[컴파일러설계]

Scanner

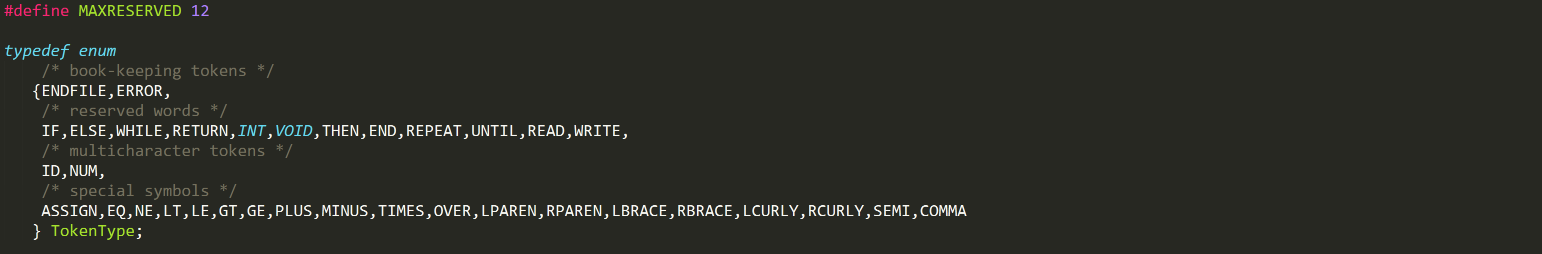
컴퓨터전공

2013011822

김병조

# Modification Tiny compiler code (globals.h, main.c, util.c)

**globals.h**



C-Minus의 새로운 reserved word (ELSE, WHILE, RETURN, INT, VOID)와 special symbol (NE, LE, GT, GE, LBRACE, RBRACE, LCURLY, RCURLY, COMMA) 를 TokenType에 추가한다. 그리고 그에 따라 MAXRESERVED 값을 수정한다. Tiny 컴파일러의 reserved words 는 아직 삭제하지 않은 상태이다. 이는 다음 과제 때 삭제 하려고 한다.

**main.c**





scanner-only compiler 와 tracing flags를 위해 flag 들을 수정한다.

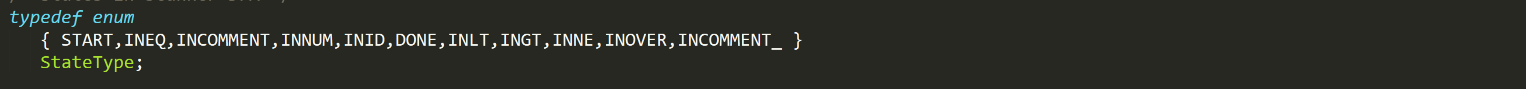
**util.c**



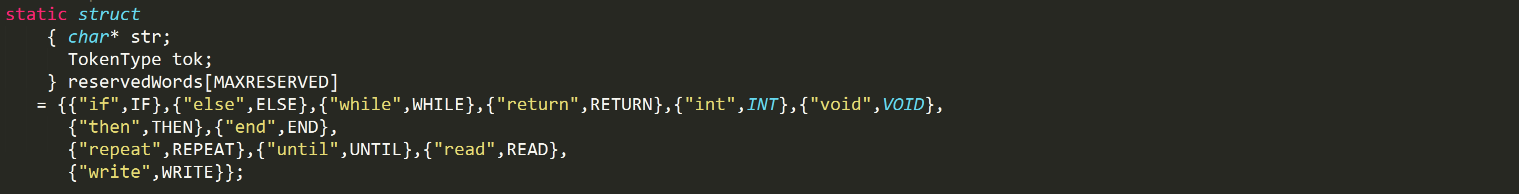
printToken에 case를 추가한다. ELSE, WHILE, RETURN, INT, VOID 라는 토큰이 들어올 경우 “reserved word: “를 출력 하도록 하고 NE, LE, GT, GE, LBRACE, RBRACE, LCURLY, RCURLY인 경우 각각의 문자열을 출력 하도록 한다.

