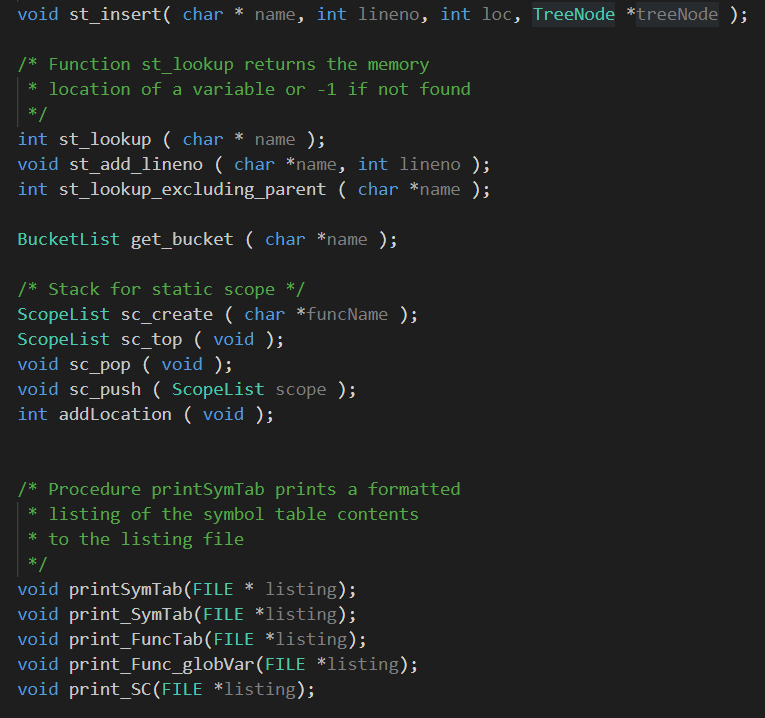
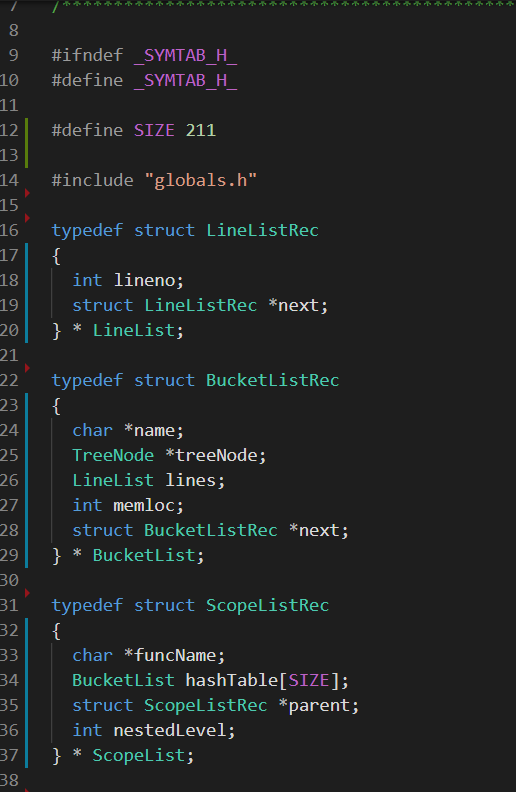
Project #3. Semantic Report

