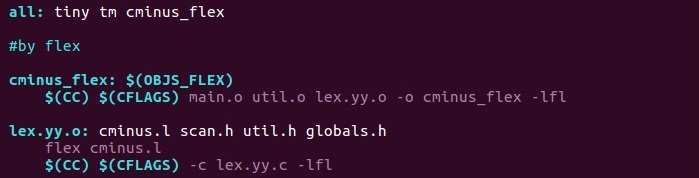
1st assignment - scanner

컴퓨터전공

2013011491

안찬영

**Compile environment**: ubuntu 16.04.4 LTS

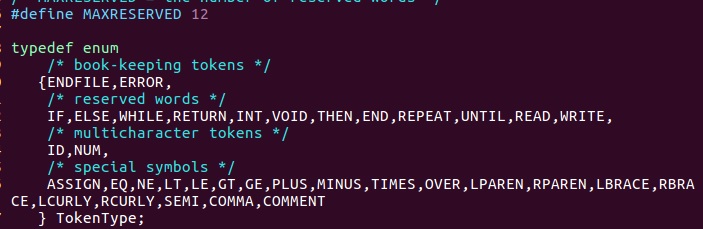
**Compile method**: 기본 제공된 Makefile에 그림과 같은 옵션이 추가 되었고,



Make all 명령어를 통해 컴파일 하면 tiny, cminus\_flex 실행파일이 생성됩니다.

**Tiny code 수정 과정**

Globals.h



먼저 MAXRESERVED의 값을 수정했습니다. 기존 reserved string 의 경우 8개였는데, 기존의 것들 것 삭제하지않고, 추가하였기 때문에 12로 증가하였습니다. TokenType에 reserved word들과 special symbol들을 추가하였습니다. 주석의 경우 ‘/\*’와 ‘\*/’를 따로 구분하지 않고 ‘/\*주석내용\*/’을 한꺼번에 COMMENT TokenType으로 처리하였습니다.

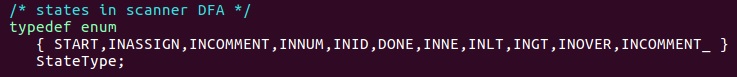
그림에 열거된 special symbols은 차례 대로 “=, ==, !=, <, <=, >, >=, +, -, \*, /, (, ), [, ], {, }, ;, **,**, /\*주석내용\*/”을 나타냅니다.

Main.c

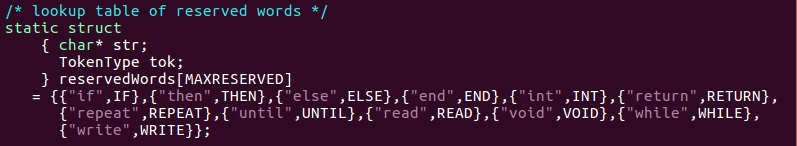


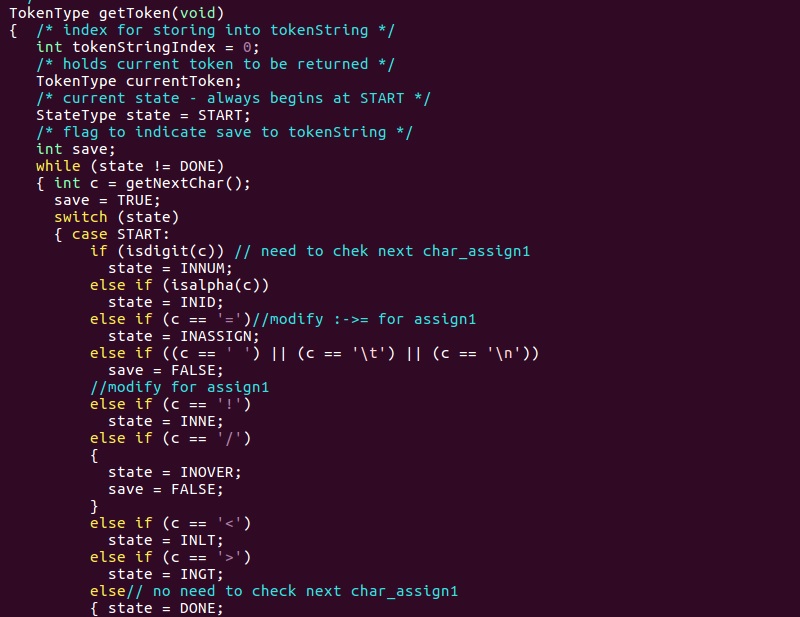
과제 hint와 주석에 나와 있듯이 Scanner만 사용하기 위해 설정을 바꿔주었습니다.