# Implementation of C-Scanner using C-code by scan.c modification

**scan.c**



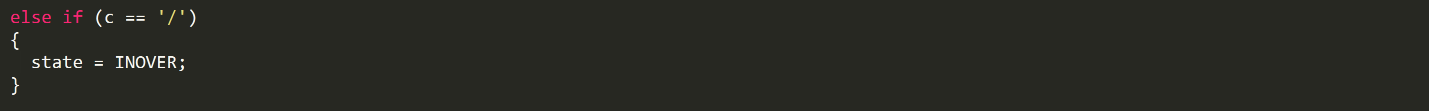
DFA 사용하기 위한 state를 추가한다. <=, < >=, >, !=, ==, =, /, /\*\*/ 를 사용하기 위해 스테이트를 추가한 모습이다.



추가된 reservedwords를 위해 스트링 값과 토큰타입을 매치 시켜 놓는다.

getToken() 함수는 문자열을 하나 받고 그것의 토큰을 판단해서 return 하는 함수 이다. START state에서 시작해서 우선 숫자, 알파벳인지, 공백, 탭, \n 를 판단 한다. /\*\*/와 /, ==와 =, <와 <=, >와 >= 그리고 !=를 판단 하기위해 문자열이 들어 왔을 때 그거에 맞는 state로 변환을 시킨다. 그 후 다음으로 들어오는 문자열을 이용하여 토큰을 판단한다. 나머지 symbols들은 switch case 문을 이용하여 토큰을 리턴 한다.

**/\*\*/ or / 판단**



state가 START 일 때 문자열에 / 가 들어오면 state를 INOVER로 바꾸고 다음 문자열을 받는다.

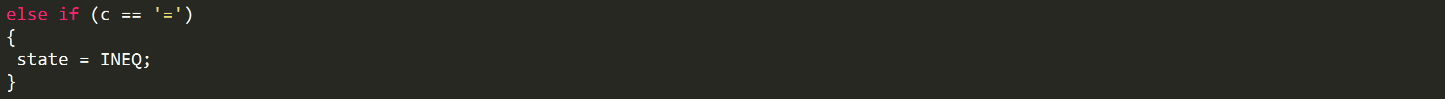


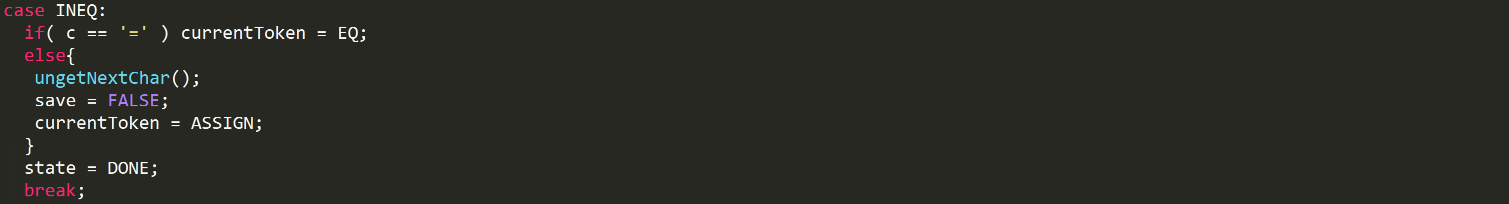
INOVER state에서 문자열 \* 이 들어오면 /\* 인 것으로 확인하고 state를 INCOMMENT로 바꾼다. 하지만 다른 문자열이 들어오면 / 를 over로 판단을 하여 currentToken을 OVER로 설정한다. 이때 문자열을 한번 더 받았기 때문에 ungetNextChar()를 호출 하고 state를 DONE으로 설정하여 다음 문자열을 다시 받도록 한다.



INCOMMENT state에서는 \* 이 들어올 때 까지 문자열을 받는다. \* 이 들어오면 state를 INCOMMENT\_로 바꾼다. INCOMMENT\_에서는 곧이어 / 라는 문자열이 들어오는지 확인한 후 /\*\*/ 주석의 완성을 판단 한 후 state를 START로 바꾼다. / 문자열이 바로 안 들어올 경우 다시 state 를 INCOMMENT로 바꾸어 \* 를 기다린다.

**= or == 판단**

 state가 START 일 때 문자열에 = 가 들어오면 state를 INEQ로 바꾸고 다음 문자열을 받는다.



