컴파일러 설계 : 프로젝트 1

2014021358

노민수

I. 실습 환경

OS : Ubuntu 16.10

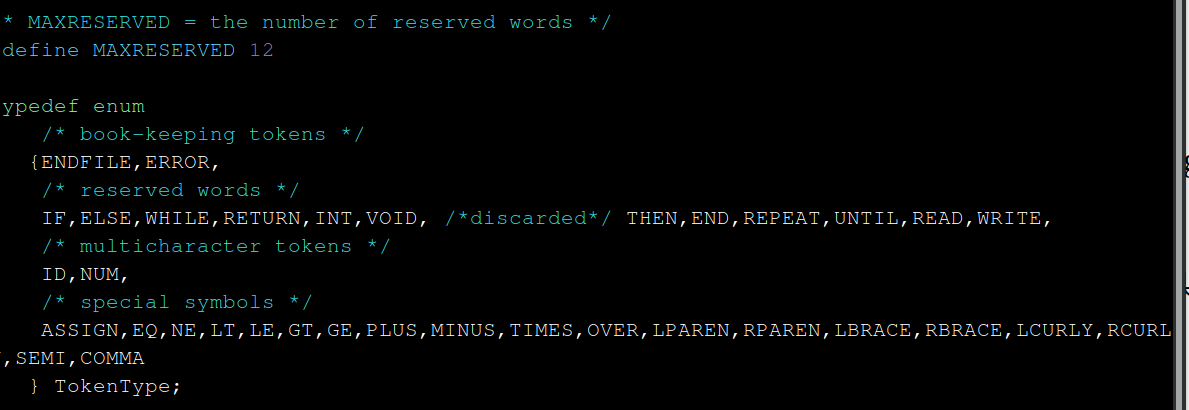
Requirement : gcc, flex

II. 실습 내용

1. Implementation of C-Scanner using C-code : modify Tiny compiler code

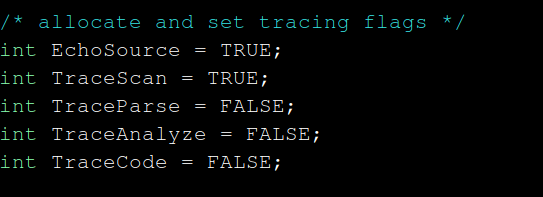
1) globals.h

maxreserved 갯수를 12로 바꾸고 , 정의한 키워드와 심볼이cminus의 문법과 맞게 토큰을 재설정함. (keywords : if, else, int, return, while, void), (symbols : +, -, \*, /, < , <=, >, >=, ==, !=, =, ;, ,, (, ), [, ], {, }, /\*, \*/)



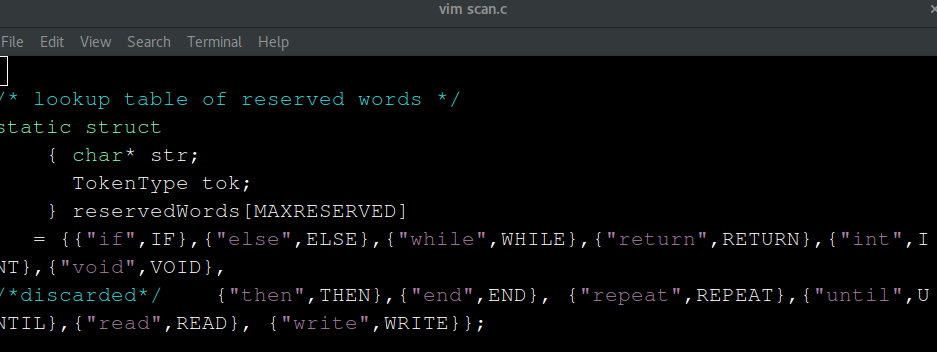
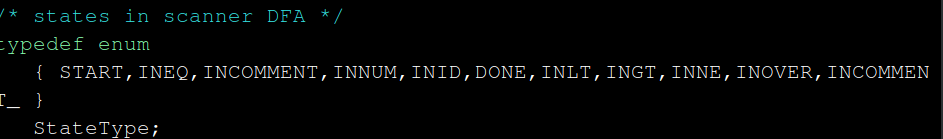
2) main.c

scanner 만 사용할 것이기 때문에 NO\_PARSE를 True로, tracing flags 에서는 EchoSource, TraceScan만 True로 설정

