |
| **T16** |  |  |  |  |  |  |  |  |  |  |  |  | T17 | T17 | T17 | T17 |
| **T17** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

(빈칸은 모두 Φ)

* 1. OPERATORS: **ARITHM BITWISE COMP ASSIGN**



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **+** | **-** | **\*** | **/** | **&** | **|** | **<** | **>** | **=** | **!** |
| **T0** | T1 | T1 | T1 | T1 | T4 | T4 | T2 | T3 | T5 | T6 |
| **T1** |  |  |  |  |  |  |  |  |  |  |
| **T2** |  |  |  |  |  |  | T4 |  | T7 |  |
| **T3** |  |  |  |  |  |  |  | T4 | T7 |  |
| **T4** |  |  |  |  |  |  |  |  |  |  |
| **T5** |  |  |  |  |  |  |  |  | T7 |  |
| **T6** |  |  |  |  |  |  |  |  | T7 |  |
| **T7** |  |  |  |  |  |  |  |  |  |  |

(빈칸은 모두 Φ)

-각 Final State의 token 이름

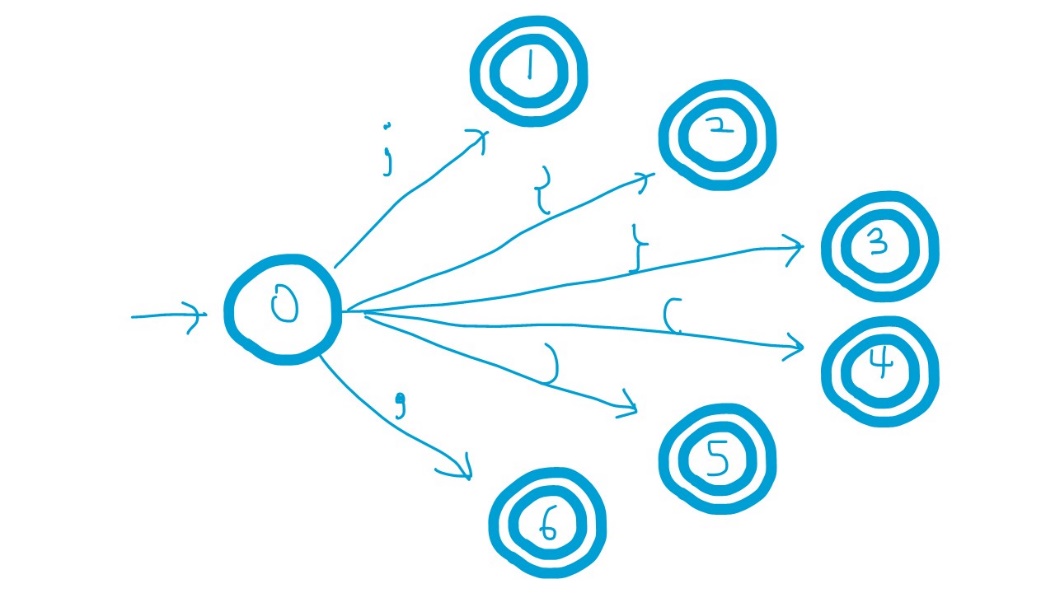
T1: ARITHM

T4: BITWISE

T5: ASSIGN

T2, T3, T7: COMP

* 1. SYMBOLS: **SEMI LBRKT RBRKT LPAREN RPAREN COMMA**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **;** | **{** | **}** | **(** | **)** | **,** |
| **T0** | T1 | T2 | T3 | T4 | T5 | T6 |
| **T1** |  |  |  |  |  |  |
| **T2** |  |  |  |  |  |  |
| **T3** |  |  |  |  |  |  |
| **T4** |  |  |  |  |  |  |
| **T5** |  |  |  |  |  |  |
| **T6** |  |  |  |  |  |  |

(빈칸은 모두 Φ)

-각 Final State의 token 이름

T1: SEMI

T2: LBKRT

T3: RBKRT

T4: LPAREN

T5: RPAREN

T6: COMMA

# 3. How lexical analyzer works for recognizing tokens

* 1. **Overall Procedures**

**-** 입력 파일로부터 한 글자를 읽은 후, 다음과 같은 10가지 함수를 이용해 token을 분류합니다. (이 중 isWHITESPACE만 symbol table을 생성하지 않고 스킵합니다)

**isWHITESPACE**

**isSYMBOL**

**isVTYPE**

**isKEYWORD**

**isBOOLEAN**

**isLITERAL**

**isREAL**

**isINTEGER**

**isOPERATOR**

**isID**

**-** 각 함수는 token을 분류하면 true를 반환하는데, 10개 함수를 모두 통과하고도 true인 적이 없다면 에러를 output파일에 출력하고 중단합니다.

**-**에러가 발생한 경우 기존에 작성하던 symbol table이 있으면 그 파일을 지우고 다시 output파일을 만들어서 에러문구만 출력합니다.

-token은 아니지만 하나의 regular expression인 letter와 natural(자연수)도 각각 isletter() isnatural() 이라는 참 거짓 여부를 반환하는 함수를 만들었습니다.

