Compiler Project  
-semantic-

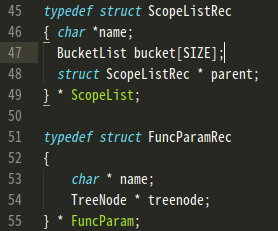
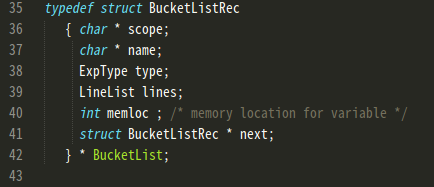
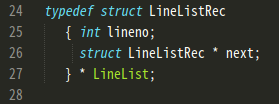
컴퓨터전공

2012003330 김선욱

1. 목적

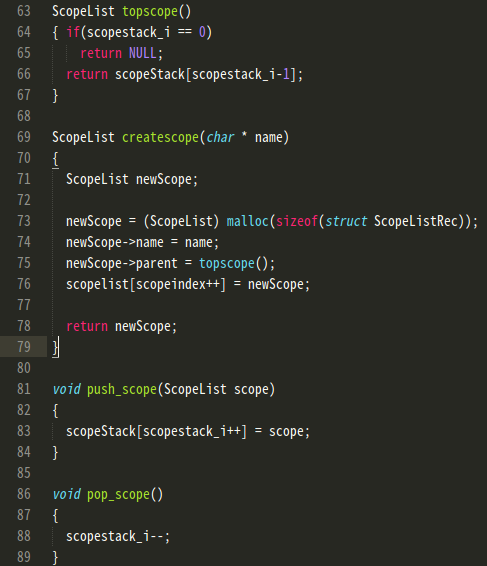
C-minus컴파일러의 문법(Semantic) 부분에 대한 구현을 실습하며 익힌다.

1. 환경
   1. OS : Ubuntu 16.04.1-desktop-amd64
   2. Compiler: cc 5.4.0
   3. Yacc/bison (GNU Bison) 3.0.4
   4. Makefile을 통해 컴파일
   5. Sublime Editor
   6. 예제파일 “simtest.cm”
2. Tiny Compiler의 코드 수정
   1. Symtab
      1. 사용되는 구조체



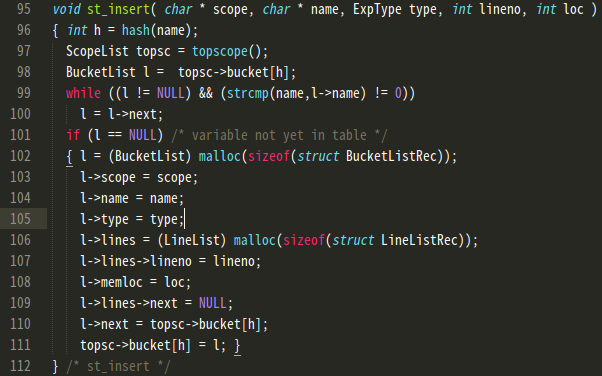
LineListRec : 호출번호를 저장하는 리스트 형태의 구조체  
BucketListRec : 선언 되는 요소들을 저장하기 위한 구조체  
ScopeListRec : Compound Statement에 따라 해당 Scope를 설정하기 위한 구조체  
FuncParamRec : 함수 호출 시 해당 함수에 대한 트리를 저장하는 구조체(파라미터 확인에 사용)

* + 1. Scope 관련 함수

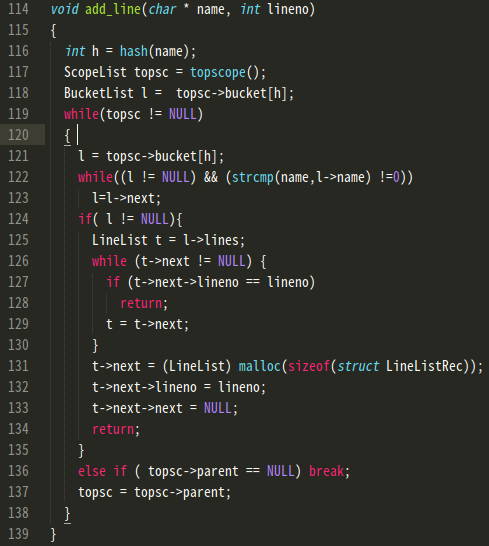


Scope를 생성하고 스택에 푸시 및 팝을 하는 부분으로   
 - topscope() : 가장 상단의 scope정보를 제공  
 - createscope() : Scope를 생성 하고 반환  
 - push\_scope() : Scope를 스택에 push  
 - pop\_scope() : Scope를 스택에서 pop

* + 1. St\_insert()

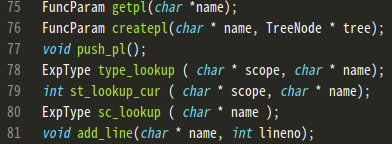


기존 로직과 동일하다. 하단 라인만 추가하던 else 부분은 다른 함수로 분리.