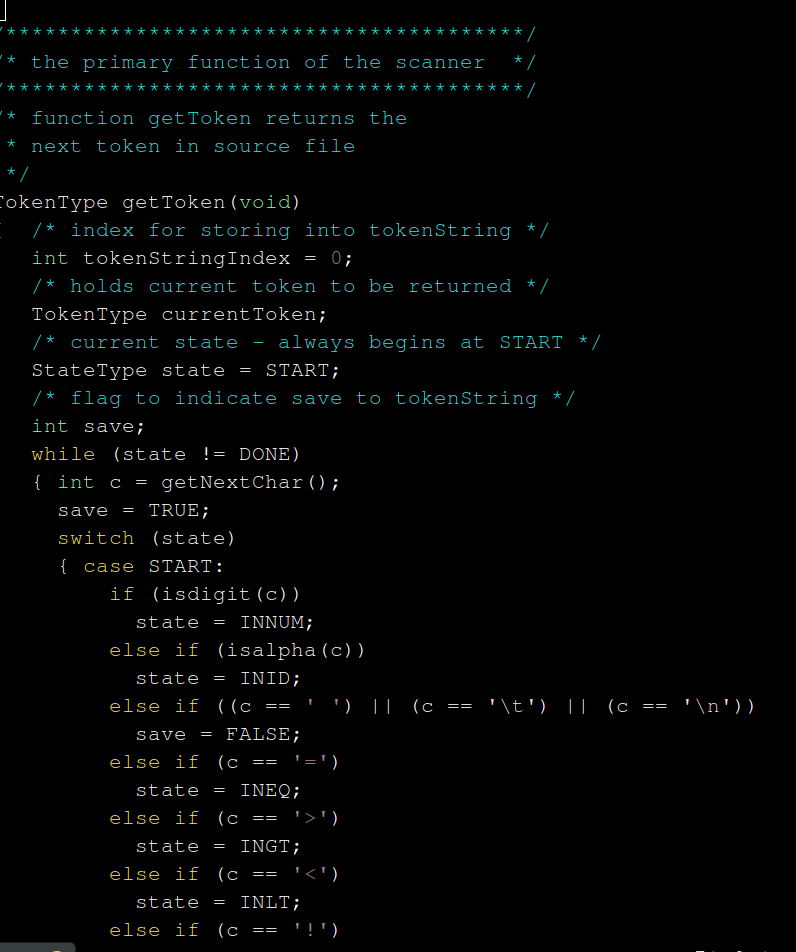
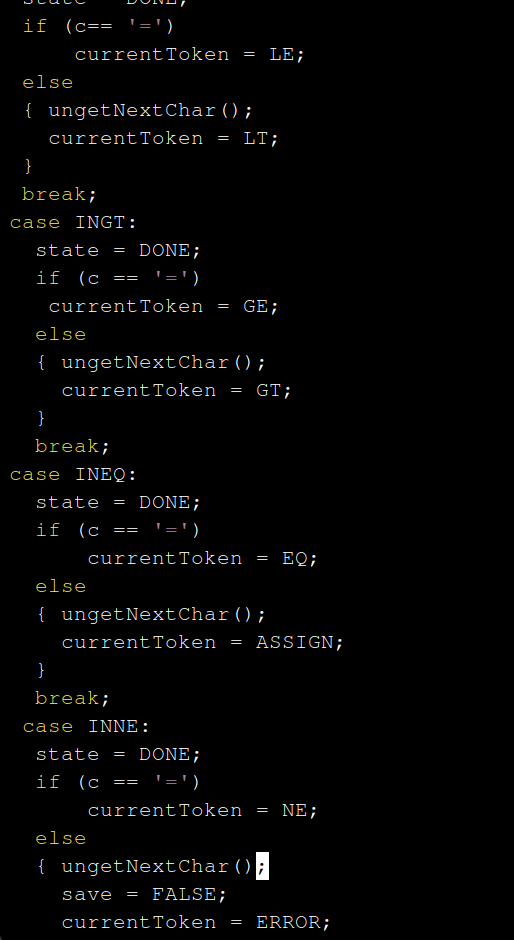
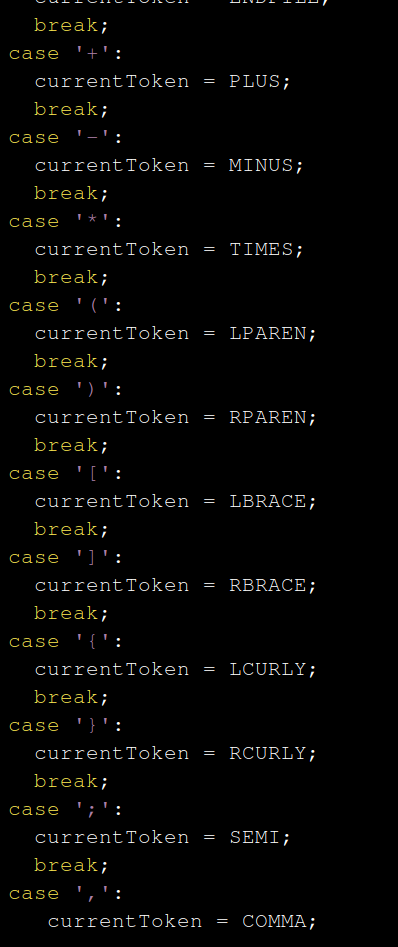


