Programming Project #1: Mini-C 언어의 Lexical Analyzer 작성

[문제]

Mini-C는 본 과제를 위해 정의한 언어이다. **Mini-C 프로그램 파일**을 입력받아 어휘를 분석하는 lexical analyzer를 flex를 이용하여 작성하시오.

[Mini-C 언어의 문법]

- 현 단계에서는 문법 자체를 자세하게 이해하지 못해도 되나 문법에서 토큰을 파악하는 것이 중요함

```
program    -> fun_list block
fun_list    -> fun_def fun_list
             |  ε
fun_def   -> type ID ( arg_list )
              { decl_list stmt_list }
arg_list -> arg , arg_list
arg_list -> arg
arg -> type variable
decl_list -> decl decl_list
decl_list -> ε
decl -> type var_list ;
type -> int | double | str
stmt_list  -> stmt stmt_list
            | stmt
stmt  -> expr ;
          | print_stmt ;
          | return_stmt ;
          | control_stmt
          | block
print_stmt -> print(expr_list)
return_stmt -> return exp
control_stmt-> if_stmt
              | while_stmt
if_stmt   -> if ( expr ) stmt else stmt
            | if ( expr ) stmt
while_stmt -> | while ( expr ) stmt
block -> { stmt_list }
var_list -> variable , var_list
            | variable
expr     -> value
```

```
            | variable
            | variable = expr
            | expr + expr
            | expr - expr
            | expr * expr
            | expr / expr
            | expr > expr
            | expr >= expr
            | expr < expr
            | expr <= expr
            | expr == expr
            | expr != expr
            | - expr
            | ( expr )
            | ID( expr_list )
expr_list ->  expr , expr_list
            | expr
value -> INTEGER | DOUBLE | STRING
variable -> ID
```

[Mini-C 언어의 어휘]

Keywords

int double str if while return

Identifiers (ID)

- 영문자 대소문자, 숫자, underscore(_)로만 이루어짐
- 첫 글자는 반드시 대소문자, _만 가능
- '_'문자만으로 이루어질 수는 없음
- 길이는 제한이 없으나 실제 구분은 첫 16자로 함

INTEGER

- 기본적으로 C 언어의 int 표기법을 따름 (십진수만 사용)
- 자리 수는 제한 없으나 값을 저장 시 최대 10자리만 저장(11자리 이상은 상위부분 절단)

DOUBLE

- C 언어의 double constant 표기법을 따름

STRING

- 기본적으로 C 언어의 표기법을 따름

Operators

```
+  -  *  /
=
> >= < <= == !=
```

기타 특수 문자들

" , ( ) ; { }

Comments

C언어의 주석을 따른다 (/* */ 와 //)

[입력 예시]

```
int f ( int a, double b)
{
   int sum;
   sum = a + b;
   print(a, b, sum);
}
```

※ 문장구조는 체크하지 않으니 실제 문법과 달라도 상관없음
가령 아래와 같이 입력해도 됨

```
int sum ; sum print ( ;
```

[출력 형식]

1. 토큰들의 리스트 (token 옆에 lexeme을 함께 출력할 것)

TOKEN	LEXEME
<INT, >	int
<ID, 1>	f
<LPAREN, >	(
<INT, >	int
<ID, 2>	a
<COMMA, >	,
...	
<PRINT, >	print
<LPAREN, >	(
<ID, 2>	a
<COMMA, >	,
<ID, 3>	b
<COMMA, >	,
...	

※ 정수상수 100은 <INTEGER, 100> 과 같은 TOKEN으로 표기된다.

2. 심볼테이블 및 스트링 테이블 (출력 형식은 달라도 됨)
   - 정수값이나 실수값은 token value란에 직접 표현

index	symbols
1	"f"
2	"a"
3	"b"
4	

index	strings
1	
2	
3	
4	

[보고서 구조]

1. 서론
   - 개요
   - 구현된 부분과 구현되지 않은 부분을 명확하게 명시할 것.
2. 문제 분석
   - token의 종류
   - transition diagram
   - 각 토큰을 위한 regular expression
3. 설계
   - 중요 자료구조 및 프로그램 모듈구조 및 모듈설명
4. 입력 데이터(Mini-C 프로그램) 2개 이상(**파일명 : ex1.txt, ex2.txt**)
   - 한개는 정상적인 프로그램
   - 다른 하나는 오류가 포함된 프로그램
   - 가능하면 모든 토큰의 종류가 포함되도록 함
5. 결과데이터(토큰리스트, 심볼테이블, 스트링테이블)
6. 첨부자료: 소스 프로그램
   - source program (lex에 의해 생성된 코드는 포함시키지 않는다)

[제출방법 및 제출일]

- "hw1_학번" 디렉토리를 만들어 코드와 보고서를 넣는다. 단, lex 컴파일러를 수행하여 나온 코드(lex.yy.c)는 넣지 않는다.
- 아주Bb 과제게시판에 "hw1_학번" 디렉토리 전체를 압축하여 올릴 것. (파일명: hw1_학번.zip)
- 제출일: 2019년 4월 1일(월) 자정 (보고서 출력본: 4월 2일 수업시간에 제출)
- **주의: 제출기한을 하루 초과시 5%감점. 제출기한 2일을 초과하는 경우 0점 처리**