Overview

이번 과제의 목표는 lex 파일을 잘 작성하여, 입력된 문자열에서 단어와 특수기호의 개수를 각각 세는 프로그램을 생성하는 것입니다.

Lexical Analysis with Flex 에 따르면, lex 파일은 `%%` 문자로 구분되는 3개의 섹션으로 구성됩니다.

- ***Definitions***
- ***Rules***
- ***User Patterns***

또한, 이번 과제에서는 flex를 이용해 lex 파일로부터 `lex.yy.c` 소스코드를 생성하고 이를 컴파일하여 프로그램을 생성했습니다.

Sections

Definitions

Definitions 섹션에서는 `lex.yy.c` 에 포함될 텍스트 또는 특정 `패턴` 을 정규표현식으로 정의할 수 있습니다.

우선 단어와 특수기호 개수를 저장할 변수를 이 섹션에서 선언합니다.

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

또한, 이번 과제는 **단어**와 **특수기호**를 판별해야 하므로 아래와 같이 4개의 `패턴` 을 정의했습니다.

```
word        [a-zA-Z0-9]*
symbol      [!@#$%^&*()-_+=|\\/<>,.:;"'`~{}[\].]*
newline     \n
whitespace  [ \t]+
```

Rules

Rules 섹션에서는 각각의 `패턴` 을 발견했을 때의 `액션` 을 정의합니다. 예를 들어, 위에서 정의한 `word` 패턴을 발견한 경우 아래와 같이 단어의 개수를 세기 위해 선언한 변수값을 1만큼 증가시키는 등의 작업을 수행할 수 있습니다.

```
{word} {
  ++word_cnt;
}
```

User Patterns

User Patterns 섹션에서는 `lex.yy.c` 에 복사되는 코드를 작성할 수 있습니다.

과제를 위해 작성한 lex 파일에서는 `main` 함수와 `yywrap` 함수를 정의했습니다.

Code

상기한 내용을 토대로 작성한 전체 lex 코드는 아래와 같습니다.

```
// Definitions
%{
  #include <stdio.h>
  int word_cnt = 0;
  int symb_cnt = 0;
%}

word [a-zA-Z0-9']*
symbol [!@#$%^&*()-_+=|\\/<>,.;"'‘’~{}[\].]*
newline \n
whitespace [ \t]+

%%

// Rules
{word} {
  ++word_cnt;
}

{symbol} {
  ++symb_cnt;
}

{newline} {
  printf("%d %d\n", word_cnt, symb_cnt);
  word_cnt = 0;
  symb_cnt = 0;
}

{whitespace} {
  // ignore
}

%%

// User Patterns
int main() {
  yylex();
  return 0;
}

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

아래와 같이 `Makefile` 을 작성하여 `simple` 이라는 이름으로 프로그램을 생성합니다.

```
all: simple

simple: lex.yy.c
    gcc lex.yy.c -o simple

lex.yy.c: simple.l
    flex simple.l

clean:
    rm -f lex.yy.c simple simple.zip

zip:
    zip simple.zip Makefile simple.l
```

아래와 같이 입력하여 `simple` 프로그램을 생성한 후 문자열을 입력하면 아래와 같은 결과를 확인할 수 있습니다.

```
$ make
flex simple.l
gcc lex.yy.c -o simple

$ ./simple
Compilers and interpreters are a very interesting topic!
8 1
```