Scan.c

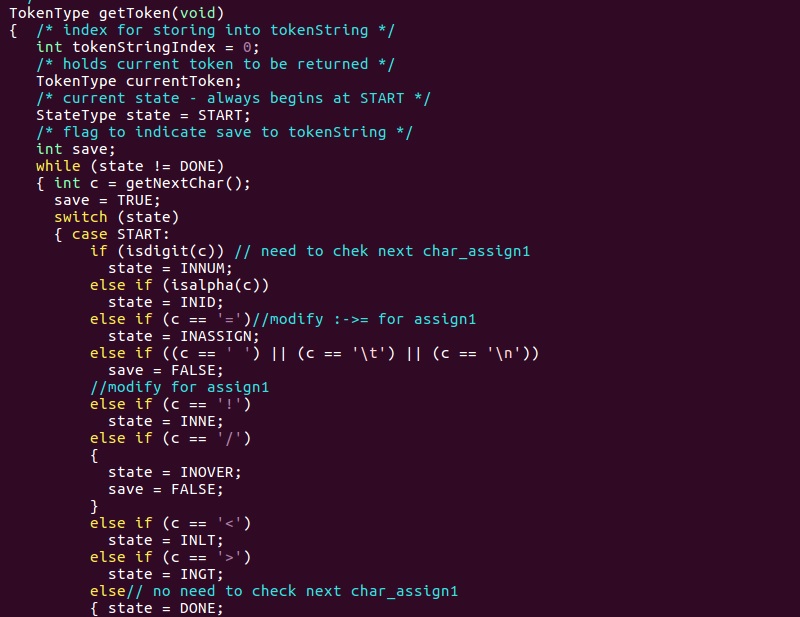


Scanner DFA에 필요한 state들입니다. 기본적으로 special symbol들 중에 그 다음 character 검사가 필요하지 않은 symbol들의 경우는 바로 DONE state로 가기 때문에 만들지 않습니다. 검사가 필요하지 않은 symbol은 “+, -, \*, ;, ,, (, ), {, }, [, ]”들이 있습니다.

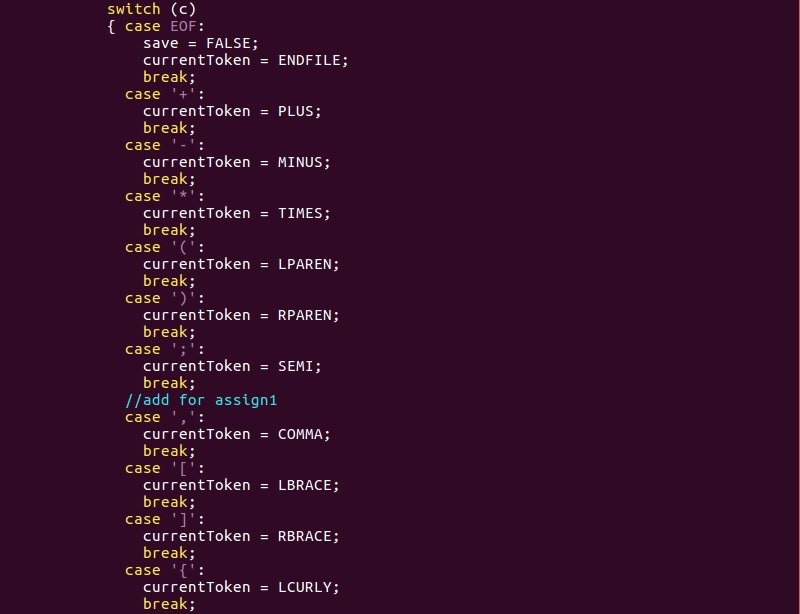


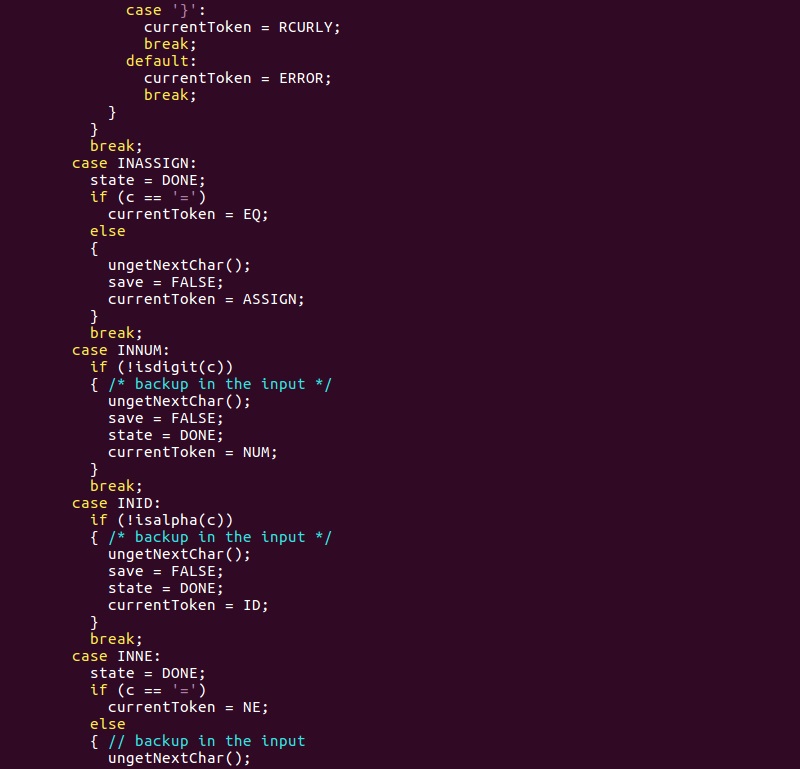
나중에 scan한 token을 해당 reserved word 테이블에서 찾아서 매칭하기 위해 word들을 추가해 주었습니다. 