INEQ state에 =가 들어오면 == 인 것으로 판단 한 후 currentToken 을 EQ로 설정 한다. 다른 문자열이 들어오면 ungetNextChar()를 호출 후 = 인 것으로 판단 해서 currentToken을 ASSIGN으로 설정 한다. 토큰이 결정 되어 졌기 때문에 state는 DONE으로 설정한다.

**< or <=, > or >= 판단**



state가 START 일 때 문자열에 < 가 들어오면 state를 INLT로 바꾸고 다음 문자열을 받는다.



INLT state에 = 가 들어오면 <= 인 것으로 판단 한 후 currentToken 을 LE로 설정 한다. 다른 문자열이 들어오면 ungetNextChar()를 호출 후 < 인 것으로 판단 해서 currentToken을 LT로 설정 한다. >(GT) 와 >=(GE) 또한 이런 방법으로 판단한다. 토큰이 결정 되어 졌기 때문에 state는 DONE으로 설정한다.

**!= 판단**



state가 START 일 때 문자열에 ! 가 들어오면 state를 INNE로 바꾸고 다음 문자열을 받는다.



INNE state에 = 가 들어오면 != 인 것으로 판단 한 후 currentToken 을 NE로 설정 한다. 다른 문자열이 들어오면 ungetNextChar()를 호출 후 error 인 것으로 판단 해서 currentToken을 ERROR로 설정 한다. 토큰이 결정 되어 졌기 때문에 state는 DONE으로 설정한다.

**나머지 토큰 판단**



나머지 토큰은 state가 START 일 때 switch case 문을 이용하여 들어오는 문자열에 따라 토큰 설정을 한다.

**Makefile** – Makefile 첨부

주어진 Makefile을 수정 없이 실행 시켜도 된다. make tiny 명령어를 이용하여 tiny 파일을 만든다.

# Implementation of C-Scanner using lex(flex) by Tiny.l modification

**cimus.l**

getToken()함수에 yylex() 함수를 이용하여 rule section 의 문자열을 비교하여 토큰을 리턴하여 currentToken을 설정한다.



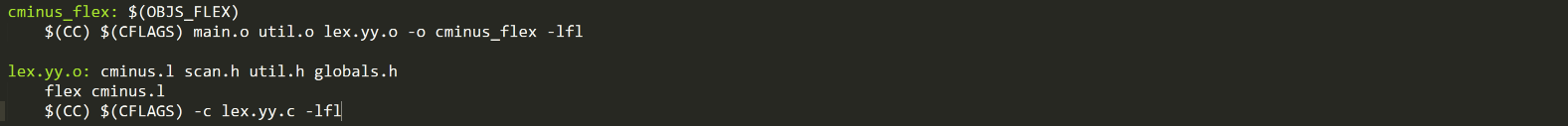
rule section 에 새로 세팅 할 스트링과 그에 따른 토큰타입을 설정한다.

**/\*\*/ 판단**



문자열에 /\* 가 들어올 경우 \*/ 이 들어오기를 기다린다. \*/ 는 두개의 문자가 연속되어 있기 때문에 prev를 이용하여 (prev == “\*” && c == ‘/’) 가 만족 할 때 /\* \*/ 의 완성을 판단한다.

**Makefile** – Makefile 첨부



flex 를 이용하여 cminus.l 로 lex.yy.c 을 만든다. 그 후 gcc 명령어를 이용하여 cminus\_flex 파일을 만든다. 이때 -lfl 옵션을 추가하여 cminus.l 이 컴파일 되도록 한다. 그 후 $make cminus\_flex 명령어를 이용하여 cminuns\_flex 파일을 만든다. 이때 Makefile 상단에 OBJS\_FLEX = main.o util.o lex.yy.o 를 추가한다.

# Compilation environment

**Os** : Ubuntu 16.04.3 LTS

**gcc** : gcc (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609

# Example and Result

**Example** : ./result/gcd.cm, ./result/sort.cm 첨부

|  |  |
| --- | --- |
| gcd.cm | sort.cm |
|  |  |

**Result** : ./result/gcd\_with\_tiny, ./result/gcd\_with\_cminus, ./result/sort\_with\_tiny, ./result/sort\_with\_cminus 첨부

|  |  |  |  |
| --- | --- | --- | --- |
| gcd\_with\_tiny | gcd\_with\_cminus | sort\_with\_tiny | sort\_with\_cminus |
|  |  |  |  |