컴파일러 설계

Project #2. Parser

Compilation method and Environment

> Ubuntu 16.04.1 LTS

gcc 컴파일러

Explanation about How to implement and How to operate

> bison을 이용해서 *y.tab.h*와 *y.tab.c*를 생성한 뒤 *makefile*에서 오브젝트 파일을 생성하여 기타 오브젝트 파일과 묶어서 실행파일 *cminus*를 생성했습니다.

생성된 *cminus*파일이 있는 **project**폴더 내에서 *./cminus 샘플파일명*을 입력하면 AST가 출력됩니다.

Explanation about modified code

1. main.c

PDF에 명시되어있는 것과 같이 Syntax Tree만을 출력하기 위해서 플래그들을 조정해주었습니다.

2. globals.h

기존 YACC폴더 내의 *globals.h*를 덮어썼고 *treenode*의 *attr*을 union에서 struct형식으로 수정해주었습니다.

또한 kind keyword를 필요에 맞게 추가해주었습니다.

3. util.h

---

*util.c*에서 사용하기 위해서 새로운 타입의 노드 *newDeclNode(DeclKind)*와 *newParamNode(ParamKind)*를

정의해주었습니다.

4. util.c

*util.h*에서 정의한 두 종류의 노드의 상세내용을 작성했습니다. 각각 선언문(변수,함수)과 parameter(single, array)를 표시하는 노드입니다.

또한 *printTree*를 수정해주어 AST를 appendix에 맞게 출력할 수 있도록 해주었습니다.

5. cminus.y

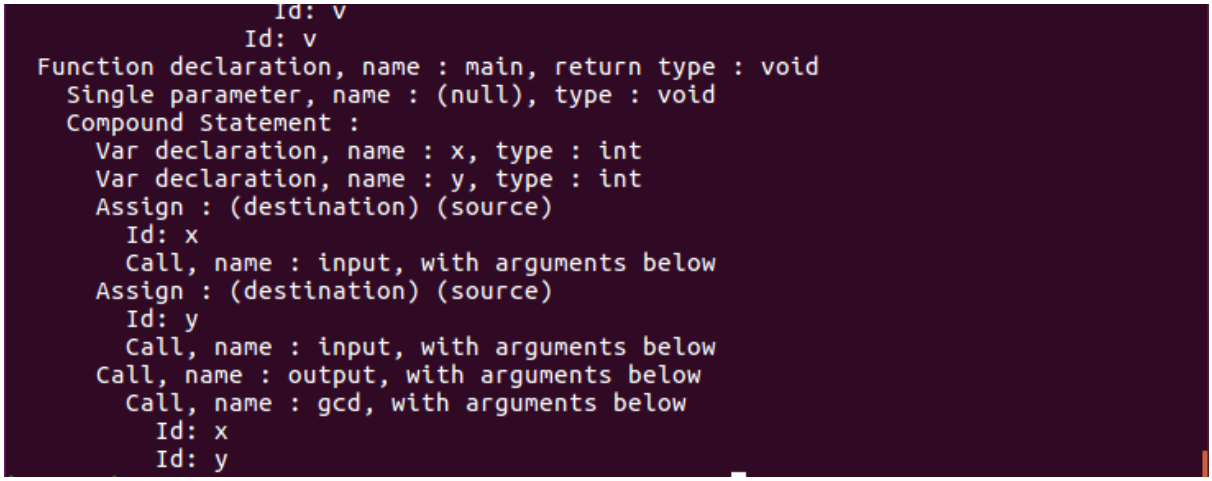
*tiny.y*를 기반으로 작성되었으며 편의를 위해서 몇 가지의 전역변수를 추가로 선언해주었습니다.

Lex에서 생성된 토큰들을 받아들일 수 있도록 token들을 추가해주었고, 생성규칙을 C- appendix에 맞게

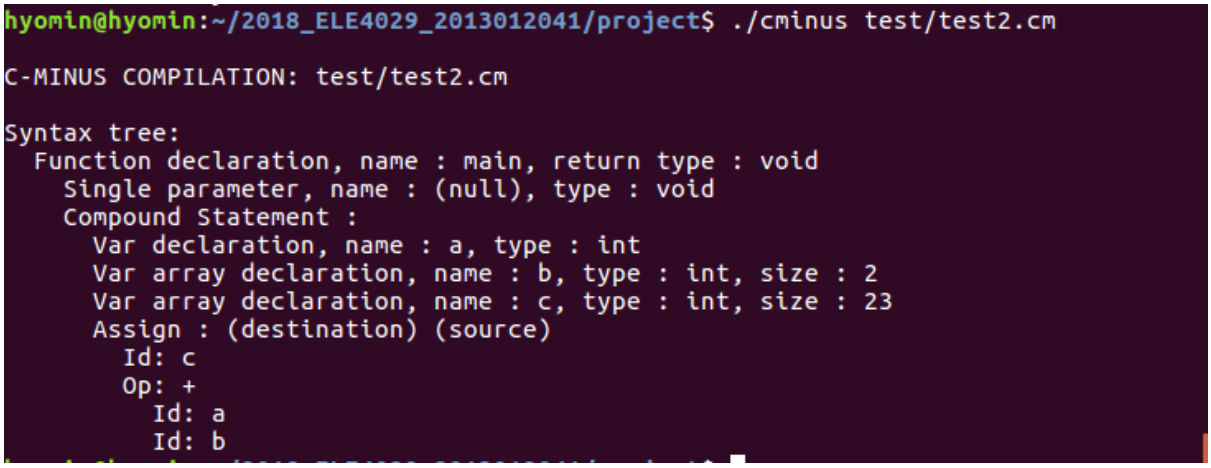
수정해주었습니다.

Result Screenshot

test1.cm



test2.cm


hyomin@hyomin:~/2018_ELE4029_2013012041/project$ ./cminus test/test2.cm
C-MINUS COMPILATION: test/test2.cm
Syntax tree:
Function declaration, name : main, return type : void
Single parameter, name : (null), type : void
Compound Statement :
Var declaration, name : a, type : int
Var array declaration, name : b, type : int, size : 2
Var array declaration, name : c, type : int, size : 23
Assign : (destination) (source)
Id: c
Op: +
Id: a
Id: b