Lex Yacc 실습하기

1 Lex 실습하기 (wordCount.l)

- wordCount.l 로 부터 생성되는 실행 파일 wc 는 표준 입력으로부터 텍스트의 라인 수와 문자 수를 카운트한 결과를 출력한다. 이 실행 파일을 생성하는 과정은 다음과 같다.
- flex 로 lex 파일을 실행하여 lex.yy.c 코드를 생성한다.
- 이 C 프로그램을 gcc 컴파일러 컴파일하여 실행 파일 wc 를 생성한다. 이때 필요한 라이브러리를 첨가하기 위히 -ll 옵션을 추가한다.
- 생성된 실행 파일 wc 를 실행한다. 이 프로그램의 입력은 표준 입력으로 받도록 정의되었다. 표준 입력 대신 파일로부터 입력을 받기 위해서는 < 를 이용한다. 프로그램 실행 결과 출력되는 내용이 맞는 지를 확인한다.

```
$ flex wordCount.l
$ gcc -o wc lex.yy.c -ll
$ ./wc < wordCount.l
```

2 lex.l 과 yacc.y 실습하기

- 강의 자료에 있는 yacc.y 와 lex.l 은 더하기 빼기의 산술식을 계산할 수 있는 계산기를 만들어 낼 수 있다.
- 강의 자료에 있는 대로 다음과 같이 실습한다.

```
$ flex lex.l
$ bison -d yacc.y
$ gcc -o parser yacc.tab.c lex.yy.c -ll
$ ./parser
10 + 30 - 5
=35
```

- 강의 자료 lex.l 에서 tab과 스페이스를 처리하는 부분에서 에러가 있으므로 이를 다음과 수정한다. 강의 자료에는 [\t]; 형태로 되어 세미 콜론이 띄워지지 않아서 white space의 처리가 제대로 이루어지지 않는다.

```
[ \t]           ;
```

- 위의 컴파일 과정에서 여러 warning 메시지가 발생하지만 실행 파일 parser 가 생성하기만 한다면 별 문제가 안된다.
- 생성되는 실행파일 parser 는 표준 입력으로 데이터를 받고 있음에 유의한다.

3 Calculator 폴더의 Makefile, calcul.l, calcul.y, hdr.h

- 여기에서는 Makefile 을 이용하여 파일간의 종속성(dependency)를 표현하고 있다. 예를 들어, calcul.l 파일로 부터 생성되는 lex.yy.c 파일은 calcul.l 에 종속적이다. 즉, calcul.l 이 수정된다면 lex.yy.c 또한 수정되어야 한다. 이때 lex.yy.c 파일의 생성은 flex 를 이용하여 수정된다.
- 여기에 소개된 코드는 강의 자료의 내용과 유사하지만 좀 더 확장된 형태이다. 강의 자료에서는 산술식과 할당문을 처리하도록 되어 있지만, 여기서 생성되는 실행 파일은 추가적으로 조건문을 실행할 수 있도록 확장되었다.
- 다음과 같이 실행해 본다. 이 과정에서 warning이 발생하지만 실행 파일이 생성되면 별 문제가 없다.

```
$ make clean
$ make     (실행 결과 다음과 같은 과정이 수행된다)
flex calcul.l
bison -d calcul.y
gcc -o cal lex.yy.c calcul.tab.c -ll

$ ./cal
x = 10
y = 20
x + y
 = 30
if (x > y) then x -3 else y + 3
=23
```