#1 Scanner

컴퓨터소프트웨어학부

2019054957 이용우

1. Compilation environment and method

- OS: Ubuntu 22.04.2 LTS(Windows 11 wsl2)
- Compiler: gcc 11.3.0, GNU Make 4.3, flex 2.6.4
- Method: 주어진 Makefile을 통해 실행파일 생성

2. How to implement & How it operates

1) C Implementation

Tiny Compiler의 구성을 C-Minus Compiler에 형식으로 변경했다.

globals.h에 TokenType을 새로 정의하고, util.h에 토큰에 맞는 출력형식도 지정했다.

StateType은 START, DONE, 그리고 IN을 prefix로 갖는 state로 scan.c에 정의되어있다.

scan.c의 getToken() 함수는 입력 파일로부터 character(이하 char)를 하나씩 읽으면서 DFA의 state transition을 진행한다. START state에서 시작하여 DONE state에 도달하면 currentToken으로 지정된 토큰을 반환한다.

첫번째 char input으로 토큰이 구분되는 single-character-token([+-*()[]{};,])의 경우는 START state에서 바로 DONE state로 transition되어 토큰이 발행된다.

첫번째 char input으로 구분되지 않는 토큰은 INNUM, INID, INEQ, INLT, INGT, INNE, INOVER, INCOMMENT, INCOMMENT_ state로 분기하여 다음 char input을 받아 이후 토큰을 결정한다.

- INNUM: digit([0-9])으로 분기. 다음 char input이 digit이면 INNUM state로 self-transition 진행. 다음 char input이 digit이 아닌 경우, 입력을 이전으로 되돌리고 NUM 토큰을 발행.
- INID: letter([A-Za-z])로 분기. 다음 char input이 (digit | letter)이면 INID state로 self-transition 진행. 다음 char input이 (digit | letter)이 아닌 경우, 입력을 이전으로 되돌리고 ID 토큰을 발행.
- INEQ: '='으로 분기. 다음 char input이 '='이면 EQ 토큰을 발행. 다음 char input이 '='이 아닐 경우, 입력을 이전으로 되돌리고 ASSIGN 토큰을 발행.

- INLT: '<'으로 분기. 다음 char input이 '='이면 LE 토큰을 발행. 다음 char input이 '='이 아닐 경우, 입력을 이전으로 되돌리고 LT 토큰을 발행.
- INGT: '>'으로 분기. 다음 char input이 '='이면 GE 토큰을 발행. 다음 char input이 '='이 아닐 경우, 입력을 이전으로 되돌리고 GT 토큰을 발행.
- INNE: '!'으로 분기. 다음 char input이 '='이면 NE 토큰을 발행. 다음 char input이 '='이 아닐 경우, 입력을 이전으로 되돌리고 ERROR로 처리.
- INOVER: '/'로 분기. 다음 char input이 '*'이면 주석으로 인지하고 INCOMMENT로 state transition 진행. 다음 char input이 '*'이 아닐 경우, 입력을 이전으로 되돌리고 OVER 토큰을 발행.
- INCOMMENT: 다음 char input이 EOF이면 ENDFILE 토큰을 발행. 다음 char input이 '*'이면 INCOMMENT_로 state transition 진행. 다음 char input이 '*'이 아닌 경우, INCOMMENT state로 self-transition 진행.
- INCOMMENT_: 다음 char input이 EOF이면 ENDFILE 토큰을 발행. 다음 char input이 '/'이면 주석이 끝난 것으로 간주하여 START로 state transition 진행. 다음 char input이 '*'이면 INCOMMENT_ state로 self-transition 진행. 다음 char input이 '*'이 아닌 경우, INCOMMENT로 state transition 진행.

ID 토큰은 Reserved words(if, else, while, return, int, void)와 일치하는지 lookup하여, 일치하면 Reserved words에 대한 토큰을 발행하고 일치하지 않으면 그대로 ID 토큰을 발행한다.

2) Lex Implementation

Lex의 문법에 맞춰 Definition Section, Rule Section, 그리고 Subroutine Section으로 구분한다.

- Definition Section

Definition Section은 C의 header를 include하고, Regular Expression을 정의한다.

C-minus의 convention에 따라 identifier의 regular expression을 수정했다.

```
identifier  {letter}({letter}|{digit})*
```

- Rule Section

Rule Section은 TokenString에 대응되는 토큰을 반환하는 Rule을 정의한다.

