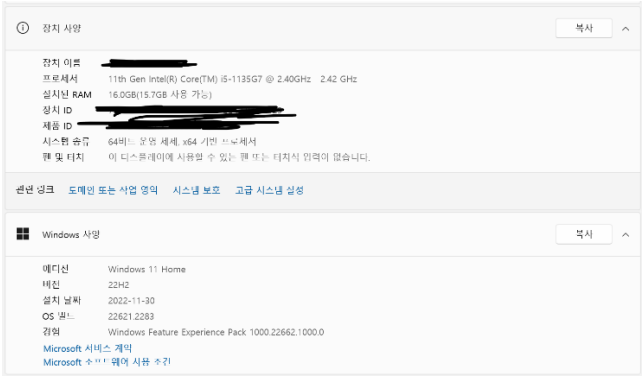
# 컴파일러설계 1_Scanner

**2019082279 김영현**

## 1. Compilation environment and method




윈도우에서 WSL(리눅스용 윈도우 하위 시스템) 사용, vscode로 작업, ubuntu 20.04

```
kyh011@DESKTOP-944FOPD:/mnt/c/Users/KYH/StudentID-2/StudentID$ ls
1_Scanner  execute..txt
kyh011@DESKTOP-944FOPD:/mnt/c/Users/KYH/StudentID-2/StudentID$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.6 LTS
Release:        20.04
Codename:       focal
```

make clean -> make all -> ./cminus_cimpl ./example/test.cm

./cminus_lex ./example/test.cm

## 2. Brief explanations about how to implement and how it operates

### A. C code

Scan.c 와 util.c를 수정했다. 양식에 맞게 reserved words는 if, else, while, return, int, void 사용했다. Symbol들도 scan.c와 util.c에 추가했다.

전체적인 구조는 변경하지 않았다. Main.c에서 scan.c의 getToken()을 호출하여 getToken()에서 ENDFILE을 return하기 전까지 계속 반복하는 구조이다. 또한 getToken()에서는 state가 DONE이 아니라면 다음 글자를 계속 읽고 save flag를 이용하여 저장할 token들은 tokenString에 하나씩 저장하는 방식을 사용한다. 그러다 state가 DONE이 되면 tokenString에 '₩0'을 넣고 tokentype을 표시하는 currentToken이 ID이면 reservedwords인지 확인하고 이후 tokenString과 currentToken을 util.c의 printToken에 보내서 출력한다. 이러한 방식으로 DFA를 구현한다. 그래서 기존에 존재하던 함수들을 이용하여 양식에 맞게 변형, 추가하였다. Scan.c에서 getToken에 currentToken의 token type 종류를 늘렸다.

또한 util.c에서 그렇게 늘린 token type에 맞게 printToken도 수정하였다.

getToken에서 START인 상태에서 =이 나오면 state를 INEQ로 설정한다. 이후 다음 루프에서 INEQ case에 들어가고 또 =이 나오면 state를 DONE으로 하고 currentToken을 EQ로 설정한다. Save flag의 경우 기본이 true이기 때문에 이후 printToken을 호출해서 출력한다. =이 나오지 않으면 assign으로 판단한다.

START에서 /이 나오면 우선 INOVER state로 설정한다. 이후 다음 글자를 읽고 *이 나오면 주석으로 판단한다. 아니라면 ungetNextChar를 호출하여 다시 읽은 글자를 복구하고 currentToken을 OVER로 판단하여 DONE state로 정한다. 주석일 경우, state를 INCOMMENT로 설정한다. INCOMMENT는 /* 이후 주석 내부라고 판단했고 INCOMMENT state에서는 다음 글자가 EOF이면 currentToken을 ENDFILE로 설정하고 아니면 *이 나오면 INCOMMENT_ state로 설정한다. INCOMMENT_ state에서는 다음 글자가 /이면 주석을 종료시키고 state를 다시 START로 설정한다. 아니면 여전히 주석 내부라 판단하여 다시 INCOMMENT state로 보낸다.

START에서 >가 나오면 state를 일단 INGT로 설정한다. 이후 다음 글자가 =이면 >=이라 판단하여 currentToken을 GE로 설정한다. 그렇지 않을 경우 다시 읽은 글자는 복구하고 currentToken을 GT로 설정한다. <도 동일한 논리로 해결했다.

START에서 !가 나오면 INNE state로 설정한다. INNE에서 다음 글자가 =이면 !=이라 판단하여 currentToken을 NE로 설정한다. 그게 아니라면 다시 글자를 복구하고 양식에선 !에 관한 내용이 없으므로 ERROR로 설정한다.

ID의 경우 첫글자가 letter인지 체크하고 INID로 보낸 후 이후 letter나 digit이 와도 되도록 수정했다.

**B. Lex**

Lex는 cminus.l을 수정했다.

Identifier의 형식을 양식에 맞게 맞춰줬다. 주석일 경우 도출하는 논리를 c code로 해결한 방식과 동일하게 접근했다. /가 나오고 그 다음이 *이면 주석이고 아니면 OVER로 판단한다. 주석 내부에서는 글자를 읽어나가며 *이후 /이 나오면 break하고 EOF나 '₩0'이 나올 경우 break한다.

## 3. Examples and corresponding result screenshots

C code와 lex 모두 test, test2, test3, test4, test5를 통과했다.