프로그래밍 언어
LEX 실습2

Lex 구조

1. 정의절
2. 규칙절
3. 서브루틴절

```
%{
/* 정의절(definition section)*/
%}
%%

        /* 규칙절(rule section) */
%%

/* 사용자 서브루틴절(user subroutine section) */
int main(){
        yylex();
        return 0;
}

int yywrap(){
        return 1;
}
```

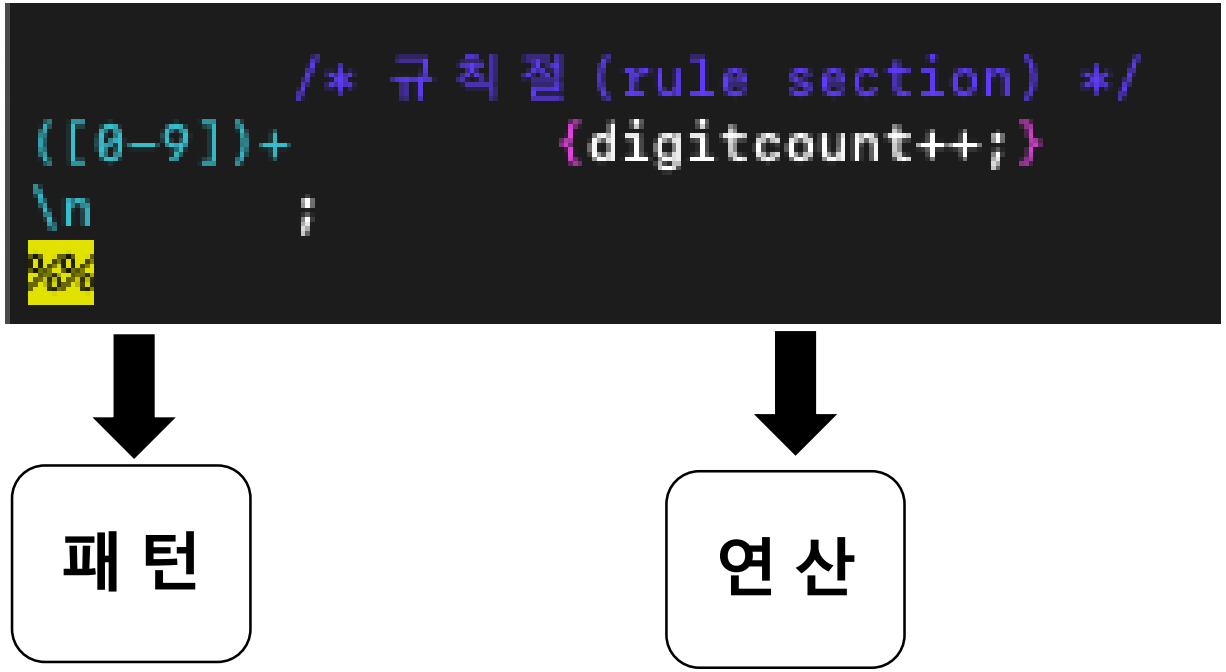
Lex 구조 - 정의절

- %{
  C 코드 선언 부
  %}
- 패턴을 단순화 하기 위한 변수 선언이 가능하다.
  Ex) DIGIT, WORD

```
%{
#include <stdio.h>
int digitcount=0;
%}

DIGIT    [0-9]
WORD     [a-zA-Z]+
%%

{DIGIT}+  {digitcount++;printf("%s\n",yytext);}
.|\n      ;
%%

int main(){
    yylex();
    return 0;
}

int yywrap(){
    return 1;
}
```

Lex 구조 - 규칙절

- 입력된 문자에서 매칭되는 문자열의 패턴과 패턴이 나타났을 때 해당하는 동작으로 구성.



Lex 구조 - 규칙절

- 정규표현식
  - 먼저 작성된 규칙이 먼저 선택됨
    즉 더 명시적으로 작성된 규칙이 상위에 있어야 한다.
  - 정의절에서 선언한 변수 사용 가능
  - Lex 컴파일 시에 오류 검사
- 연산 부분
  - C코드 사용 가능
  - 명시적인 처리가 없으면 default로 echo를 반환한다

Lex 구조 - 규칙절

- 정규표현식

```
M*        : 0회 이상 반복
M+        : 1회 이상 반복
M?        : 0회 또는 1회
M{2,5}    : M이 2번 ~ 5번 반복
M{2, }    : M이 두번이상 반복
M{4}      : M이 정확히 4번 반복
(M)       : M, 우선순위를 위해 사용
{기호}    : 정의절에서 정의된 규칙
[a-z]     : a|b|...|z
[^A-Z]    : A-Z가 아닌 문자
M$        : M다음 문자가 개행일 경우
^M        : M이 패턴의 시작일 경우
.         : \n을 제외한 1byte 문자
<<EOF>>   : 입력의 종료를 의미
```

- 연산부

```
C 코드 사용 가능
default값은 ECHO
yytext: 정규표현식으로 인식된 문자열
    (3p의 경우 DIGIT)
```

Q & A

- Lex 과제 1, 2 질문, 유의사항
  - 테스트케이스 입력 시 출력은 강의록과 완전 동일하게 결과가 나와야 함
  - 1번은 추가적인 채점 테스트 케이스 X
  - 2번은 문자열 전체가 danger일 때 danger 출력
  - 알려주지 않은 내용을 추가로 학습해서 사용해도 문제 없음
- 질문 : pemta818@gmail.com
  - 메일제목 앞에 [Lex] 추가