PL Assignment #4 : Cute19 Scanner

과제물 부과일 : 2019-04-03 (수)
Program Upload 마감일 : 2019-04-18(목) 23:59:59

문제

Cute19 문법에 따라 작성된 program이 as04.txt에 저장되어 있다. 이를 input file로 하여, 모든 token을 인식하여 token과 lexeme을 모두 출력하여 파일(hw04.txt)에 저장하는 program을 작성하시오.

예를 들어, Cute19으로 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         )
L_PAREN         (
TRUE            #T
L_PAREN         (
PLUS            +
INT             1
L_PAREN         (
ID              length
L_PAREN         (
CDR             cdr
ID              x
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

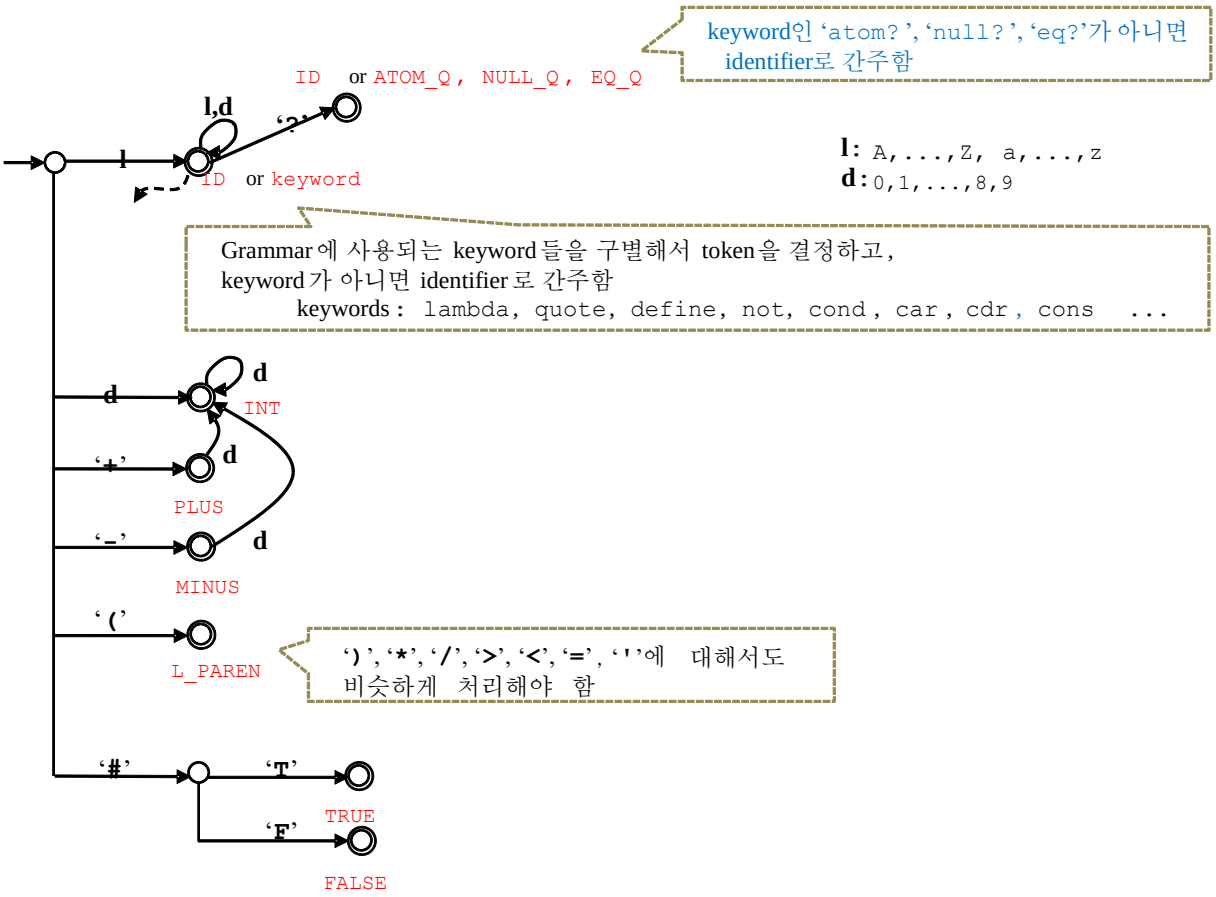
```
QUESTION:  ID '?'
ID:        Alpha[Alpha|Digit]*
INT:       Digit+ | PLUS Digit+ | MINUS Digit+

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) TokenType 클래스의 switch 문 작성

2) Char 클래스의 getType 함수 if 문 수정

3) State 클래스 START 상태에서 특수 문자의 경우 추가

Program 작성

TokenType 클래스

