프로그래밍 언어
LEX 실습

Part 1.

Lex란?

1.1 역사

History

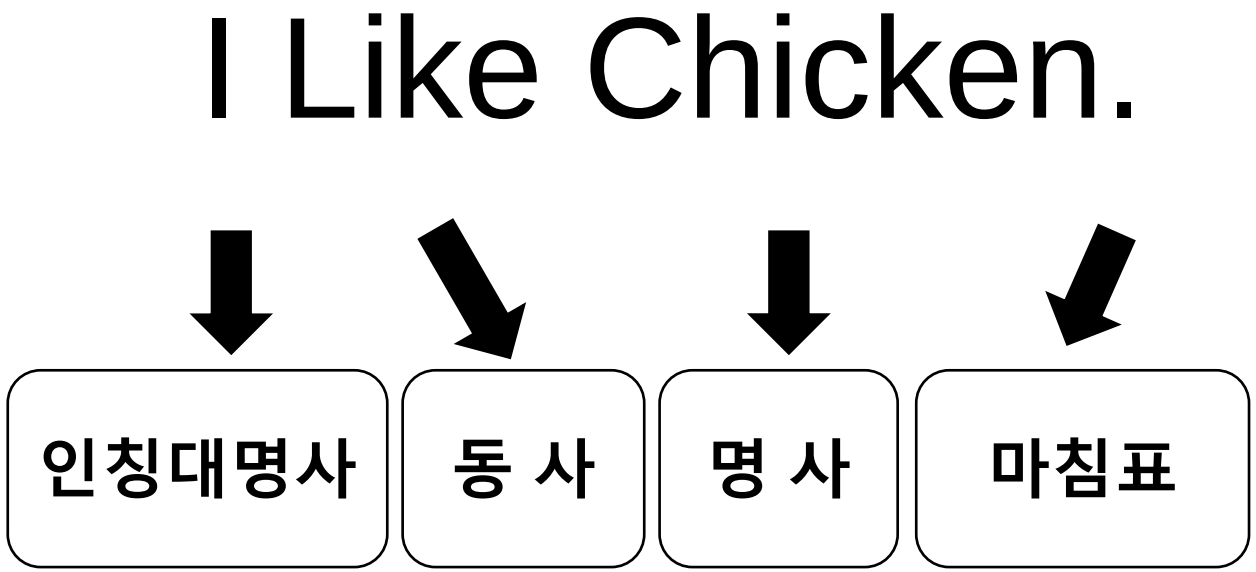
- Lexical Analyzer Generator
- 1975년 AT&T Bell 연구소에서 Eric Schmidt와 Mark Lesk UNIX 시스템의 유틸리티로 개발한 소프트웨어
- Yacc parser generator와 같이 사용

1.2 Scanner

Scanner

- 일정한 구조를 가진 입력을 의미 있는 단위(Unit)으로 분해하여 관련성을 파악 할 수 있게 도와주는 프로그램
- 입력을 토큰이라는 단위로 나누고, 식별할 수 잇는 루틴을 생성한다. 이때 루틴을 Scanner라고 한다.
- Scanner : Text의 어휘 패턴을 인식하는 프로그램

1.3 어휘 분석기



Part 2.

Lex 구조

2.1 Lex 구조

Structure

- 정의절
- 규칙절
- 서브루틴절

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

        /* 규칙절 (rule section) */
%%

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

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

2.1 Lex 구조 - 정의절

Definition Section

- 최종 프로그램에 포함하고자 하는 C 프로그램의 내용을 삽입.
- 출력을 위한 header와 분석한 코드의 카운트를 담당하는 변수가 포함됨.

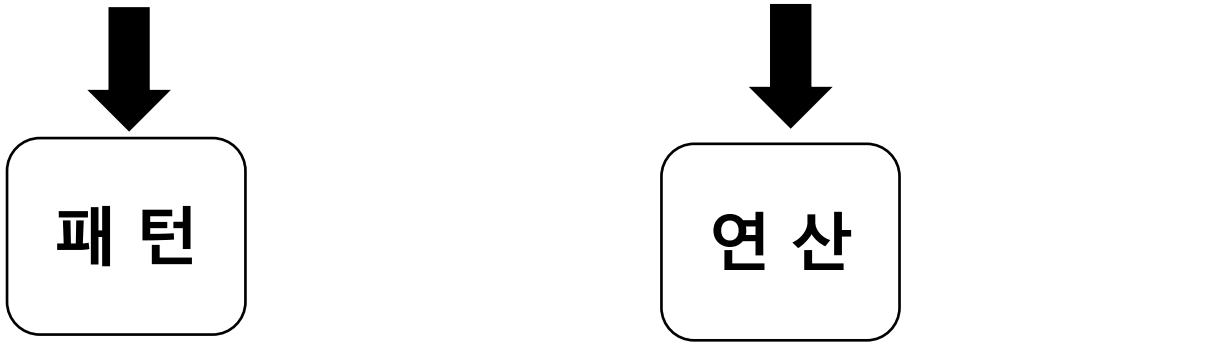
```
%{
/* 정의절 (definition section)*/
#include <stdio.h>
int digitcount = 0;
%}
%%
```

