**Design and Development of Compiler for C-Language**

**Phase 1: Design and Implementation of Lexical Analyzer**

**제출일 : 2019년 3월 27일**

**개발자 : 이희락 (20120121, 철학과)**

**1. 개발 목표 :** flex tool을 이용하여 C- language를 위한 lexical analyzer를 만든다.

**2.** **개발 범위 및 내용**

**가. 개발 범위 :**

C- language의 lexical convention를 참조[[1]](#footnote-1)하여 C-에서 유효한 token을 파악할 수 있도록 다음과 같은 코드파일을 수정 후 사용했다. 개발을 위해 수정한 파일들은 다음과 같다.

***main.c globals.h scan.h util.c util.h tiny.l[[2]](#footnote-2)***

**나. 개발 내용 :**

(1) 정규식 정의 : C- language의 lexeme들을 분석하기에 앞서 유효한 token들의 정규식이 어떻게 정의되어 있는지를 파악하고, 각각의 정의를 lex file인 tiny.l에 기술한다.

(2) C source code 수정 : lex file을 만든 이후, analyzer 생성을 위해 필요한 위의 C 파일들을 C- language에 맞도록 수정한다.

(3) C- 어휘분석 프로그램 생성 : lex.yy.c(tiny.l의 flex 결과 파일)과 위의 C 파일들을 이용하여 C-에 특화된 lexical analyzer를 만든다.

**3.** **추진 일정 및 개발 방법**

**가. 추진 일정**

3월 23일 : tiny.l, util.c, globals.h main.c를 수정 및 C- 어휘 분석 프로그램 생성.

3월 24일 : 예제 프로그램 생성. 및 디버깅. 결과 보고서 작성

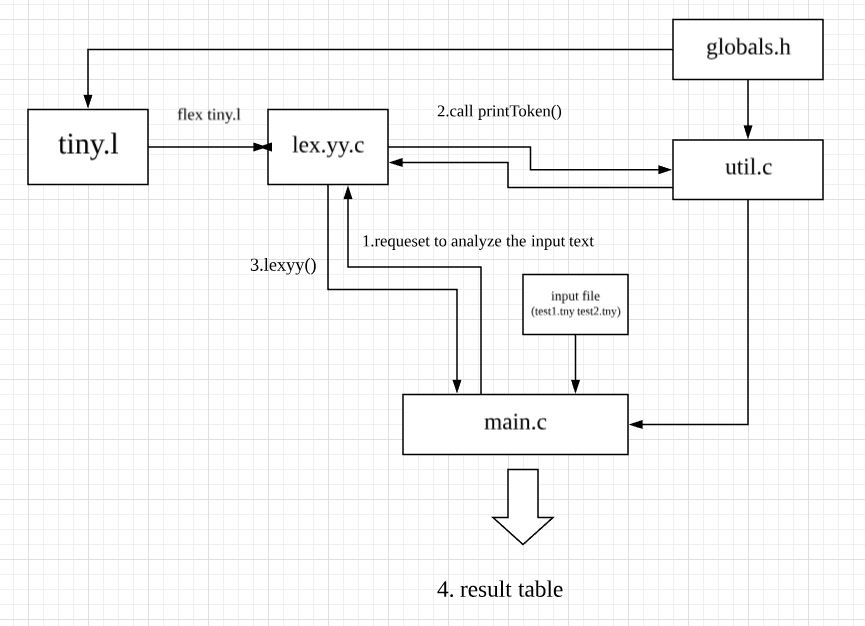
**나. 개발 방법**

tiny language lexical analyzer를 만드는데 필요한 ***main.c globals.h scan.h util.c util.h tiny.l*** 파일들(교재 502~511 제공) 기본 골격으로 삼아 이들을 C- language에 맞게 수정하는 방식으로 개발을 진행했다.

