PL Assignment #3 : Cute14 Scanner

과제물 부과일 : 2015-03-23 (월)
Program Upload 마감일 : 2015-03-29(월) 23:59:59

문제

Cute14 문법에 따라 작성된 program이 as03.txt에 저장되어 있다. 이를 input file로 하여, 모든 token을 인식하여 token과 lexeme을 모두 출력하는 program을 작성하시오.
예를 들어, Cute14으로 작성된 program이 아래와 같을 경우,

```
( define length
  ( lambda ( x )
    ( cond ( ( null? X ) 0 )
          ( #T ( + 1 ( length ( cdr x ) ) ) ) ) ) )
```

(주의: token과 token 사이에 반드시 공백이 있다.)

출력은 아래와 같아야 한다.

```
L_PAREN         (
DEFINE          define
ID              length
L_PAREN         (
LAMBDA          lambda
L_PAREN         (
ID              x
R_PAREN         )
L_PAREN         (
COND            cond
L_PAREN         (
L_PAREN         (
NULL_Q          null?
ID              x
R_PAREN         )
INT             0
R_PAREN         )
INT             0
R_PAREN         )
L_PAREN         (
TRUE            #T
L_PAREN         (
PLUS            +
INT             1
L_PAREN         (
ID              length
L_PAREN         (
CDR             cdr
R_PAREN         )
R_PAREN         )
R_PAREN         )
R_PAREN         )
R_PAREN         )
R_PAREN         )
R_PAREN         )
```

Programming 순서

1. Regular expression 작성
2. DFA 작성
3. Program 작성

Regular Expression

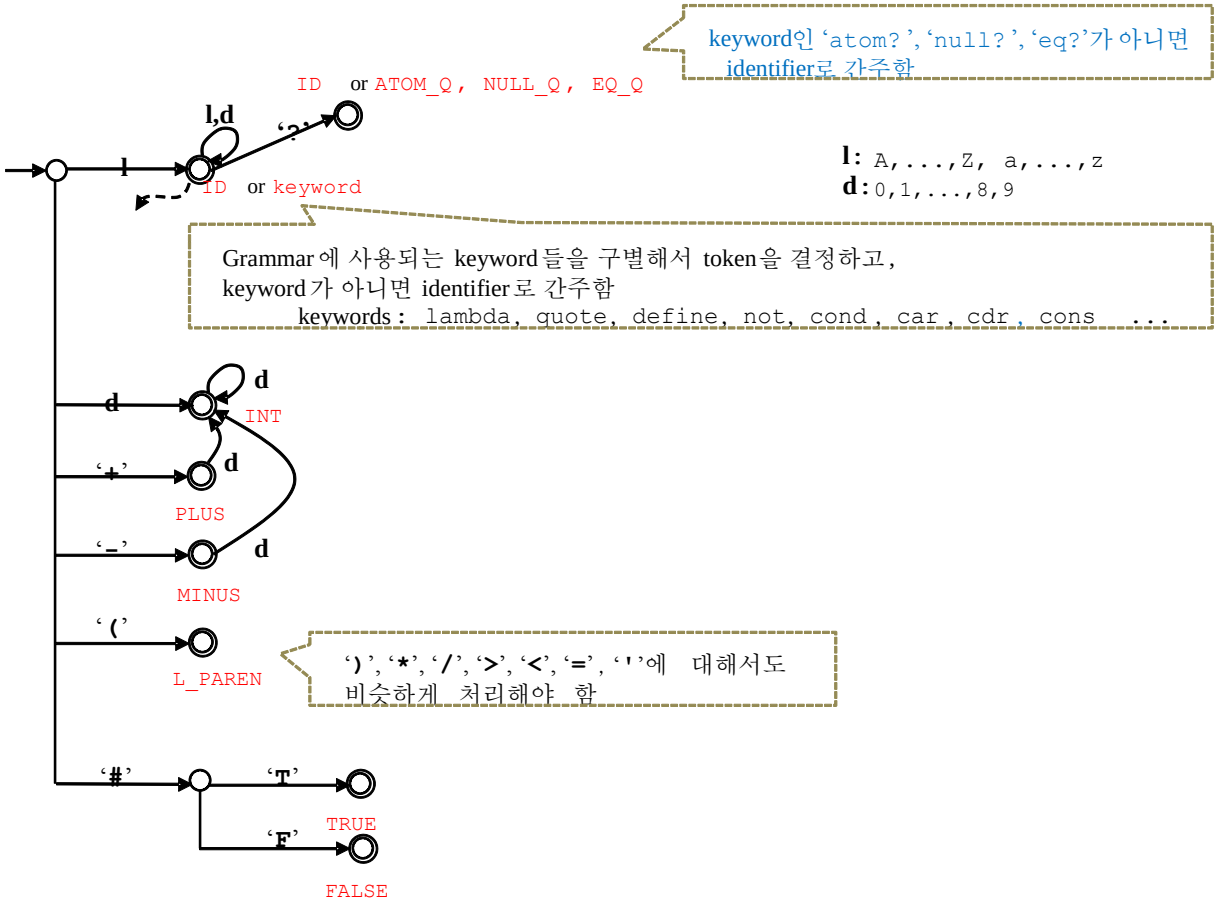
```
QUESTION:  ID '?'
ID:        Alpha[Alpha|Digit]*
INT:       Digit+ | PLUS Digit+ | MINUS Digit+

L_PAREN:   '('
R_PAREN:   ')'
PLUS:      '+'
MINUS:     '-'
TIMES:     '*'
DIV:       '/'
LT:        '<'
EQ:        '='
GT:        '>'
APOSTROPHE: '\''
TRUE:      Sharp 'T'
FALSE:     Sharp 'F'

Sharp:     '#'
Alpha:     [A-Z] | [a-z]
Digit:     [0-9]
```