* 1. **Implementation Details**

Algorithms

**-** 10개의 is~함수 중에서 주의할 우선순위는 다음과 같습니다:

**isREAL > isINTEGER**

(예: 51.6 => isINTEGER를 먼저 하면 51을 integer로 분류하고 .에서 에러가 발생합니다)

**isINTEGER > isOPERATOR**

(예: -1 => isOPERATOR를 먼저 하면 -를 operator로 분류하고 1을 integer로 분류하는 문제가 발생합니다)

**isVTYPE > isID**

(예: int => isID를 먼저 하면 int를 identifier로 분류하는 문제가 발생합니다)

**isKEYWORD > isID**

(예: if => isID를 먼저 하면 if를 identifier로 분류하는 문제가 발생합니다)

**isBOOLEAN > isID**

(예: true => isID를 먼저 하면 true를 identifier로 분류하는 문제가 발생합니다)

interesting과 같이 처음은 vtype(혹은 keyword, Boolean)에 분류될 수 있는데 뒤에 글자가 더 있는 경우 즉, **isID > isVTYPE** 와 같은 우선순위를 띄어야 하는 경우 id로 분류하기 위해, DFA table에서 회색 처리된 부분과 같이 non-final state를 하나씩 추가함으로써 이 문제를 해결했습니다.

-각 10개의 is~함수는 거의 동일한 형식으로 구성되어 있습니다.

-위의 DFA Table에 맞게 state가 바뀌도록 IF-ELSE 구문들이 적혀 있습니다. 더 이상 input에 따라 바꿀 수 있는 State가 없는 경우가 될 때까지 이것을 반복합니다.

-마지막에 도달한 State가 Final State라면 classifier 함수를, 아니면 rewinder 함수를 작동하게 됩니다.

-classifier 함수는 output파일에 symbol table의 한 행을 작성합니다. rewinder 함수는 token 분류가 안된 마지막 위치로 되돌아갑니다. Read를 할 때 ngets라는 변수의 값을 1씩 증가시키는데, rewinder 함수는 ngets만큼 파일 포인터 위치를 되감습니다.

-isWHITESPACE 함수는 input이 ‘\n’인 걸 판단할 때 변수 linenumber를 1 증가시켜 에러가 났을 때 몇 번째 줄에서 에러가 난 건지 알 수 있습니다.

Data Structures

- 전역변수: char **attribute[]**

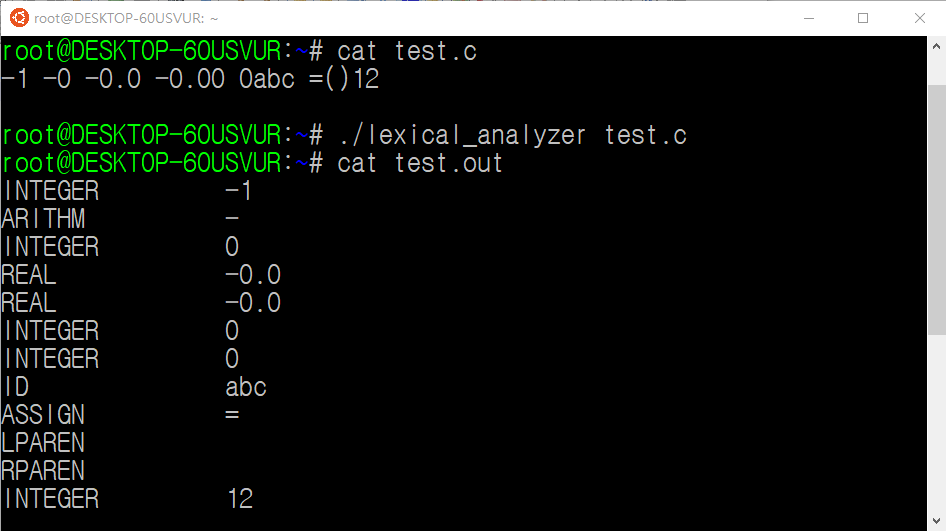
글자를 읽을 때 이 배열에 저장해두어 나중에 classifier 함수에서 symbol table의 attribute부분을 적을 때 가져오게끔 구현하였습니다. 이 배열은 한 번만 선언해서 재사용합니다.