교재 491~492page에 정의되어있는 C-의 reserved word, special word, ID NUM과 같은 token들을 나타내는 정규 표현식을 확인한 후, lex 파일인 tiny.l의 deficition 부분을 앞선 정규표현식에 맞도록 수정했다. lex manual을 참고하여 위의 token들 뿐만 아니라 주석에 대한 상태도 추가로 정의하여 유효하지 않은 주석에 대해서는 예외처리하도록 설계했다.

cspro9 서버에 설치되어있는 flex tool을 사용했으며 tiny.l을 이용하여 만들어진 lex.yy.c와 앞서 수정한 C파일들을 gcc 컴파일러를 이용하여 최종적인 프로그램 20120121을 생성하였다.

**4.** **연구 결과**

**가. 합성 내용 **

**<그림1. 프로그램 구성도>**

tiny.l은 C- language의 유효한 token들을 정의하고 이에 해당하는 lexeme들에 따른 action을 정의하고 있는 lex 파일이다. flex tool을 통해 lex.yy.c를 생성할 수 있으며 이는 어휘 분석을 위한 모듈인 lex.yy()를 제공한다. lex는 %x를 통해 특정한 상태를 정의하고 이에 이어지는 action을 <name of state> expression으로 명세할 수 있도록 기능을 제공하는데, 주석 처리를 위해 CMT(comment)와 NCMT(nested comment)를 정의하고 이어지는 action을 정규표현으로 명세하여 유효하지 않은 주석에 대해서는 예외처리했다.[[3]](#footnote-3)

util.c는 tiny.l에서 정의한 getToken()함수에서 호출하는 printToken()을 정의하고 있으며, 분석된 어휘들에 대한 결과를 table의 형태로 출력해준다.

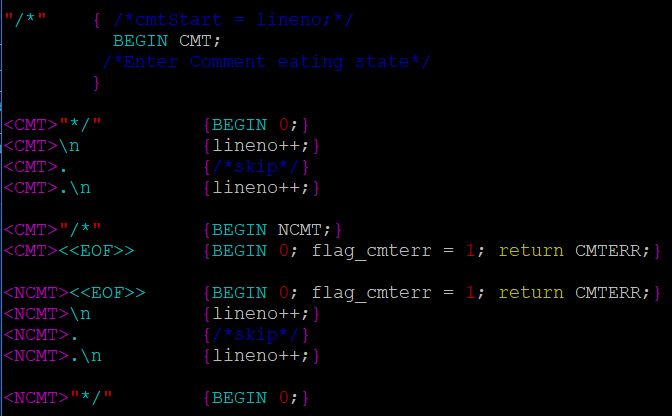
main.c에서는 globals.h에서 정의된 전역변수들을 초기화 하고있으며, input으로 받은 text파일 (여기서는 c- source code 파일)을 분석하기 위한 getToken()함수를 lex.yy.c로부터 호출한다.

**나. 분석 내용**

**(1)tiny.l** : C-에서 사용되는 token들을 인식할 수 있도록 definition part의 정규식을 수정했다.

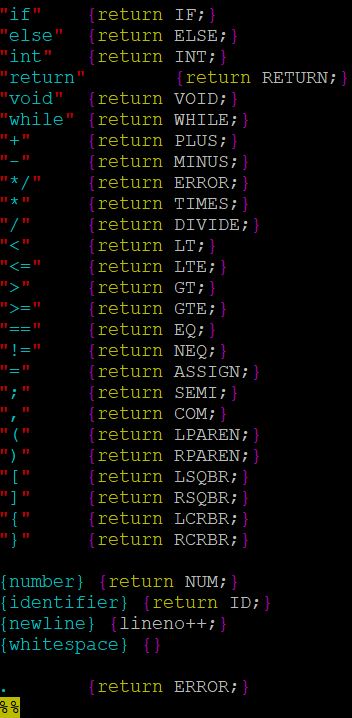
**-주석** : 주석처리의 경우 CMT(주석)와 NCMT(nested된 주석)의 두 가지 상태를 별도로 정의한 뒤, 유효하지 않은 주석에 대해서는 적절한 예외처리가 이루어지도록 설계했다. 유효한 주석은 /\* ~ \*/ 뿐이며 nested comment의 경우 /\* /\* \*/는 허용하나, /\* /\* \*/ \*/와 같은 경우 마지막 \*/를 유효하지 않은 token으로 인식하도록 예외 처리 했다. /\*만 사용하고 \*/로 마무리 짓지 않은 상황은 주석 처리 예외로 이후 나오는 모든 token을 그냥 consume하도록 설계했다. 이를 포함한 모든 주석 예외 상황의 경우 CMTERR이라는 추가로 지정한 token state를 반환하도록 했다.





