컴파일러설계 3_Semantic_Analysis

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
```

chmod +x testcase_result.sh

./testcase_result.sh

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

analyze.c를 수정했다. cminus.y와 global.h, symtab.c, symtab.h, analyze.h 및 pdf를 참고했다.

A. analyze.c-insertNode

먼저 main.c를 다시 보고 symtab.h, symtab.c, analyze.h, analyze.c를 봤고, 작업하면서 cminus.y를 다시 봤다.

```
case Params:
```

insertNode의 params case의 경우 variable declaration case를 참고했다. 왜냐하면 어차피 parameter도 local variable과 유사하게 동작하기 때문이다. 또한 cminus.y의 내용을 보면 variable과 동일함을 알 수 있다.

```
case VarAccessExpr:
```

varAccessExpr의 경우 callExpr을 참고했다. 변수에 접근하는 것과 call해서 함수에 접근

하는 것과 유사하다고 판단했기 때문이다. 다만 차이점은 variable에 access할 경우, scope를 currentScope로 생각해야 한다는 것이었다.

B. analyze.c-checkNode

다음은 checkNode함수이다.

```
// If/If-Else, While Statement
case IfStmt:
case WhileStmt:
{
```

Pdf에 나와있는 조건대로, ifStmt와 WhileStmt는 둘 다 조건문에 integer 값이 들어와야 한다는 공통점이 있다. 따라서 해당 자리는 cminus.l에 의하면 child[0]의 자리이며, child[0]의 타입만을 체크해주면 된다. Type이 다르면 error함수를 호출한다.

```
// Return Statement
case ReturnStmt:
{
```

ReturnStmt는 함수 타입과 리턴 타입이 같은가를 체크해야 했다. 하지만 return stmt 자체는 함수 내부에 있기 때문에 scope를 거슬러 올라가서 바깥 함수 scope로 가서야 타입이 같은지 확인할 수 있다. 그렇기 때문에 currentScope값을 tmpScope에 저장하여 tmpScope의 parent가 globalScope가 될때까지 tmpScope에 tmpScope의 parent값을 넣었다. 왜 굳이 parent인가는 symtab.h를 보고 판단했다. 처음엔 globalScope까지 가도록 거슬러 올라갔지만 그것이 잘못된 방법임을 나중에 알았다. globalScope의 경우 가장 바깥쪽 scope이고 globalScope는 해당 리턴 스테이트를 가지고 있는 함수 스코프가 아니다. 단지 globalScope의 symbolList에 해당 함수 심볼이 존재할 뿐이다. 그렇기 때문에 globalScope의 직전 scope까지만 간다면 찾고자 하는 함수일 것이라 판단했다.

이후, cminus.l을 바탕으로 RETURN expression SEMI와 RETURN SEMI의 경우에 맞게 나눠서 문제를 해결했다.

```
// Assignment, Binary Operator Expression
case AssignExpr:
case BinOpExpr:
{
```

Assign의 경우 LHS와 RHS의 type이 같아야 했다. 또한 BinOpExpr의 경우 LHS와 RHS의 type이 같으며 둘 다 integer이어야 했다. 그 조건에 맞게 cminus.l과 globals.h를 참고해서 작성했다.