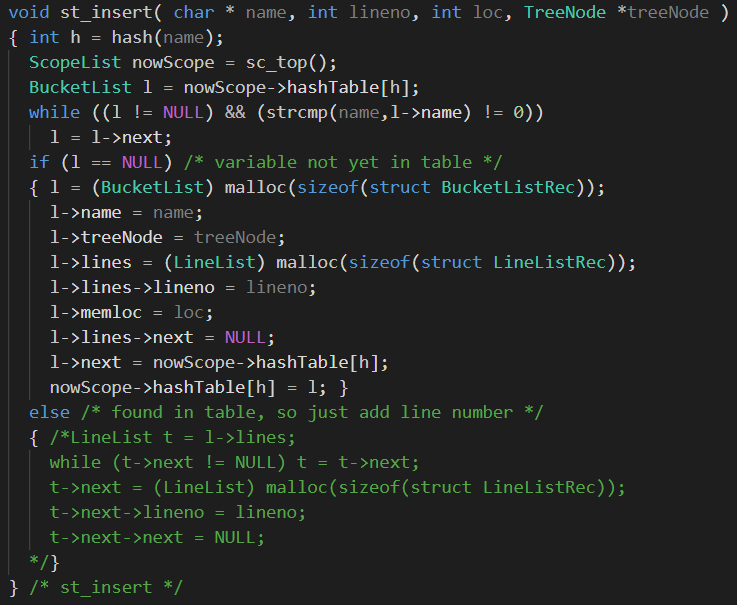
2017030328. 조지훈

**Compilation method and environment  
사용 에디터: Visual Studio Code & VIM editor**

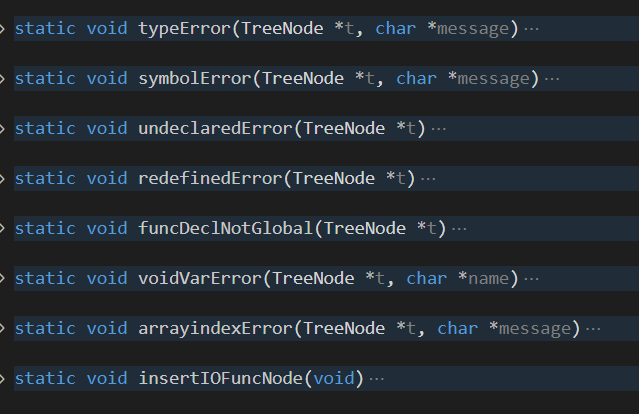
**개발 OS: Ubuntu 20.04.1 LTS (Windows 10 PRO 21H1버전에서 WSL2 이용)  
컴파일러: gcc 7.5.0  
장치 사양  
프로세서 Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz   
RAM 16.0GB  
시스템 종류 64비트 운영 체제, x64 기반 프로세서**

**Explanation about how to implement how to operate.**

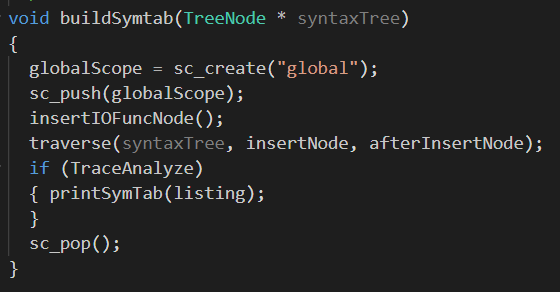
**symtab.h & symtab.c****c-minus에 맞게 BucketListRec와 hashTable을 변경.  
hashTable은 scope를 만드는데 이용하고, 계층구조를 만들기 위해 parent를 이용. scope끼리의 구분을 위해 funcName을 추가.**

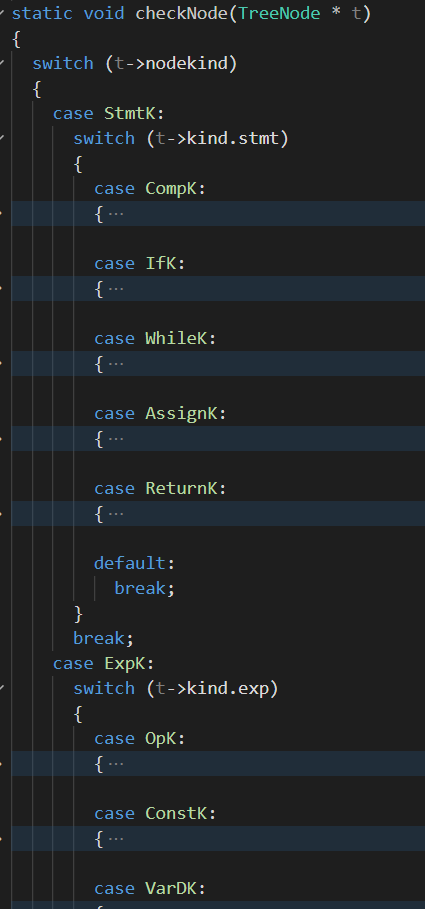
**scope를 찾고 생성하기 위해 scope와 관련되 함수들을 추가. 또한, scope를 스택의 형태로 구현.  
전체에서 해당 이름이 있는지 확인하는 st\_lookup과 해당 scope에서 확인하는 st\_lookup\_excluding\_parent으로 확장. 또한, 과제 명세서 대로 table를** **출력하기 위해서 4개의 출력 함수를 추가.**

**st\_insert에서는 변수를 table의 추가하는 역할만 진행하고 중복되는 변수의 line number는 더 이상 저장하지 않음.  
(st\_add\_lineno로 따로 구현)   
각각의 print 함수들은 명세서 대로 출력할 수 있도록 구현.**