<그림2. 주석 처리를 위한comment state CMT와 nested comment state NCMT>

-**token 정의**: C-에서 사용하는 reserved, special word, ID, NUM 등은 다음과 같이 definition 부분에서 정의하였다.

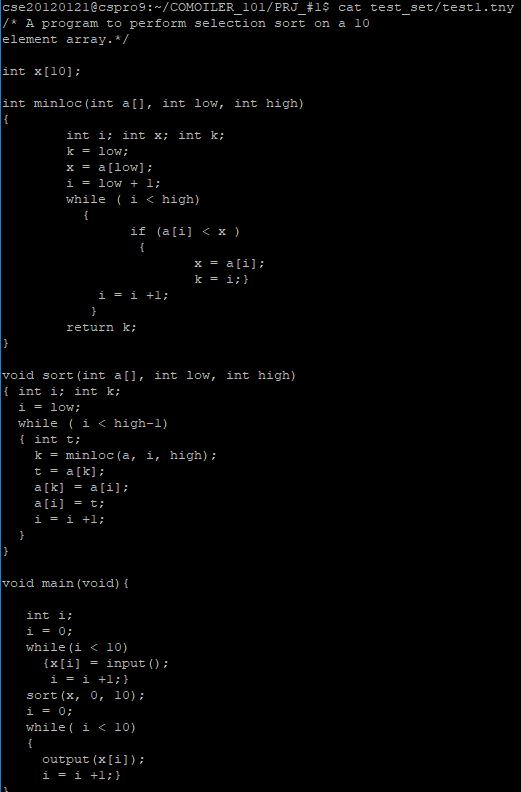


**(2) globals.h :** 어휘 분석과정에서 사용되는 전역 변수를 선언하고 있는 파일. 열거형변수TokenType을 C-에 맞도록 수정하였으며 교재에서 제공한 것과 별도로 주석 오류를 나타내는 CMTERR를 추가하였다.

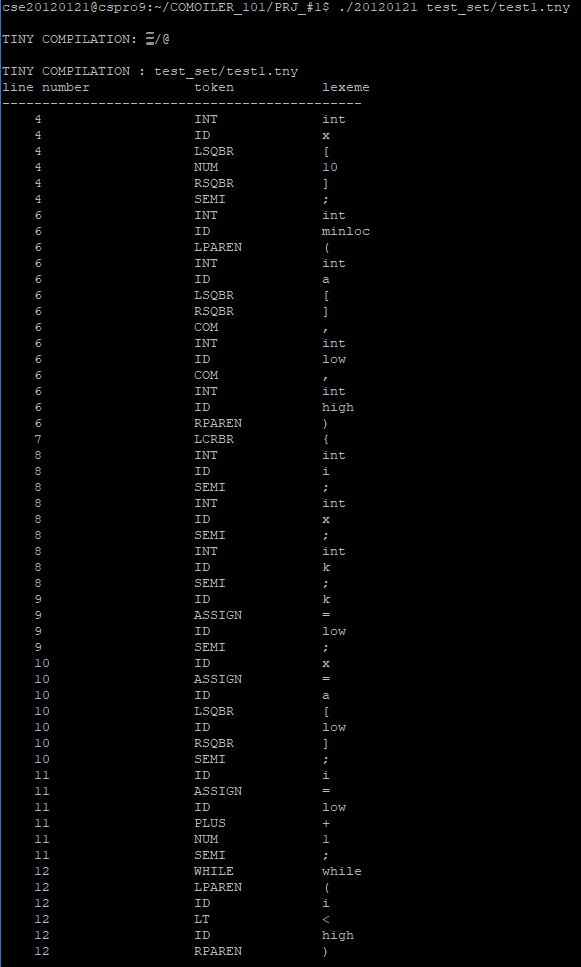
**(3) util.c :** tiny.l에서 정의한 보조 함수 getToken()에서 어휘 분석 결과 출력을 위해 호출하는 printToken()을 정의하고 있는 파일이다. table의 형태로 유효한 token에 대해서 각 token의 type과 내용을 table의 형태로 출력하도록 설계하였다.