```
public enum TokenType {
	INT,
	ID,
	TRUE, FALSE, NOT,
	PLUS, MINUS, TIMES, DIV,   //special chracter
	LT, GT, EQ, APOSTROPHE,    //special chracter
	L_PAREN, R_PAREN, QUESTION, //special chracter
	DEFINE, LAMBDA, COND, QUOTE,
	CAR, CDR, CONS,
	ATOM_Q, NULL_Q, EQ_Q;

	static TokenType fromSpecialCharactor(char ch) {
		switch ( ch ) {
			case '+':
				return PLUS;
			//나머지 Special Character에 대해 토큰을 반환하도록 작성
			default:
				throw new IllegalArgumentException("unregistered
char: " + ch);
		}
	}
}
```

Char 클래스

```
class Char {
	private final char value;
	private final CharacterType type;

	enum CharacterType {
		LETTER, DIGIT, SPECIAL_CHAR, WS, END_OF_STREAM,
	}

	…….
	private static CharacterType getType(char ch) {
		int code = (int)ch;
		if ( ) { //letter가 되는 조건식을 알맞게 채우기
			return CharacterType.LETTER;
		}

		if ( Character.isDigit(ch) ) {
			return CharacterType.DIGIT;
		}

		switch ( ch ) {
			case '-': case '+': case '*': case '/':
			case '(': case ')':
			case '<': case '=': case '>':
			case '#': case '\'':
				return CharacterType.SPECIAL_CHAR;
		}

		if ( Character.isWhitespace(ch) ) {
			return CharacterType.WS;
		}

		throw new IllegalArgumentException("input=" + ch);
	}
}
```

State 클래스

```
enum State {
	START {
		@Override
		public TransitionOutput transit(ScanContext context) {
			Char ch = context.getCharStream().nextChar();
			char v = ch.value();
			switch ( ch.type() ) {
				case LETTER:
					context.append(v);
					return GOTO_ACCEPT_ID;
				case DIGIT:
					context.append(v);
					return GOTO_ACCEPT_INT;
				case SPECIAL_CHAR: //special character가 들어온 경우
					if ( ) { //부호인 경우에 상태 반환

					}
					else if (  ) {  //boolean인 경우에 상태 반환

					}
					else { //그 외에는 type을 알아내서 알맞은 상태로
반환

					}
				case WS:
					return GOTO_START;
				case END_OF_STREAM:
					return GOTO_EOS;
				default:
					throw new AssertionError();
			}
		}
	},
	……
}
```

유의 사항

- 입력 data는 프로그램을 제대로 검증할 수 있는 data로 구성되어야 함
- Input file은 직접 만들어 사용함
- 이번 과제 진행과 이후 과제 진행을 위해 전체 코드를 자세히 읽어보는 것을 추천함
- 주어진 코드는 lexer pakage안에 위치하게 프로젝트 구성함
- 출력파일이름이 다르면 채점 시 불이익 있을 수 있으므로 유의 바람
- 오류가 없는 입력만 들어온다고 가정함

수정: 2019-04-03