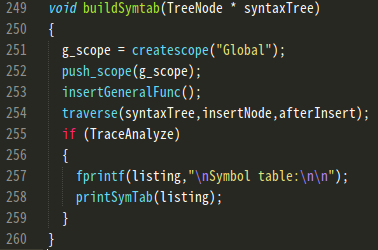


이미 존재하는 변수인지 확인되면 계속 상위 scope를 따라가며 해당 변수를 찾고 라인만 추가한다.

* + 1. 기타 사용되는 함수들



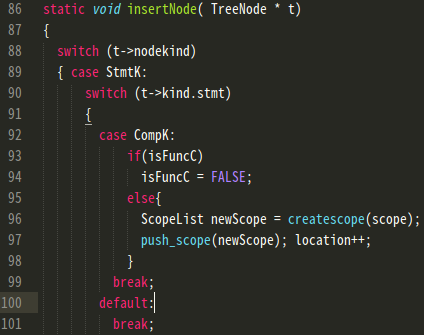
* getpl, createpl, push\_pl 함수들의 경우 위에 Scope와 동일하게 동작하며 함수호출에 대한 treenode를 가지고 있는 구조체이다.
* Type\_lookup : 해당 이름의 자료형 타입을 반환한다.
* St\_lookup\_cur : 현재 scope에서 선언된 변수인지 확인한다.
* Sc\_lookup : 해당 Scope를 찾는다.
* Add\_line : 이미 존재하는 이름일 경우 찾아서 라인만 추가한다.
  1. Analyze



SymbolTable을 만드는데 진입점이 되는 함수로 travserse 하기전에 “Global” scope를 설정하고 inserGeneralFunc()를 통해 “input, output” 함수 두가지를 정의해 추가해준다.

traverse함수 호출 시 기존 nullProc을 호출하던 부분에 Scope를 벗어나며 nested level를 낮추기 위해 afterInsert 함수를 생성해 바꿔준다.

1. Insertnode



Statement 부분에 대해 Compound에서 호출된 함수에 의한 Scope 추가인지를 isFuncC값을 통해 확인하고 아닐 경우 새로운 스코프를 추가하고 location값을 증가해 nested level를 조정한다.

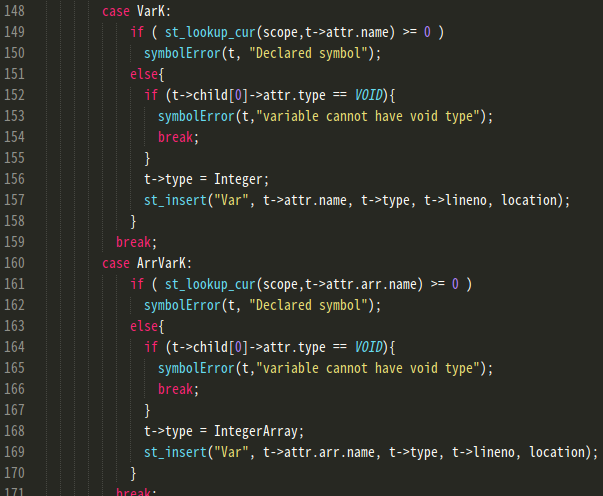


Expression에 대해서 변수나 함수 호출인 경우 호출되어 있는지 확인하고 라인만 추가해준다.

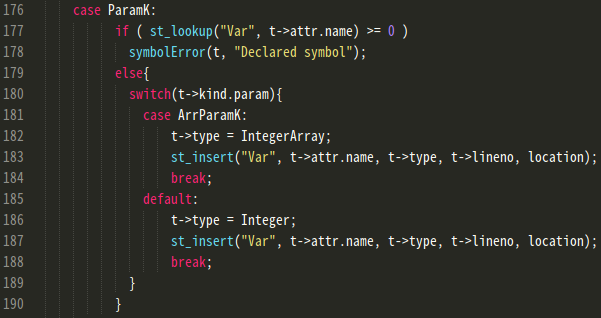
Assign시 배열을 일반 interger처럼 사용하는 경우 에러 발생.