**다. 제작 내용**

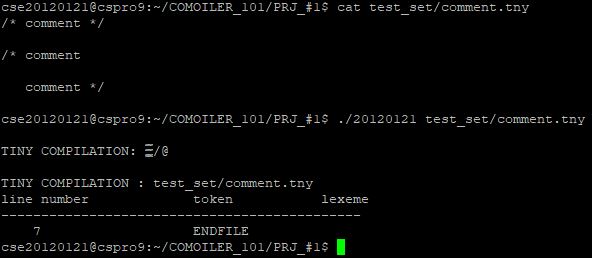
***test case1) input - 10개의 수를 insertion sort하는 프로그램***

****

**test case1) - result**

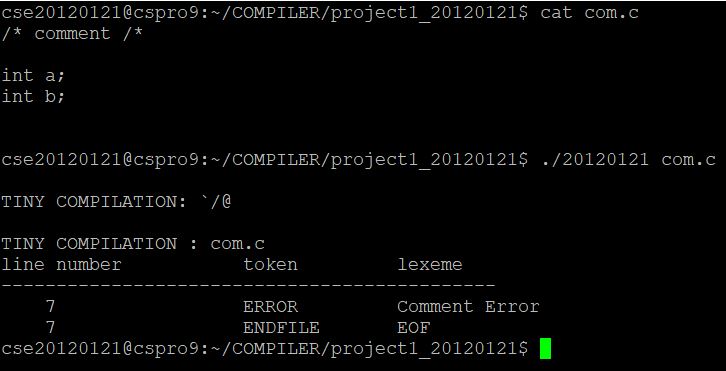
****

**test case 2) comment – 허용되는 comment**

****

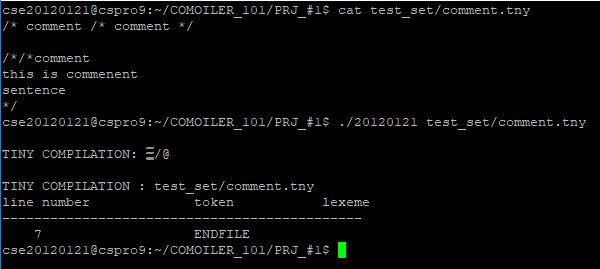
주석은 /\* ~ \*/을 원칙으로 한다. 그 안의 내용은 모두 consume되어 표시되지 않는다.

**test case 3) comment – 허용되지 않는 comment**

****

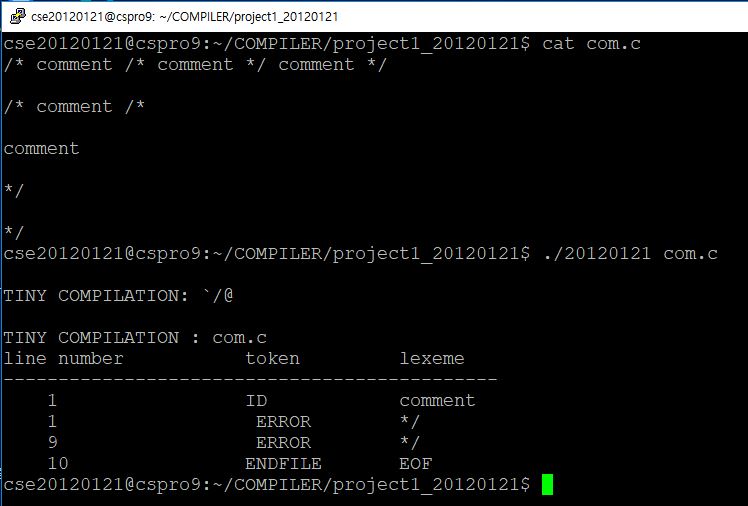
위와 같이 /\* 이후 \*/로 종료되지 않는 경우 comment error가 되며 이후 오는 모든 lexeme들은 인식하지 않고 모두 종료되지 않는 comment로서 consume되어 표시하지 않는다.