- ID나 QUESTION중에서 특별한 의미를 가지는 keyword와 해당 token 이름은 다음과 같다.(keyword가 아니면 ID로 간주)

"define"	DEFINE
"lambda"	LAMBDA
"cond"	COND
"quote"	QUOTE
"not"	NOT
"cdr"	CAR
"car"	CDR
"cons"	CONS
"eq?"	EQ_Q
"null?"	NULL_Q
"atom?"	ATOM_Q

mDFA

오류가 있는 input은 없다고 가정, 모든 토큰 사이에는 공백이 있다고 가정



작성해야 할 코드

1) 괄호 안에 TokenType 에 맞는 state number 입력
2) 키워드 작성
3) TransitionMatrix 초기화
4) init_TM() 작성
5) nextToken()에서 에러 처리하는 코드 작성

Program 작성

Token 표현

```
	public enum TokenType{
		//1) 괄호 안에 TokenType에 맞는 state number 입력
		INT(…), ID(…), MINUS(…), PLUS(…),
		L_PAREN(…), R_PAREN(…), TRUE(…), FALSE(…),
		TIMES(…), DIV(…), LT(…), GT(…),  EQ(…),
		APOSTROPHE(…),

		DEFINE(…),   LAMBDA(…),   COND(…), QUOTE(..),
		NOT(…), CAR(…), CDR(…), CONS(…),

		ATOM_Q(…), NULL_Q(…), EQ_Q(…);

		private final int finalState;

		TokenType(int finalState) {
			this.finalState = finalState;
		}

		public static TokenType keyWordCheck(String str){
			return KeyWord.m.get(str); //if not keyword return null
		}

		private enum KeyWord{
			DEFINE("define", TokenType.DEFINE),
			LAMBDA("lambda", TokenType.LAMBDA),
			//2) 위와 같은 방법으로 나머지 키워드 작성
			…
			…

			final String word;
			final TokenType type;

			KeyWord(String word, TokenType type){
				this.word = word;
				this.type = type;
			}

			private static final Map<String, TokenType> m = new
HashMap<String, TokenType>();
			static{
				for(KeyWord k : KeyWord.values()){
					m.put(k.word, k.type);
				}
			}
		}
	}
```

Data Type

```
public static class Token {
		public final TokenType type;
		public final String lexme;

		Token(TokenType type, String lexme) {
			this.type = type;
			this.lexme = lexme;
		}

		@Override
		public String toString() {
			return String.format("[%s: %s]", type.toString(), lexme);
```

```
        }
    }
```