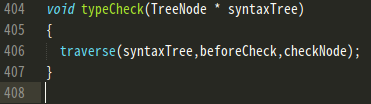
함수 호출시에 대해 이미 존재하는 함수인지 확인하고 아닐 경우 타입을 정해주고 심볼테이블에 추가, scope 설정해준다.



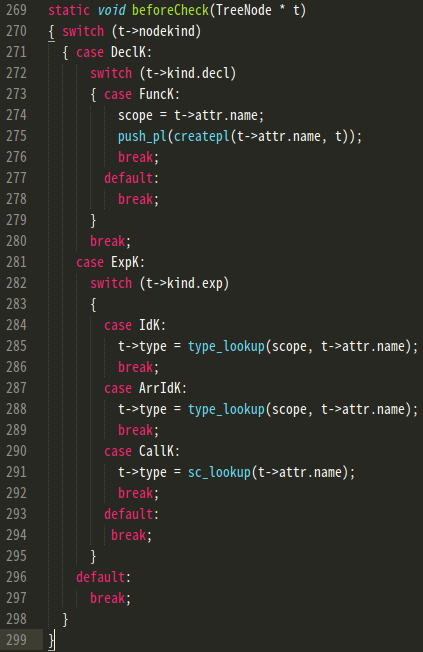
변수 설정에 경우 이미 있는 변수인지 확인한 후 심볼 테이블에 넣어준다.

파라미터에 경우에도 사용가능한 이름인지 확인한후 심볼테이블에 넣어준다.

* 1. Typecheck



nullProc 대신에 타입확인에 필요한 사전 함수인 beforeCheck를 넣어준다.



함수에 진입시 scope를 잡아주고 해당 함수의 treenode를 구조체에 저장한다.

각 변수 및 함수 호출 부분에 대해 타입을 설정해준다.

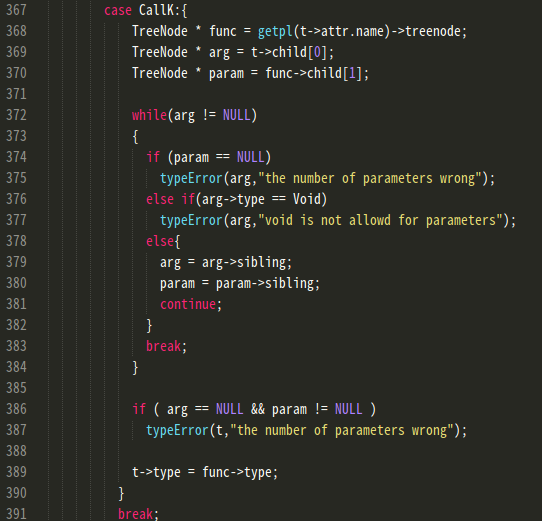
* 1. CheckNode



* IF, WHILE 문에 대해 인자 값이 비어 있는 경우 에러 출력한다.
* RETURN시 해당 함수의 반환 타입과 동일한지 확인한다.



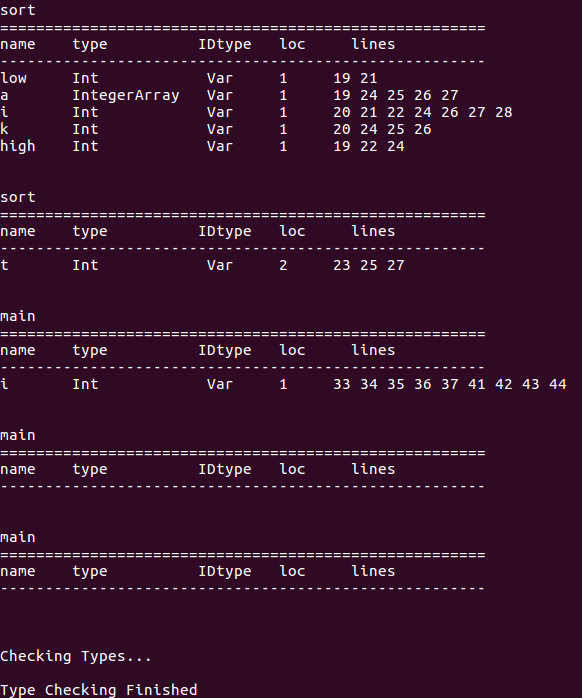
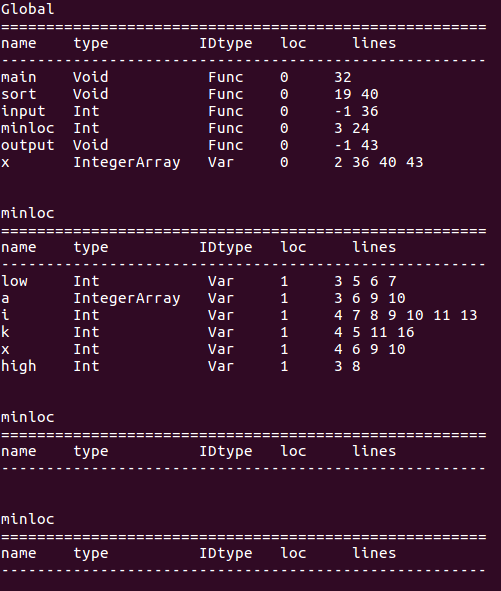
* ASSIGN에 경우 좌우 타입을 확인후 한쪽이 VOID일 경우 에러를 발생시킨다.
* 타입 값을 계속 좌측으로 넘겨줘 확인 할 수 있도록 한다.
* OP 비교에 대해서는 VOID는 비교할 수 없으므로 에러를 출력한다.
* 상수에 대해서는 정수타입으로 설정한다.