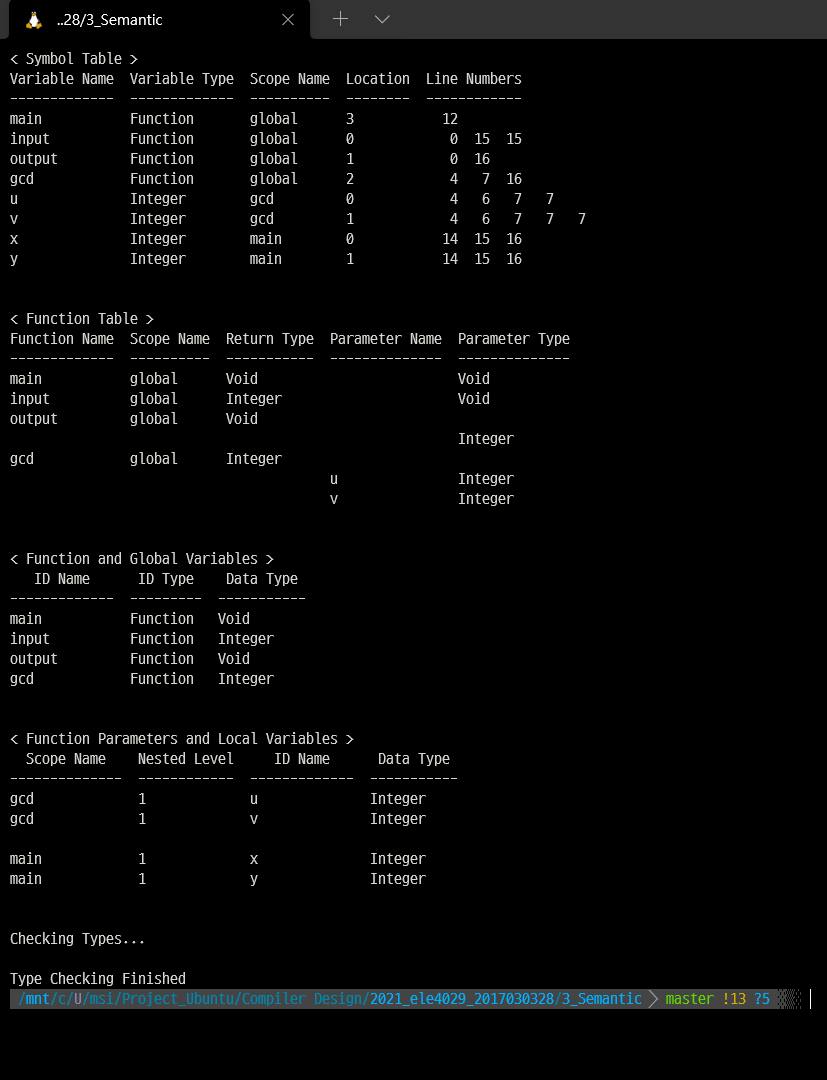
**analyze.c  
analyze는 analyze.h 파일을 건드리지 않고 analyze.c만 수정, 변경.**

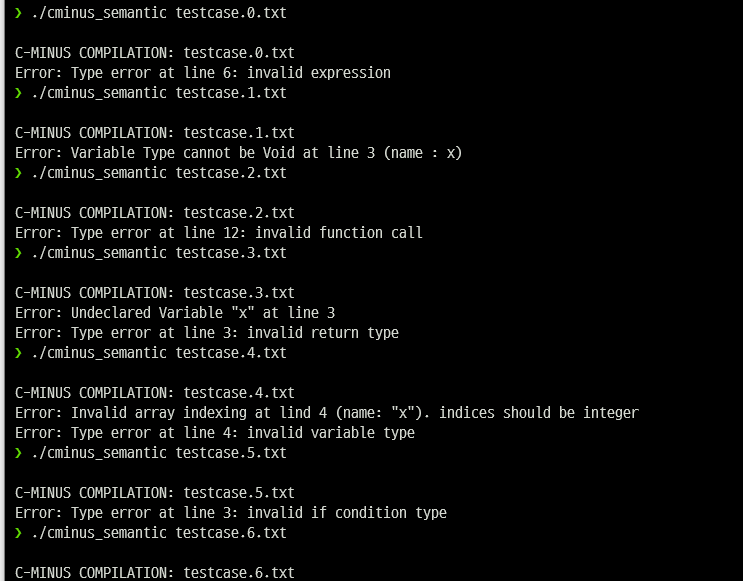
각각의 함수는 각각의 에러를 출력하는 함수, 상황에 맞아 해당 함수가 출력된다면 에러에 정해진 출력값을 출력.   
**insertIOFuncNode는 초기 글로벌 scope에 input, output 함수를 추가하기 위한 초기화 함수.**

**table을 만들기 시작할 때 global scope를 우선적으로 생성. 그 이후 syntaxTree를 따라 table를 작성.**

**checkNode에서는** **Type Check를 담당. Symbol Table의 정보와 각각 syntaxTree의 node를 이용해 에러가 있는지 확인.**

**각각의 노드의 kind에 따라 생성될 수 있는 오류를 확인하여 해당 오류가 발생 시 구현해 둔 에러 함수를 호출하여 에러 메시지를 출력하게 구현.**

**Example and Result Screenshot** **test.1.txt 결과 명세서와 같은 형식으로 출력이 되는 것을 확인.  
예시로 나온 결과와 비교를 하였을 때 output의 paramter가 조금 다르게 출력이 되기는 했지만 명세서와 동일한 출력을 하기 위해 유지.**



**각각의 testcase.0.txt ~ testcase.6.txt까지의 결과.  
실제 test case들은 ‘test\_case’에 첨부하였음.  
결과는 조금 다르지만 과제 명세서의 에러를 모두 탐지하는 것을 확인.**