```
// Call Expression
case CallExpr:
{
```

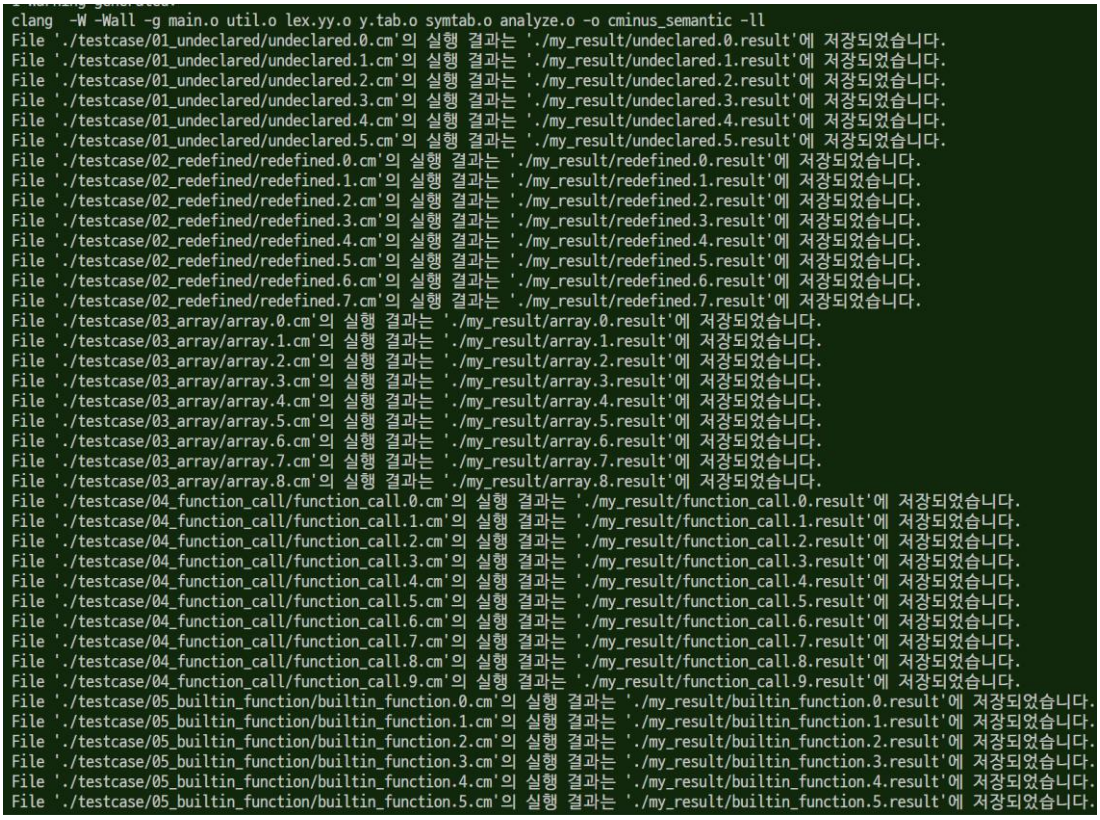
제시된 paramNode와 argNode를 바탕으로 cminus.l을 보고 작성했다. argNode가 NULL인 경우와 아닌 경우를 나누고, argNode가 NULL이 아닌 경우, 둘 중 하나가 NULL일때일 때 paramNode와 argNode의 sibling을 건너건너 타입 체크를 해준다. 이후, 둘 중 하나만 NULL이라면 개수에 맞지 않으므로 이것도 에러 처리해준다.

```
// Variable Access
case VarAccessExpr:
{
```

Variable이 integerArray 인지, integer인지 판단한다. 만약 integerArray의 경우, cminus.l에 의하면 child[0]에 index가 들어가므로 해당 index의 type또한 체크해준다.

3. Examples and corresponding result screenshots

모든 테스트 예시 결과와 동일하게 my result가 만들어졌으며 result결과도 모두 동일함을 체크했다.



