**Compiler project1(Scanner) Report**

2013011112 소프트웨어전공

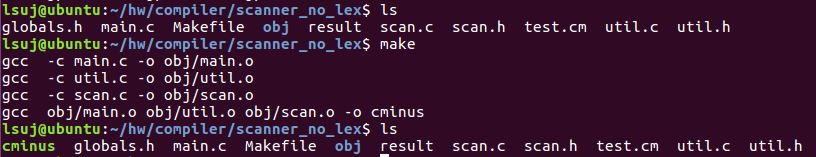
이 수 종

● **Compilation method and environment**

**<Compilation Environment>** - **OS** ( Ubuntu 16.04 - 64bit ) **gcc** 5.4.0

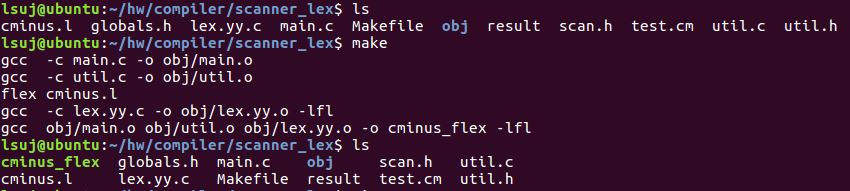
**<Compile method>**

**scan.c version**



scanner\_no\_lex 폴더에서 make하면 cminus파일이 생성됩니다.

**flex version**



scanner\_lex 폴더에서 make하면 cminus\_flex파일이 생성됩니다.

● **Explanation about how to implement and how to operate**

**<commonness>**

**main.c**



이번 project의 목표인 scanner기능만 사용하기 위해 TRUE값으로 변경하였습니다.

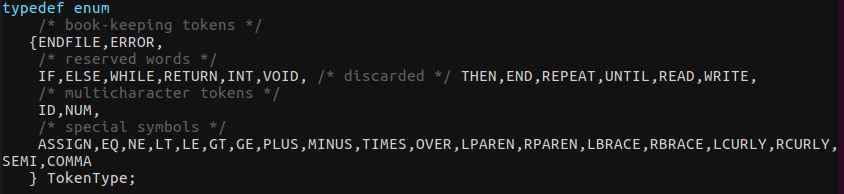


scanning을 수행하는 source code에서 해당 라인을 출력하고 scanning의 결과를 출력하기

위해 TRUE값으로 변경하였습니다.

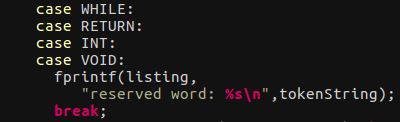
**globals.h**





기존 reserved word에 WHILE, RETURN, INT, VOID를 추가하여 MAXRESERVED를 12로 변경하였고 symbol에 NE, LE, GT, GE, LBRACE, RBRACE, LCURLY, RCURLY, COMMA를 추가하였습니다.

**util.c**



reserved word의 경우 출력하는 tokenString이 const char\*로 저장 되어있기 때문에 case만

추가하였습니다.



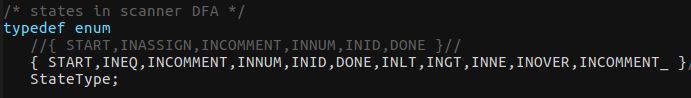
symbol의 경우 해당 case에 대한 symbol을 출력하도록 각각 위와 같은 형식으로 추가하였습니다.

**<scanner\_no\_lex>**

**scan.c**

getToken()함수에서 현재의 token을 계속 scanning합니다. single char로 구성되는 comma(,),

brace([, ]), curly({, })등은 scanning하는데 문제가 없습니다. 하지만 assign(=)과 eq(==)의 경우 다음 char까지 확인해야 판단할 수 있습니다.



위와 같이 INEQ, INLT, INGT, INNE, INOVER, INCOMMENT\_의 새로운 state를 정의하였습니다.

예를 들어 INEQ의 경우 EQ와 ASSIGN을 구분해야 하는데 ‘=’의 토큰을 만났을 경우 state를

INEQ로 바꾸고 다음 char를 받습니다. 다시 ‘=’를 만났을 경우 currentToken값은 EQ가 되고 ‘=’이

아닌 경우 ungetNextChar()함수로 뒤로 되돌리고 currentToken값은 ASSIGN이 됩니다. 마지막으로

State를 DONE으로 바꿔 결과를 출력하도록 합니다. 이는 INLT, INGT의 경우에도 같은 방식으로

구현됩니다.

INNE의 경우 ‘!=’는 정의되어 있으나 ‘!’는 정의되어 있지 않기 때문에 ‘!’이후에 ‘=’를 만나지

않은 경우 ungetNextToken()함수로 뒤로 되돌리고 state값을 START로 하여 다시 다음 token에

대한 scanning을 시작합니다.

OVER의 경우 ‘/’하나로 이루어 지는데 COMMENT는 /\*로 시작하여 \*/로 끝납니다. ‘\*’이외의

char이 나온 경우 currentToken값을 OVER로 하고 state를 DONE으로 하여 출력합니다. ‘/’뒤에

‘\*’이 나온 경우 state를 INCOMMENT로 하여 진행합니다. 이는 COMMENT가 시작되었음을

알립니다. 진행 도중 EOF를 만나면 currentToken을 ENDFILE로, state를 DONE으로 하여 getToken()

함수를 종료합니다. ‘\*’을 만나면 state를 INCOMMENT\_로 바꾸어 진행합니다. INCOMMENT\_에서

‘/’을 만나면 COMMENT의 종료이므로 state를 START로 바꾸어 다음 token을 scanning합니다.