3) scan.c

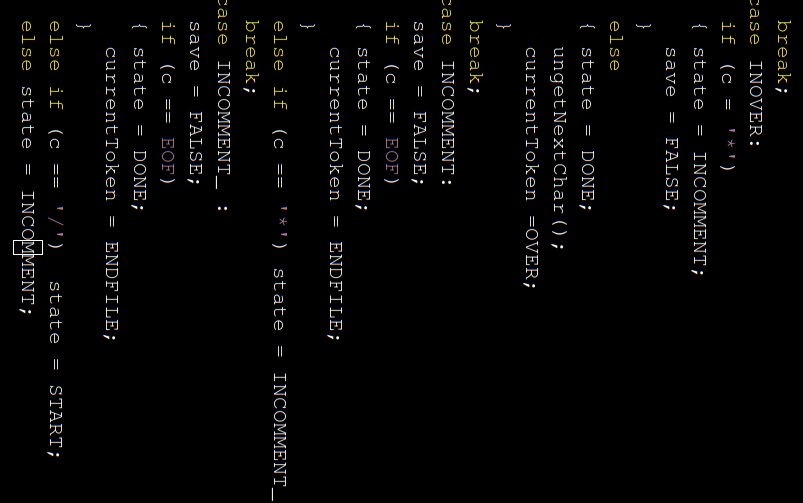
(1) c-minus로 DFA를 표현하기위해 그에 맞는 state들을 추가한다.



(2) 이번 과제에서 제일 중요한 primary function of scanner 부분을 c- minus DFA를 수행할 수 있게끔 고친다.(global.h에서 수정한대로 단어 와 심볼을 바꿔줌.)