2.2 Lex 구조 - 규칙절

Rules Section

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

```
        /* 규칙절 (rule section) */
([0-9])+        {digitcount++;}
\n      ;
%%
```



2.2 Lex 구조 - 규칙절

Rules Section

- 정규표현식
  - 특정한 규칙을 가진 문자열의 집합을 표현하는데 사용하는 형식 언어.
  - 먼저 작성된 규칙이 먼저 선택됨

```
M* : 0회 이상 반복
M+ : 1회 이상 반복
M? : 선택적, 0 또는 1회
[a-z] : a|b|...|z
[a-zA-Z] : a|b|...|z|A|...|Z
[0-9] : 0|1|...|9
. : 앞에 선언된 문자들을 제외한 모든 문자
```

2.3 Lex 구조 – 사용자 서브루틴절

User Subroutine Section

- 사용자가 서브루틴을 정의하는 부분
- 렉서(C 파일)에 그대로 복사

```c
/* 사용자 서브루틴절 (user subroutine section) */
int main(){
        yylex();
        printf("digitcount : %d\n",digitcount);
        return 0;
}

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

2.3 Lex 구조 – 사용자 서브루틴절

User Subroutine Section

- Lex 함수

```
yylex()      : 스캐너 함수
yywrap()     : yylex()가 EOF를 만나면 호출
yymore()     : 인식된 토큰의 문자열을 yytext 뒤에 첨가
yyless(n)    : n 문자만 yytext에 남기고 나머지는 입력으로 되돌린다
input()      : 다음 문자 반환
output(c)    : 문자 c를 출력으로
unput(c)     : 문자 c를 입력으로
```

2.4 예시

Example

```
%{
/* 정의절 (definition section)*/
#include <stdio.h>
int digitcount = 0;
%}
%%

        /* 규칙절 (rule section) */
999     {printf("%s is appeared\n",yytext);}
([0-9])+        {digitcount++;}
\n      ;
%%

/* 사용자 서브루틴절 (user subroutine section) */
int main(){
        yylex();
        printf("digitcount : %d\n",digitcount);
        return 0;
}

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

- 컴파일 방법

lex hw2_1.l
cc lex.yy.c –o lex
./lex < a.txt

Vi a.txt
999
033
024
156

Part 3.

과제과제

3.1 과제 1

- 텍스트에서 ‘love’라는 단어가 몇 번 나오는지 카운트하는 lex코드 작성
- 파일명 : hw2_1.l

```
@localhost abc]$ lex hw_2_1.l
@localhost abc]$ cc lex.yy.c -o lex
@localhost abc]$ ./lex < love.txt
number of love=10
```

3.2 과제 2

- Python 과제 6번을 lex 코드로 작성 (문자열의 개수 입력 받지 않음)
- (100~1~|01) ~을 만나면 is danger 출력

- 파일명 : hw2_2.l

```
[redacted]localhost pl]$ ./lex < pl6.txt
1000101 is danger
```

3.2 과제 3

- Word 참고
- 파일명 : hw2_3.l

3.2 과제 4

- 과제 1 ~ 3의 내용을 Latex으로 작성 후 제출

- 섹션 깔끔하게 나눌 것
- 파일명 : hw2.tex , hw2.pdf

Part 4.

Q & A

4.1 Q & A

- 제출 마감일, 방법
  - 제출 마감 시간 : 4월 15일 목요일 23시 59분까지 (1,2분반)(기한 : 2주)
    4월 16일 금요일 23시 59분까지 (3,4분반)(기한 : 2주)
  - 제출 파일 : hw2_1.l, hw2_2.l, hw2_3.l, hw2.tex, hw2.pdf
  - 제출 방법 : submit pem_ta hw2x
    (1분반 : a , 2분반 : b, 3분반 : c, 4분반 : d)
  - 제출 확인 : submit pem_ta hw2x -l
- 질문 : pemta818@gmail.com (메일제목 앞에 [Lex] 추가)
- 유의사항
  - Space 대신 tab 사용 권장
  - Late 제출, 메일 제출은 받지 않습니다
  - Cheating 시 무조건 F