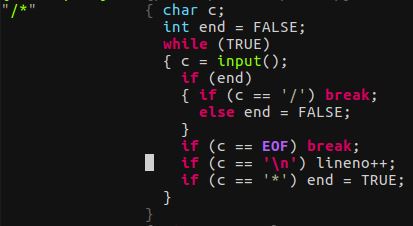
반대로 ‘/’이외의 값이 나오면 COMMENT가 계속 되어야 하므로 state를 다시 INCOMMENT로

바꿔줍니다.

**<scanner\_lex>**

**cminus.l**

lex는 ambiguous rule에 대해 longest match를 가장 우선시 하고 길이가 같을 경우 먼저 define된 rule을 우선시 합니다. 즉, ‘=’와 ‘==’와 같은 경우에도 longest match를 우선시하기 때문에 ‘==’가 ‘=’, ‘=’로 잘못 scanning되지 않습니다. COMMENT이외의 모든 추가된 Reserved word와 symbol에 대한 rule은 순서에 상관 없이 define해주었습니다.



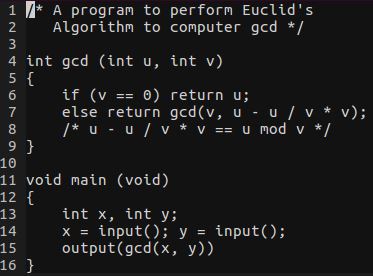
COMMENT의 경우에는 위와 같이 ‘/\*’입력을 받으면 계속해서 char를 받아오도록 하였습니다.

일단 ‘\*’를 만나면 end를 TRUE로 하고 이어서 ‘/’를 받으면 break하도록 하였습니다. 또한 ‘/’

이외의 값을 받으면 end를 FALSE로 하고 계속해서 입력을 받습니다.

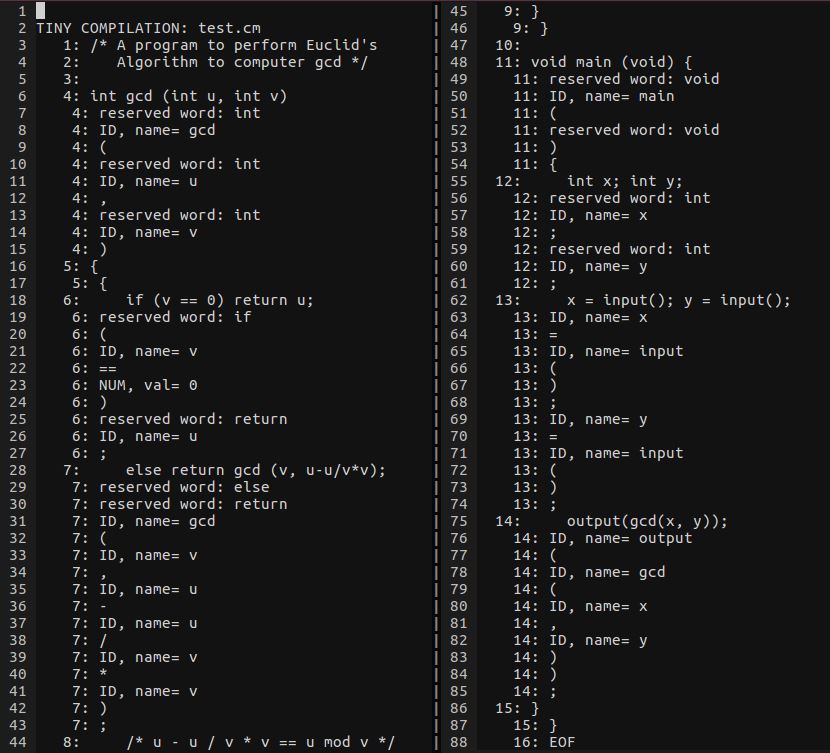
● **Example and Result Screenshot**

**<Example>**

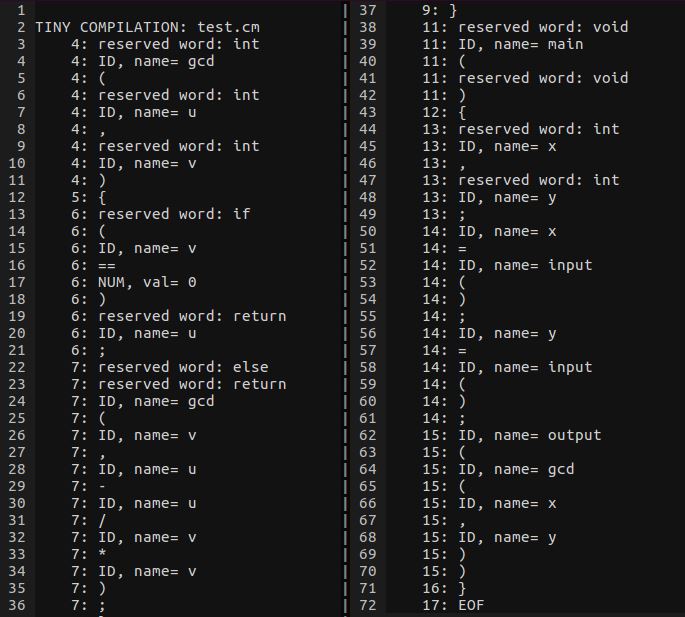
****

**<Result>**

**scanner\_no\_lex(cminus)**

****

**Scanner\_lex(cminus\_flex)**

****