Programming-TM

```
    private int transM[][];
    private boolean accept[];
    private String source;
    private StringTokenizer st;
    public Scanner(String source) {
        //3) state 개 수 만큼 배열 초기화
        this.transM = new int[?][128];
        this.source = source == null ? "" : source;
        st = new StringTokenizer(source, " ");
        init_TM();
    }


    private void init_TM() {
        for (int i = 0; i < transM.length; i++)
            for (int j = 0; j < transM[i].length; j++)
                transM[i][j] = -1;

        int i;
        for (i = '0'; i < '9'; i++) // digit
        {
            transM[0][i] = 1;
            transM[1][i] = 1; // digit
            transM[2][i] = 1; // '-'
            transM[3][i] = 1; // '+'
            transM[4][i] = 4; // id
        }

        for (i = 'a'; i < 'z'; i++) // alpha
        {
            transM[0][i] = 4;
            transM[4][i] = 4;
        }
        for (i = 'A'; i < 'Z'; i++) {
            transM[0][i] = 4;
            transM[4][i] = 4;
        }
        //4) '(', ')', '+', '-', '*', '/', '<', '=', '>', '\'', '#T', '#F', #에 대해서 작동하도록
    작성
    }


    private Token nextToken(){
        int StateOld = 0, StateNew;
        if (!st.hasMoreTokens())
            return null;
        // 그 다음 토큰을 받음
        String temp = st.nextToken();
        Token result = null;

        for (int i = 0; i < temp.length(); i++) {
            StateNew = transM[StateOld][temp.charAt(i)];
            // 입력문자로 새로운 상태 판별
            if(...){
                //5) 에러에 해당하는 조건을 만들고 그에  해당하는 코드 작성

            }
            StateOld = StateNew;
```

```
                }

                for (TokenType t : TokenType.values()) {
                        if (t.finalState == StateOld) {
                                TokenType keyWord =
TokenType.keyWordCheck(temp);
                                if (keyWord != null)
                                        result = new Token(keyWord, temp);
                                else result = new Token(t, temp);
                                break;
                        }
                }
                return result;
}

        public static void main(String[] args){
                //txt file to String

                String source = …
                Scanner s = new Scanner(source);
                List<Token> tokens = s.tokenize();

                //print
                …
        }
```

유의 사항

- 입력 data는 프로그램을 제대로 검증할 수 있는 data로 구성되어야 한다.
- 반드시 transition matrix를 사용하여야 한다.
- 5)을 반드시 작성한다.

참고자료 – Java API를 이용하여 작성한 Scanner

다음은 Java에서 제공하는 패키지 java.util.regeX로 작성한 Scanner의 nextToken()이다. 정규표현을 input으로 받아 패턴 매칭해주는 기능이 있다.

```
private Token nextToken() {
        // 토큰이 더 있는지 검사
        if (!st.hasMoreTokens())
                return null;
        // 그 다음 토큰을 받음
        String temp = st.nextToken();
        Token result = null;

        if (Pattern.matches("-?+[1-9]+[0-9]*", temp))
                result = new Token(TokenType.INT, temp);
        else if (Pattern.matches("[a-zA-Z]+[0-9]*[?]?+", temp)) {
                TokenType keyWord = TokenType.keyWordCheck(temp);
                if (keyWord != null)
                        result = new Token(keyWord, temp);
                else result = new Token(TokenType.ID, temp);
        } else if (temp.equals(")"))
                result = new Token(TokenType.R_PAREN, temp);
        else if (temp.equals("("))
                result = new Token(TokenType.L_PAREN, temp);
        else if (temp.equals("+"))
                result = new Token(TokenType.PLUS, temp);
        else if (temp.equals("-"))
                result = new Token(TokenType.MINUS, temp);
        else if (temp.equals("*"))
                result = new Token(TokenType.TIMES, temp);
        else if (temp.equals("/"))
                result = new Token(TokenType.DIV, temp);
        else if (temp.equals(">"))
                result = new Token(TokenType.GT, temp);
        else if (temp.equals("\'"))
                result = new Token(TokenType.APOSTROPHE, temp);
        else if (temp.equals("#T"))
                result = new Token(TokenType.TRUE, temp);
        else if (temp.equals("#F"))
                result = new Token(TokenType.FALSE, temp);

        return result;
}
```

밑의 링크에서 regeX를 검색하면 더 자세한 내용을 볼 수 있다.

http://docs.oracle.com/javase/7/docs/api/

수정: 2015-03-19