Comment의 경우에만 로직이 있는데, "/*"로 매칭하여 다음 char input이 '*'인지 검사하는 flag 변수인 star를 선언한다. 다음 char input을 받아서 '*'이면 star 변수를 TRUE로 설정하고 다음 입력을 받는다. Char input이 '/'라면 star 변수가 true일 때 주석을 빠져나온다.

- Subroutine Section

Subroutine Section은 Rule Section에서 사용할 함수들을 구현하는 부분이다.

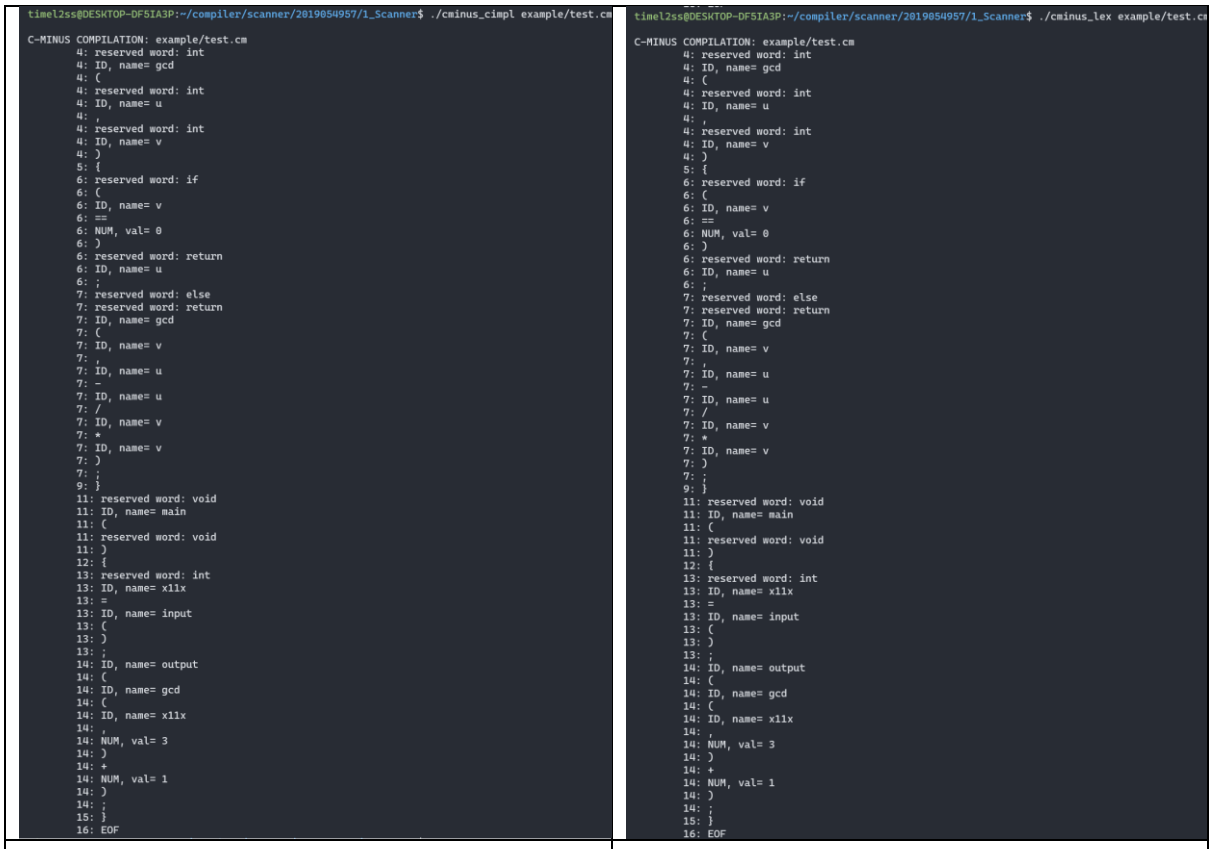
Rule Section에서 사용하는 getToken()이 구현되어 있고, tiny compiler의 내용을 변경하지 않고 그대로 사용하였다.

3. Examples & Results

```
int gcd (int u, int v)
{
    if (v == 0) return u;
    else return gcd(v,u-u/v*v);
    /* u-u/v*v == u mod v */
}

void main(void)
{
    int x11x = input();
    output(gcd(x11x,3) + 1);
}
```

위의 test input으로 실행파일을 구동했을 때의 결과는 아래와 같다.



cminus_cimpl	cminus_lex