getToken 함수입니다. Input stream에서 한 character씩 얻어와 검사하는 함수입니다. 기본적인 변수로는 save와 state, currentToken, tokenStringIndex가 있습니다. Save변수는 input받은 character를 저장할지 말지를 나타내고, state는 말그대로 dfa에서 현재 state를 나타내고, currentToken은 Token이 검사가 완료 되었을 때, 어떤 word인지를 나타냅니다.



우선 Start 상태로 시작해서 Input stream을 하나씩 받아들입니다. 이 때 해당 character뿐만 아니라 다음 character를 볼필요가 있는지 없는지에 따라 두가지 분기로 나뉘게 됩니다. 해당 character 외에도 다음 char 검사가 필요한 symbol은 “/, <, >, !, =, number, letter”가 해당 됩니다. 해당 하는 char에 따라 state가 INOVER, INLT, INGT, INNE, INASSIGN으로 바뀌게 됩니다.

 두 번째 분기는 이 다섯가지 symbol들을 제외하고는 나머지 char에 해당하고 이중에 char가 “+, -, \*, ;, ,, (, ), {, }, [, ]” 일 경우에는 state가 바로 DONE이 되며, 해당 character만 save=true로 저장하고, 어떤 symbol에 해당하는지 확인후 currentToken에 저장합니다. 나머지는 모두 ERROR Token이 됩니다.

 첫 번째 분기로 갈라진 다섯가지 state는 각각 다음과 같은 검사를 통해 다시 분류 됩니다.

INASSIGN : 현재 저장된 symbol은 ‘=’인 상태로, input이 또다시 ‘=’ 들어오면 TOKEN은 ‘==’에 해당하는 EQ상태로 분류를 끝내게 되고, 그 외에 input이 들어오면 해당 input을 저장하지 않고 ‘=’에 해당하는 ASSIGN Token으로 끝나게 됩니다.

INNUM: 현재 저장된 char가 number에 해당하고 input이 숫자이면 계속해서 저장하고, 숫자 외input이 들어오면 해당 input을 저장하지 않고 NUM Token으로 끝나게 됩니다.

INID: INNUM과 비슷하며 alphabet일 때만 받아들입니다.

INNE : 현재 저장된 symbol은 ‘!’인 상태로, input이 또다시 ‘=’ 들어오면 TOKEN은 ‘!=’에 해당하는 NE상태로 분류를 끝내게 되고, 그 외에 input이 들어오면 해당 input을 저장하지 않고 ERROR를 나타내게 됩니다.

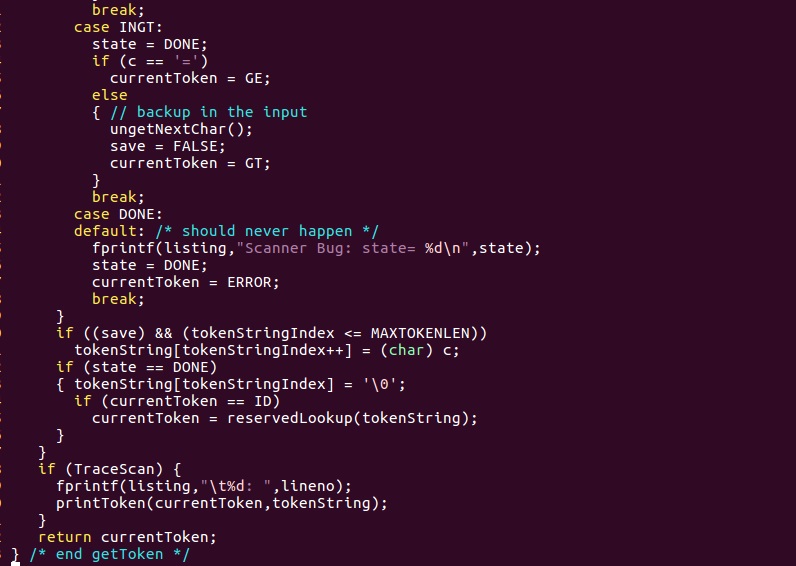
INOVER: 현재 저장된 symbol은 ‘/’인 상태로, input이 또다시 ‘\*’ 들어오면 주석에 해당하여 state가 INCOMMENT로 바뀌고, 그 외에는 저장하지 않고 OVER Token 상태로 끝나게 됩니다.