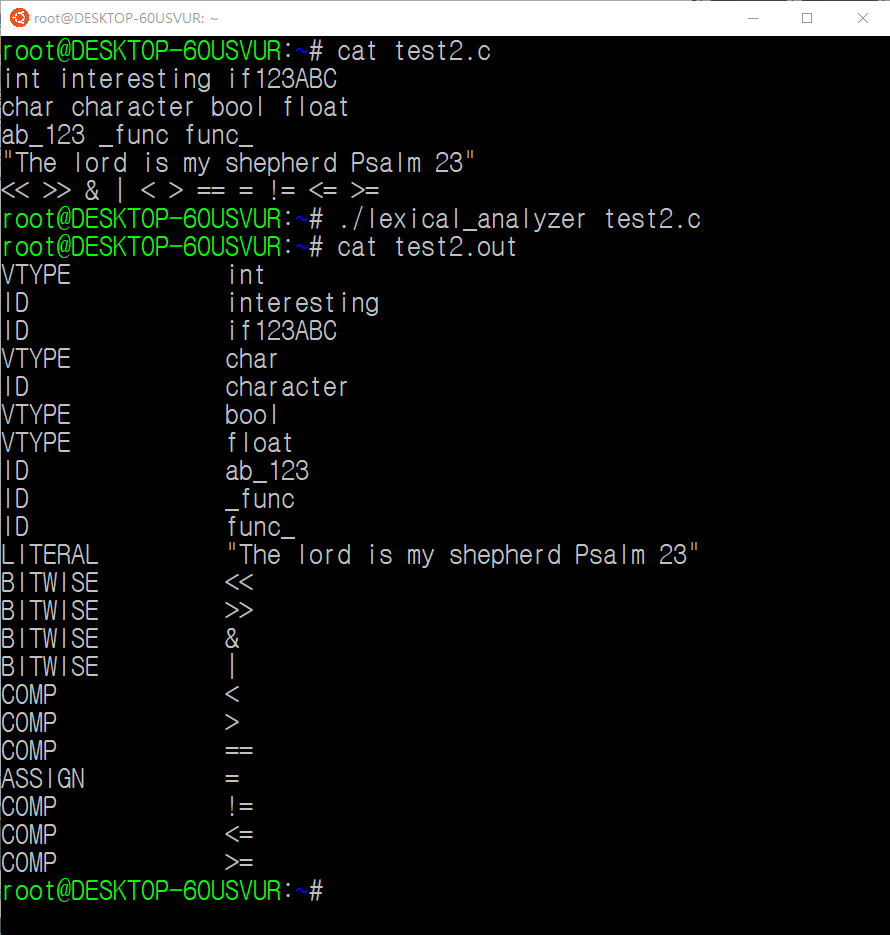
키워드로 분류된 경우 symbol table의 오른쪽 부분은 비우고, 배열 attribute[]의 내용을 대문자화하여 symbol table의 왼쪽에 출력합니다.

