Compiler Project #1. Scanner Report

2018008277 노주찬

1. Compilation Environment

아래의 환경에서 프로젝트를 컴파일하고 테스트했습니다.

항목	값
OS	Ubuntu 20.04.4 LTS
Architecture	x86_64
C Compiler	gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Flex	flex 2.6.4

`1_Scanner` 디렉토리에서 `make all` 커맨드로 프로젝트를 빌드했습니다.

2. C Implementation Explanation

- 기존 구현에서는 identifier로 letter만 사용할 수 있었지만, C-Minus에서는 두번째 글자 이후로 letter와 digit을 자유롭게 사용할 수 있기 때문에, INID state에서 digit도 허용하도록 변경했습니다.
- START state에서 현재 character를 보고 currentToken을 바로 결정하는 토큰 case에 '[', ']', '{', '}', ','을 추가하고, '=', '<', '/'을 제거했습니다.
- START state에서 '<', '>', '=' 이 들어온 경우, 각각 별도의 state로 이동하여 다음 character로 '=' 이 들어온 경우에는 각각 '<=', '>=', '==' 토큰으로 처리하고, 그렇지 않은 경우에는 character를 unget하고 '<', '>', '=' 토큰으로 처리했습니다.
- START state에서 '!' 이 들어온 경우, 별도의 state로 이동하여 다음 character로 '=' 이 들어온 경우에는 '!=' 토큰으로 처리하고, 그렇지 않은 경우에는 character를 unget하고 ERROR 토큰으로 처리했습니다.
- START state에서 '/' 이 들어온 경우, 별도의 state로 이동하여 다음 character로 '*'이 들어온 경우에는 INCOMMENT state로 이동하고, 그 외의 경우에는 '/' 토큰으로 처리했습니다. 처음에 '/' 이 들어온 상황에서는 이 문자를 토큰에 저장해야 하는지 확실치 않아, 다음 character를 보고 '/' 토큰이 확정되었을 때 현재 character 값을 '/'으로 변경하여 저장하도록 처리했습니다.
- INCOMMENT state에서 '*'이 들어온 경우, 별도의 state로 이동하여 다음 character로 '/'이 들어온 경우에는 START state로 이동하고, 그 외의 경우에는 다시 INCOMMENT state로 이동 처리했습니다.

3. Lex Implementation Explanation

- Definition에서 identifier가 두번째 글자 이후로 digit을 허용하도록 정규식을 수정했습니다.
- Rule에서 “then”, “repeat” 등 C-Minus에서 존재하지 않는 예약어를 제거했습니다.
- Rule에서 ‘=’을 C-Minus에서의 의미에 맞게 EQ에서 ASSIGN으로 변경했습니다.
- Rule에서 ‘{‘을 C-Minus에서의 의미에 맞게 주석 처리 action에서 LCURLY로 변경했습니다.
- Ruel에 C-Minus의 새로운 예약어와 토큰을 추가했습니다. 이 때, ‘⟨=’, ‘⟨’와 같이 prefix와 토큰 전체가 일치하는 경우에는, 길이가 더 긴 쪽이 먼저 매칭되도록 순서를 위에 두었습니다.
- Rule에서 기존 “{“에서 사용하고 있는 Action을 기반으로 “/*”과 일치할 때, input()으로 받은 최근 토큰 2개가 “*/” 일 때까지 주석을 받아들이는 Action을 작성했습니다.

4. Examples and Results

```
1   /* new C-Minus Reserved Words */
2   int void while return
3
4   /* new C-Minus Tokens */
5   = ; , == > >= < <= != [ ] { }
6
7   /* new C-Minus ID */
8   int abcdABCD1234 = 1234;
9
10  /* state transition tests with an intended error */
11  >1 <2 =3 /5 !6 /* asdf ** */
12
13  /* unterminated comment */
14  /* it may cause infinite loop with an improper lex implementation
```

- Line 2에서는 C-Minus에서 추가된 예약어를 인식하는지 테스트합니다.
- Line 5에서는 C-Minus에서 추가되거나 변경된 토큰을 인식하는지 테스트합니다.
- Line 8에서는 C-Minus에서 digit이 포함된 ID를 정상적으로 지원하는지 테스트합니다.
- Line 11에서는 별도의 state에서 의도하지 않은 문자가 등장했을 때에도 정상적으로 동작하는지 테스트합니다.
- Line 14에서는 comment가 닫히지 않은 채로 input이 종료되었더라도 무한루프가 발생하지 않는지 테스트합니다.

```
C-MINUS COMPILATION: sample.cm
    2: reserved word: int
    2: reserved word: void
    2: reserved word: while
    2: reserved word: return
    5: =
    5: ;
    5: ,
    5: ==
    5: >
    5: >=
    5: <
    5: <=
    5: !=
    5: [
    5: ]
    5: {
    5: }
    8: reserved word: int
    8: ID, name= abcdABCD1234
    8: =
    8: NUM, val= 1234
    8: ;
    11: >
    11: NUM, val= 1
    11: <
    11: NUM, val= 2
    11: =
    11: NUM, val= 3
    11: /
    11: NUM, val= 5
    11: ERROR: !
    11: NUM, val= 6
    15: EOF
```

C Implementation과 Lex Implementation 모두 같은 output을 보여주어 정상적으로 작동함을 확인했습니다.