```
File './testcase/06_return/return.0.cm'의 실행 결과는 './my_result/return.0.result'에 저장되었습니다.
File './testcase/06_return/return.1.cm'의 실행 결과는 './my_result/return.1.result'에 저장되었습니다.
File './testcase/06_return/return.2.cm'의 실행 결과는 './my_result/return.2.result'에 저장되었습니다.
File './testcase/06_return/return.3.cm'의 실행 결과는 './my_result/return.3.result'에 저장되었습니다.
File './testcase/06_return/return.4.cm'의 실행 결과는 './my_result/return.4.result'에 저장되었습니다.
File './testcase/06_return/return.5.cm'의 실행 결과는 './my_result/return.5.result'에 저장되었습니다.
File './testcase/06_return/return.6.cm'의 실행 결과는 './my_result/return.6.result'에 저장되었습니다.
File './testcase/06_return/return.7.cm'의 실행 결과는 './my_result/return.7.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.0.cm'의 실행 결과는 './my_result/assignment.0.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.1.cm'의 실행 결과는 './my_result/assignment.1.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.2.cm'의 실행 결과는 './my_result/assignment.2.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.3.cm'의 실행 결과는 './my_result/assignment.3.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.4.cm'의 실행 결과는 './my_result/assignment.4.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.5.cm'의 실행 결과는 './my_result/assignment.5.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.6.cm'의 실행 결과는 './my_result/assignment.6.result'에 저장되었습니다.
File './testcase/07_assignment/assignment.7.cm'의 실행 결과는 './my_result/assignment.7.result'에 저장되었습니다.
File './testcase/08_operation/operation.0.cm'의 실행 결과는 './my_result/operation.0.result'에 저장되었습니다.
File './testcase/08_operation/operation.1.cm'의 실행 결과는 './my_result/operation.1.result'에 저장되었습니다.
File './testcase/08_operation/operation.2.cm'의 실행 결과는 './my_result/operation.2.result'에 저장되었습니다.
File './testcase/08_operation/operation.3.cm'의 실행 결과는 './my_result/operation.3.result'에 저장되었습니다.
File './testcase/08_operation/operation.4.cm'의 실행 결과는 './my_result/operation.4.result'에 저장되었습니다.
File './testcase/08_operation/operation.5.cm'의 실행 결과는 './my_result/operation.5.result'에 저장되었습니다.
File './testcase/09_parameter/parameter.0.cm'의 실행 결과는 './my_result/parameter.0.result'에 저장되었습니다.
File './testcase/09_parameter/parameter.1.cm'의 실행 결과는 './my_result/parameter.1.result'에 저장되었습니다.
File './testcase/09_parameter/parameter.2.cm'의 실행 결과는 './my_result/parameter.2.result'에 저장되었습니다.
File './testcase/09_parameter/parameter.3.cm'의 실행 결과는 './my_result/parameter.3.result'에 저장되었습니다.
File './testcase/09_parameter/parameter.4.cm'의 실행 결과는 './my_result/parameter.4.result'에 저장되었습니다.
File './testcase/09_parameter/parameter.5.cm'의 실행 결과는 './my_result/parameter.5.result'에 저장되었습니다.
File './testcase/10_void_variable/void_variable.0.cm'의 실행 결과는 './my_result/void_variable.0.result'에 저장되었습니다.
File './testcase/10_void_variable/void_variable.1.cm'의 실행 결과는 './my_result/void_variable.1.result'에 저장되었습니다.
File './testcase/10_void_variable/void_variable.2.cm'의 실행 결과는 './my_result/void_variable.2.result'에 저장되었습니다.
File './testcase/10_void_variable/void_variable.3.cm'의 실행 결과는 './my_result/void_variable.3.result'에 저장되었습니다.
File './testcase/10_void_variable/void_variable.4.cm'의 실행 결과는 './my_result/void_variable.4.result'에 저장되었습니다.
File './testcase/10_void_variable/void_variable.5.cm'의 실행 결과는 './my_result/void_variable.5.result'에 저장되었습니다.
File './testcase/11_condition/condition.0.cm'의 실행 결과는 './my_result/condition.0.result'에 저장되었습니다.
File './testcase/11_condition/condition.1.cm'의 실행 결과는 './my_result/condition.1.result'에 저장되었습니다.
File './testcase/11_condition/condition.2.cm'의 실행 결과는 './my_result/condition.2.result'에 저장되었습니다.
File './testcase/11_condition/condition.3.cm'의 실행 결과는 './my_result/condition.3.result'에 저장되었습니다.
File './testcase/11_condition/condition.4.cm'의 실행 결과는 './my_result/condition.4.result'에 저장되었습니다.
File './testcase/11_condition/condition.5.cm'의 실행 결과는 './my_result/condition.5.result'에 저장되었습니다.
File './testcase/12_scope/scope.0.cm'의 실행 결과는 './my_result/scope.0.result'에 저장되었습니다.
File './testcase/12_scope/scope.1.cm'의 실행 결과는 './my_result/scope.1.result'에 저장되었습니다.
File './testcase/12_scope/scope.2.cm'의 실행 결과는 './my_result/scope.2.result'에 저장되었습니다.
File './testcase/12_scope/scope.3.cm'의 실행 결과는 './my_result/scope.3.result'에 저장되었습니다.
File './testcase/12_scope/scope.4.cm'의 실행 결과는 './my_result/scope.4.result'에 저장되었습니다.
File './testcase/12_scope/scope.5.cm'의 실행 결과는 './my_result/scope.5.result'에 저장되었습니다.
```