INCOMMENT: 어떤 input이 오든 받아들이다가 input이 EOF일 경우 ERROR, \*을 받을 경우에는 주석이 끝나는지 검사를 위해 state가 INCOMMENT\_상태로 변하게 됩니다.

INCOMMENT\_: 주석이 끝나는지 검사하게 됩니다. Input이 ‘/’이 될 경우 해당 주석은 끝나게 되고 아무 처리도 하지 않기 때문에 state는 START로 돌아가게 됩니다. EOF를 받을 경우에는 ERROR상태가 되고, 그 외에 input은 다시 state가 INCOMENT로 돌아가 주석 내용을 계속해서 받아들이게 됩니다.

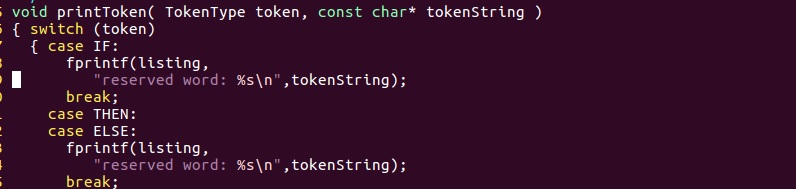
INLT: 현재 저장된 symbol은 ‘<’인 상태로, input이 또다시 ‘=’ 들어오면 TOKEN은 ‘<=’에 해당하는 LE상태로 분류를 끝내게 되고, 그 외에 input이 들어오면 해당 input을 저장하지 않고 ‘<’에 해당하는 LT Token으로 끝나게 됩니다.

INGT: 현재 저장된 symbol은 >’인 상태로, input이 또다시 ‘=’ 들어오면 TOKEN은 ‘>=’에 해당하는 GE상태로 분류를 끝내게 되고, 그 외에 input이 들어오면 해당 input을 저장하지 않고 ‘>’에 해당하는 GT Token으로 끝나게 됩니다.



Token이 분류되었으면 해당 Token이 ID일 경우에는 reserved word인지 검사하게 됩니다.

Util.c

해당 word를 적절하게 출력하는 함수입니다.

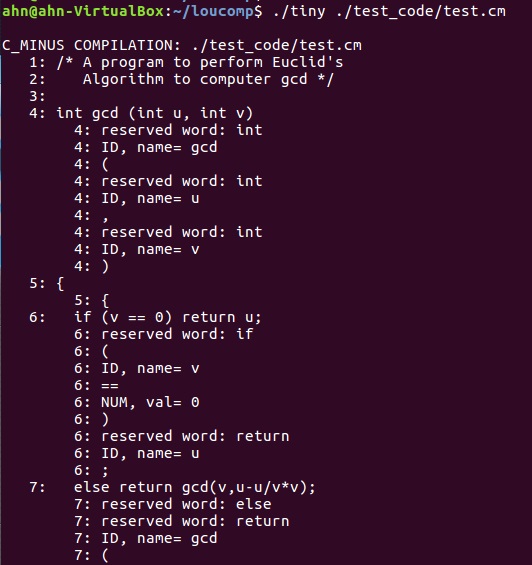
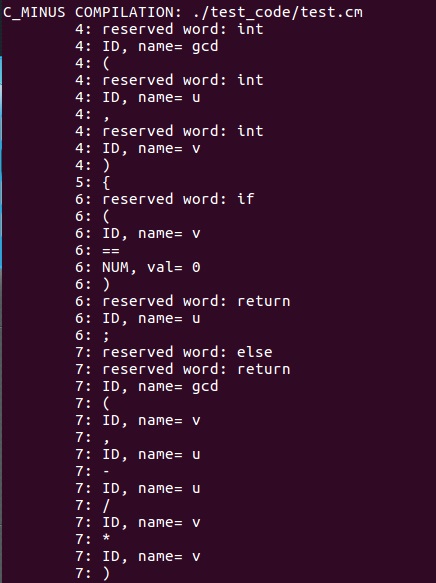
**Flex code 수정 과정**

Flex를 이용한 scanner 수정은 rule section에 해당 symbol들을 추가해 주기만 하면 됩니다.

주석의 경우(/\*)만 특별히 eof나 \*/이 오는지 검사하여 실행하게 됩니다.

**Example\_output**

제공된 test.cm 파일을 tiny와 cminus\_flex를 이용하여 실행하였을 때 입니다.

<tiny> <cminus\_flex>