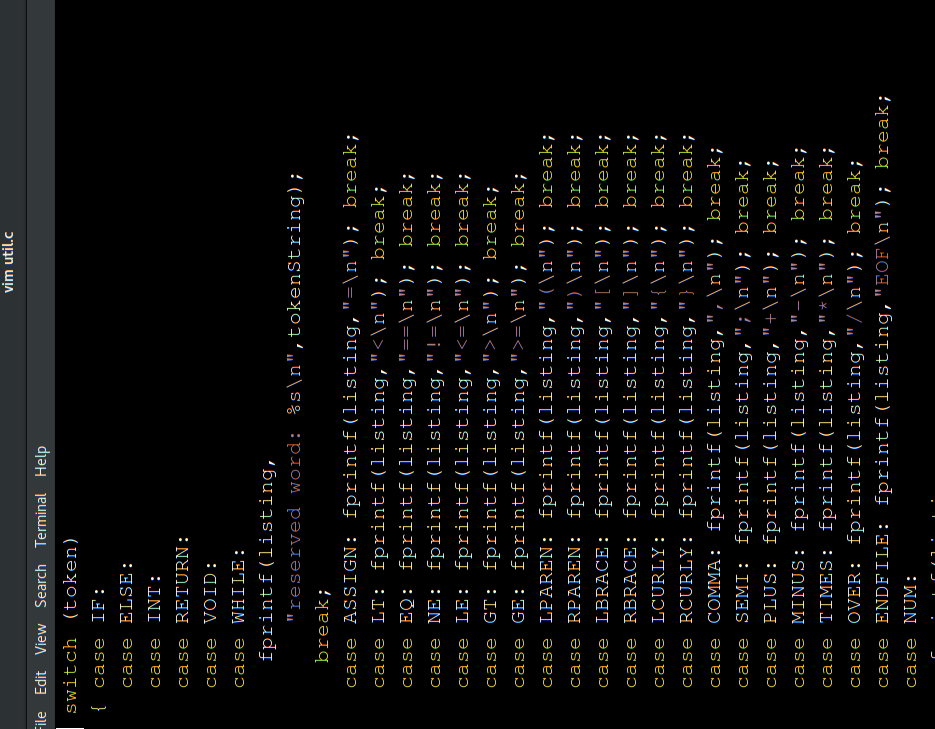


(3) 주석처리방식 다르므로 INOVER와 INCOMMENT(/\*인상태), INCOMMENT\_(/\* ~ \*인 상태, /가 한번 더 나오면 DONE으로 변경 후 읽지 않고 있던 STATE를 START로 바꿈)를 이용해 매 회 루프를 돌 때 해당 토큰에 맞게 상태를 수정하 는 방식을 취한다.



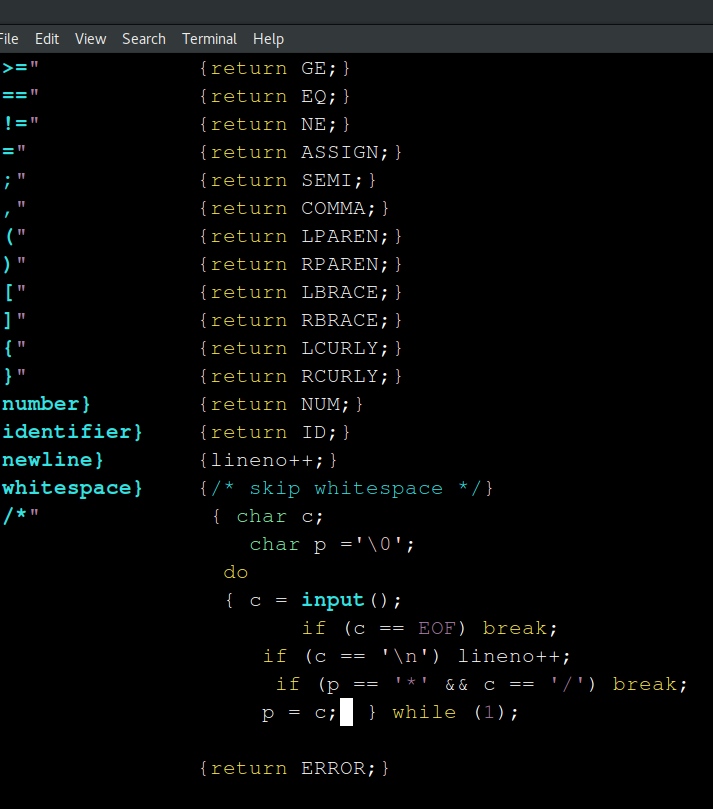
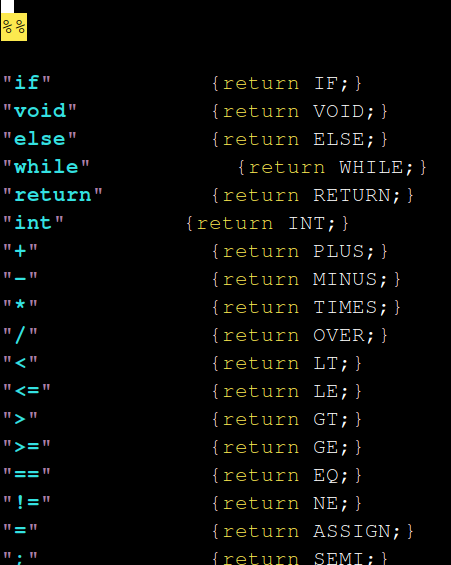
(4) util.c

여기서도 역시 상태 및 심볼들을 앞의 변수와 맞춰준다.