* 함수 호출 시 해당 함수의 인자 값 설정과 동일 한지 확인하여 파라미터 수 등이 다르면 에러를 출력한다.

1. 실행결과

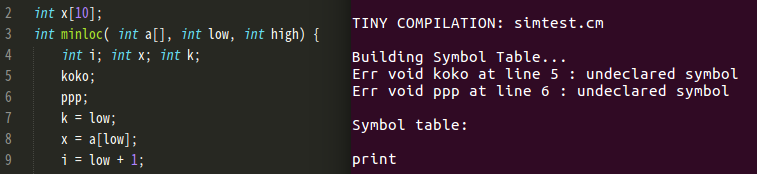
주어진 예제 코드 실행시



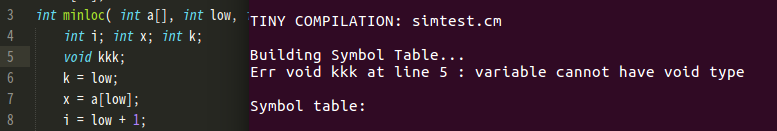
Nested level에 맞추어 선언된 순서가 잘 나타나는 것을 확인할 수 있고 테이블의 loc 값으로 nested level를 확인 할 수 있다.  
(0 = Global,시작 이후 Compound 마다 증가)

* 예외처리

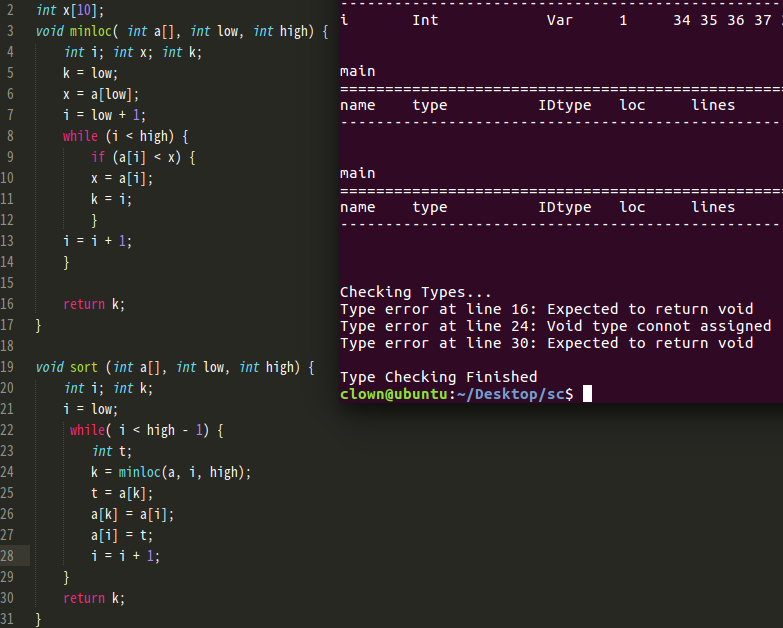
1. Undeclared Symbol



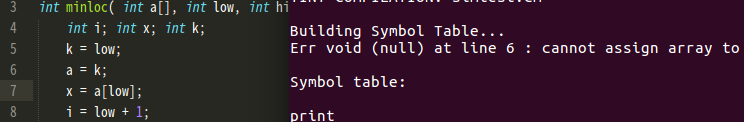
1. Void variable



1. Return type



1. Assignment Expression



1. Function Parameter