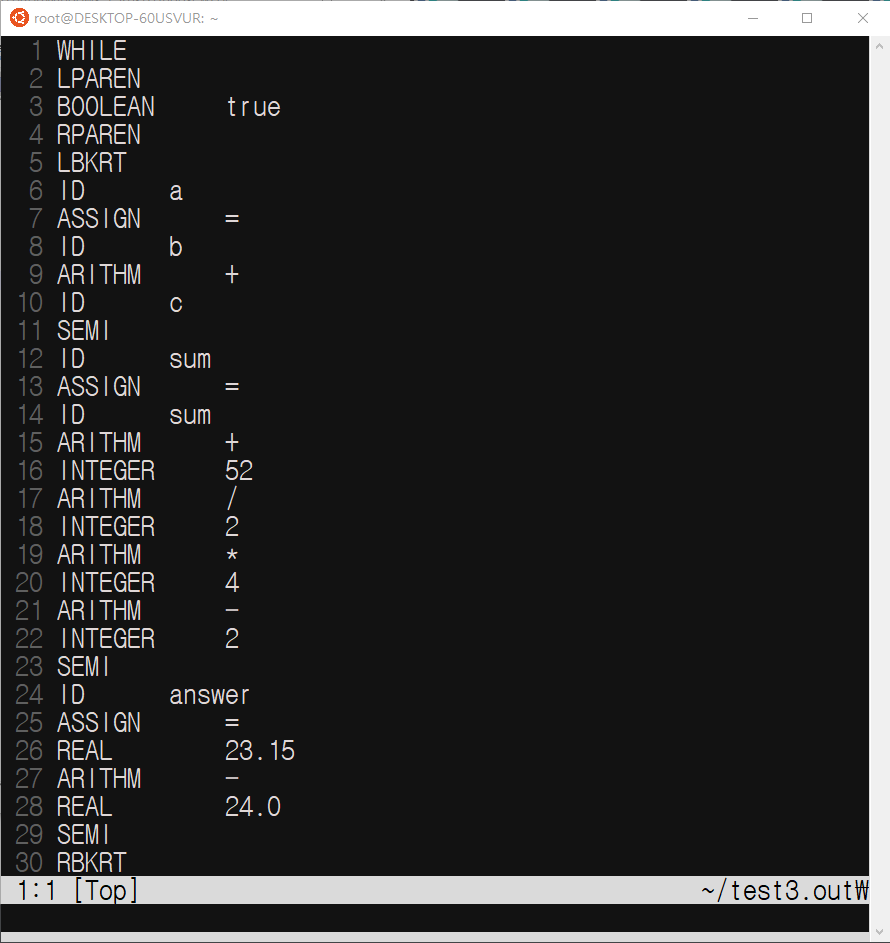
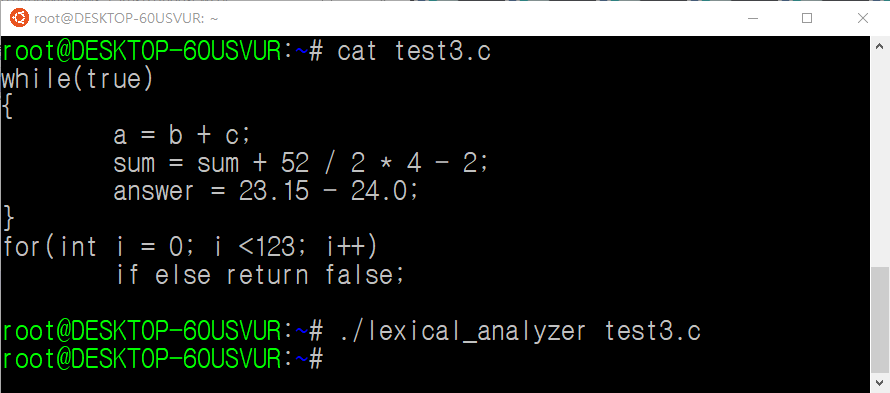
- (10개의 is~함수에서 각각 쓰는) 지역변수: int **state**

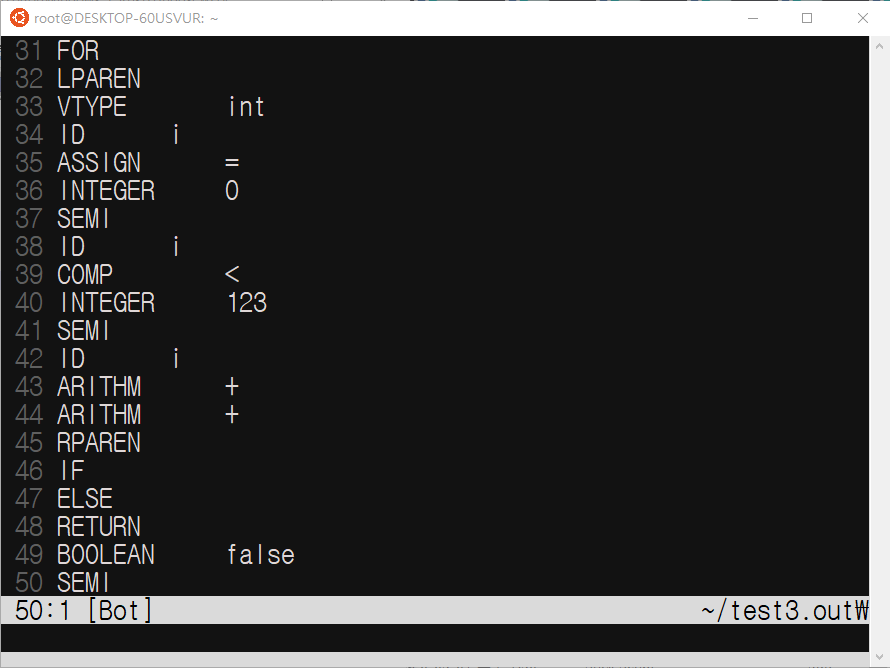
현재 어떤 state에 있는지는 int 형식으로 저장합니다.

* 1. **Working Examples**



****

<= test3.out vim 화면(상)

<= test3.out vim 화면(하)

에러 발생 예시