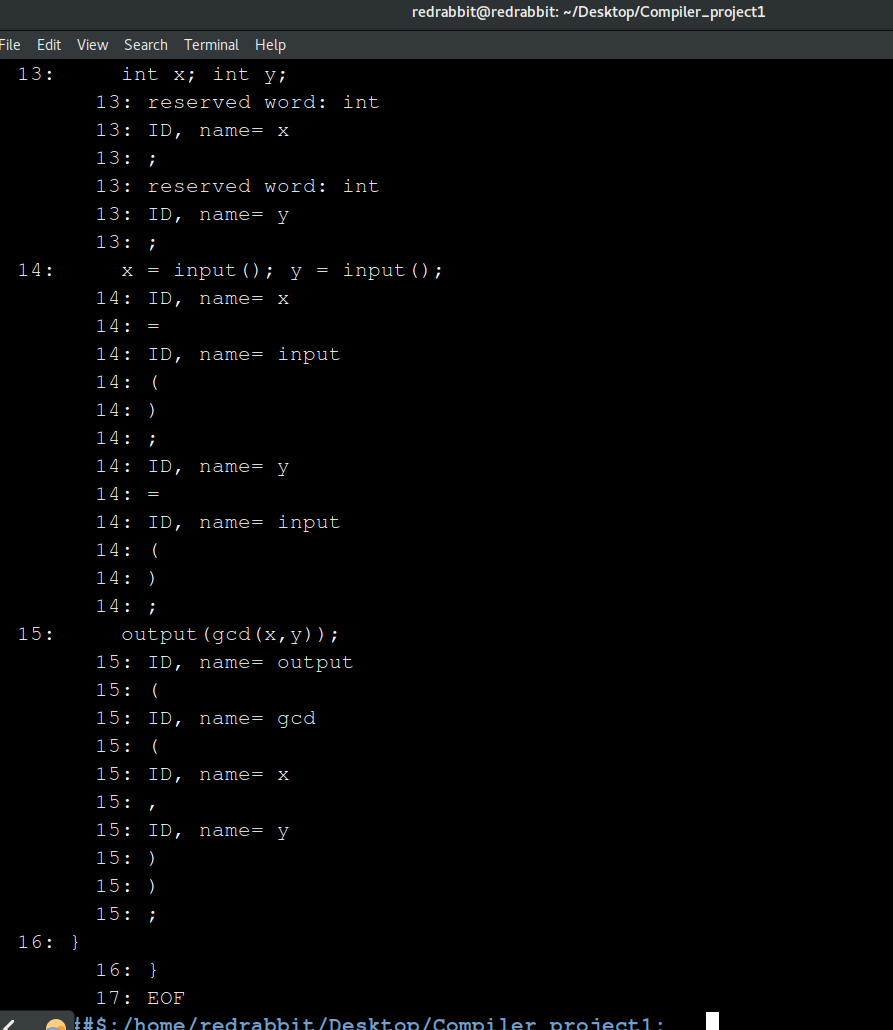
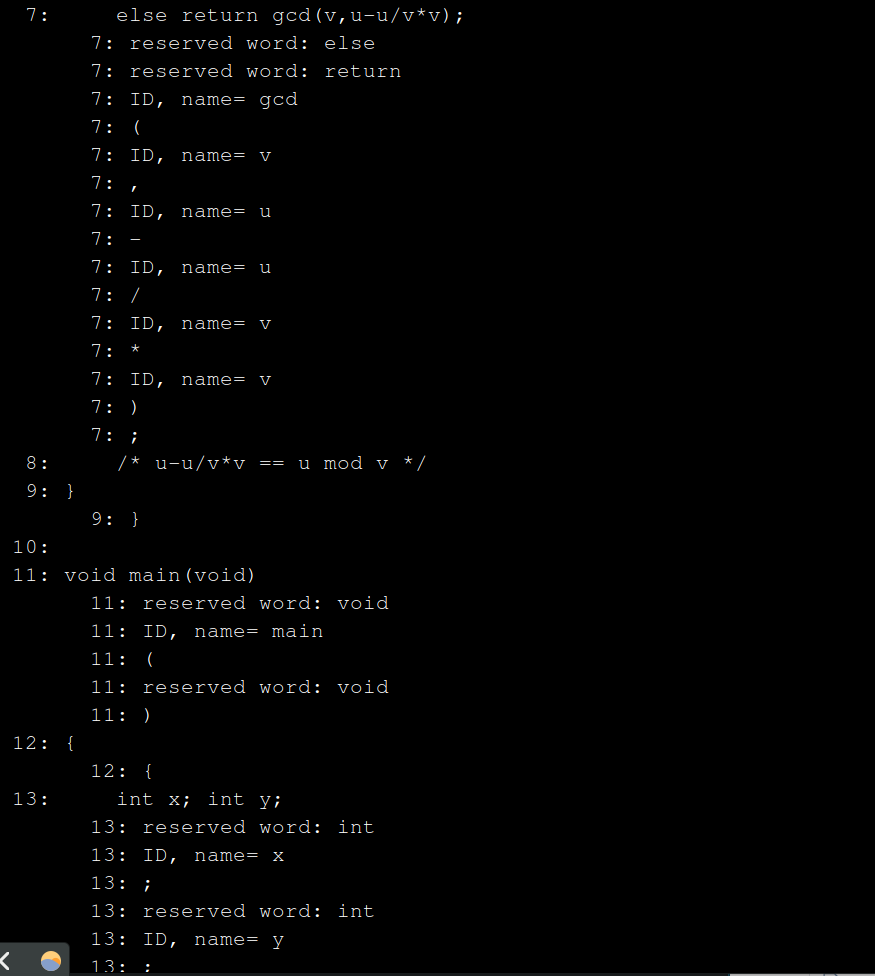
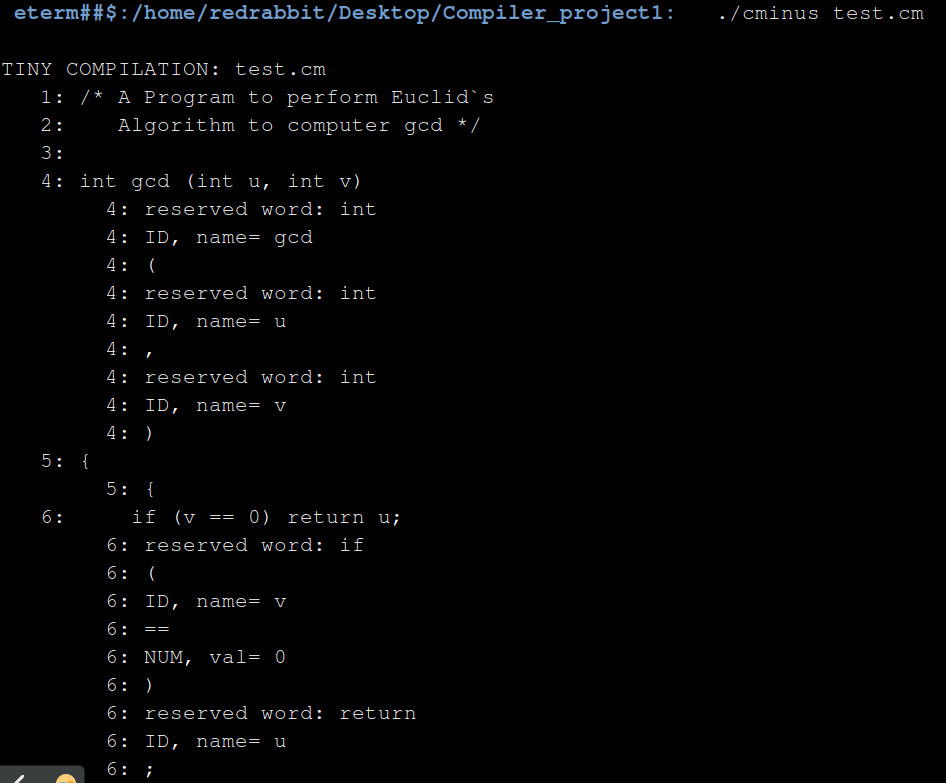
2) Implementation of C-Scanner using lex by Tiny.l modification

tiny.l파일만 cminus 문법에 맞게 수정해주면 된다.



III. 실습 결과

1. Manual : make cminus, ./cminus test.cm



2. lex : lex cminus.l, make cminus (scan.c 없이 새로) make lex.yy.o, make cminus, ./cminus\_flex test.cm