**test case 4) comment – 허용되는 comment(2)**

****

그러나 /\*~ /\*~ \*/의 경우는 내부의 /\*는 주석의 일부로 인식하여 허용한다.

**test case 4) comment – 허용되지 않는 comment(2)**

****

/\* /\* \*/ \*/ 와 같은 경우 앞선 세개의 /\* /\* \*/를 하나의 주석으로 종료 짓고 이후 나오는 \*/는 “/\*”없이 종료된 comment로 인식하게 했다. 그러므로 이러한 상황은 COMMENT ERROR로 처리하여 nested comment를 허용하지 않았다.

**라. 시험 내용**

결과로서 만들어진 C- Lexical Analyzer는 C-에서 허용하는 Special word, Reserved word, Identifier, Number의 형식을 파악하여 각각에 해당하는 lexeme들을 분류할 수 있어야 한다. 또한 /\*~\*/으로 표현되는 주석의 경우는 안정적으로 skip할 수 있어야하며 잘못된 주석의 경우 COMMENT ERROR을 표시해줄 수 있어야한다.

tiny.l 파일에 C- language에 적합한 regular expression을 definition 부분에 추가하여 각각에 대한 token type을 적절히 반환하여 출력하도록 설계하였으며, 주석의 경우 LEX에서 제공하는 state expression 및 state operation인 %x BEGIN {} <>등을 적절히 활용하여 자칫 infinite loop에 빠질 수 있는 주석처리를 안정적으로 수행하는 프로그램을 설계할 수 있었다.

**마. 평가 내용**

주석 처리에 있어서 LEX가 제공하는 state operation을 사용하지 않았을 때에는 예외사항을 모두 hard coding해야했기 때문에 infinite loop에 빠지거나 code가 매우 난잡해지는 문제에 당면했었다. 그러나 COMMENT state를 별도로 지정하고, comment error를 두 가지 경우로 분할하여 각각을 처리하는 방식을 도입함으로써 코드 작성에서의 편리함과, 가독성 그리고 예외처리에 있어서의 안정성을 높일 수 있었다.

**5. 기타**

**가. 자체 평가**

교재에서 제공하는 open source를 활용한 것이므로 독창적인 아이디어를 발휘하는 것보다는 compiler 수업에서 배운 개념을 어떻게 도입하며, 새로운 tool인 flex를 어떻게 활용하는지에 더욱 초점을 맞추어 1번 프로젝트를 진행했다. 그러므로 프로젝트의 목적 역시 독창성보다는 안정성을 높이는 것에 더욱 초점이 맞추어 졌는데, 이점에 있어서 tiny.l을 regular expression을 잘 활용하여 수정했으며 여러 예외 처리를 수행할 수 있도록 설계했다는 점에서 의미가 있다고 생각한다.

**나. 느낀 점**

교재의 내용을 꼼꼼히 살펴보고, 참고도서 lex & yacc를 차근히 살펴본 것이 매우 도움이 많이 되었다. 새로운 것을 시작하기에 앞서 필요한 내용을 숙지하고 충분히 이해하는 것이 프로그램 작성의 속도를 높이고 이해를 돕는다는 점을 다시금 느낄 수 있었다.

1. Compiler Construction-Principles and Practice(1997), Kenneth C. Louden , 491~492p [↑](#footnote-ref-1)
2. 같은 책 502~511p [↑](#footnote-ref-2)
3. <https://www.ibm.com/support/knowledgecenter/ko/ssw_aix_71/com.ibm.aix.genprogc/lex_program_start.htm> [↑](#footnote-ref-3)