2017029770

컴퓨터소프트웨어학부

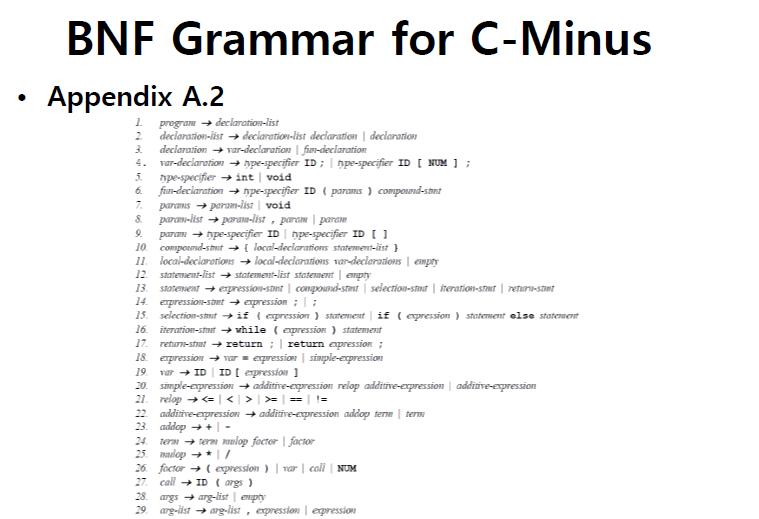
서윤희

**Project 2 Parser 과제**

실행환경: Ubuntu 18.04 (조교님 참고부탁드립니다! Project1에서 틀렸던 부분을 고치고 반영하면서cminus.l의 EOF 를 ‘\0’ null 캐릭터로 구현해두었습니다.)

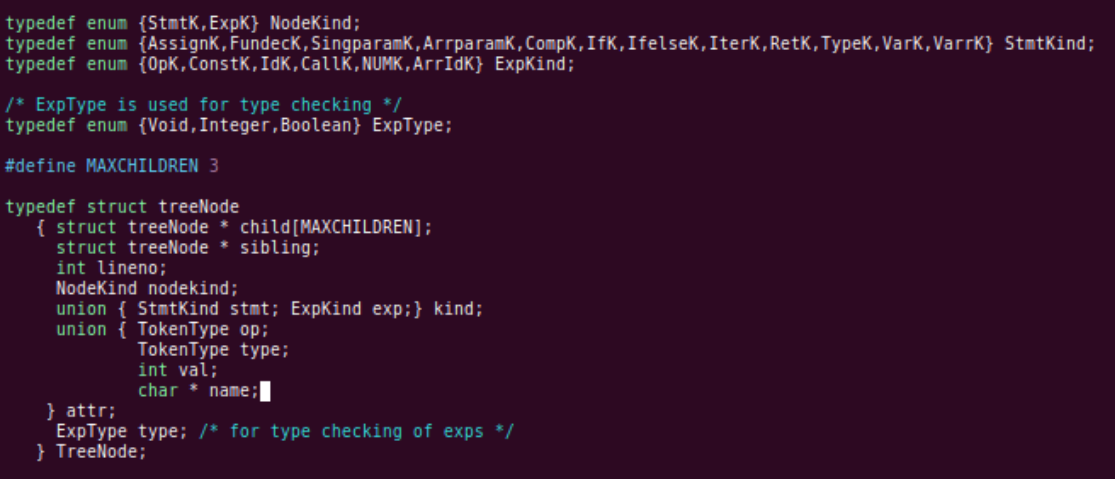
목적: C-mius 문법에 맞게 parsing 하는 parser 구현

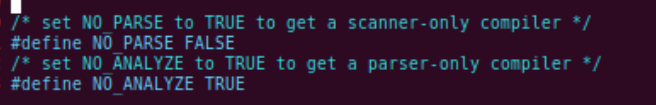
실행방법: make 후 생성된 executable file > cminus에 (파싱할 파일)을 인자로 넣어준다

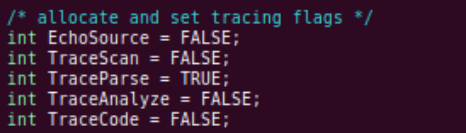


C-minus 문법에 해당하는 BNF와 과제에서 참고로 주신 Syntax tree definition을 따라 rule 생성(type이나 num, id등을 파싱하기 위한 rule이나, node의 추가가 있었습니다.)

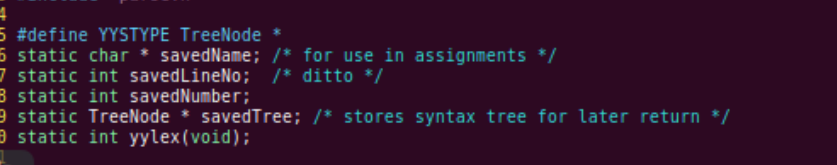
변경된 Code 부분

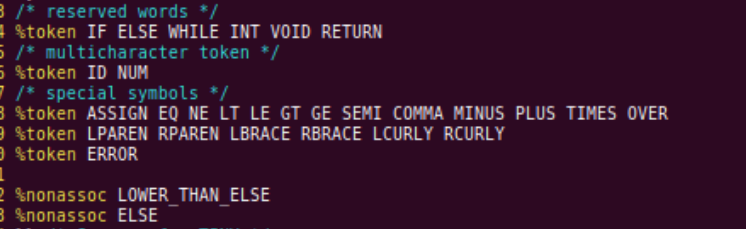
1. globals.h - Syntax tree와 rule을 참고하여 Derivation을 세분화하기 위해 필요한 노드 타입을 기존의 StmtKind, ExpKind에 추가, 사용되지 않는 노드 타입은 삭제. ExpType enum과 treeNode 형태는 기존에 있던 것을 그대로 사용. 
2. main.c – 과제 명세에 나와있듯, 일부 설정 flag값 변경



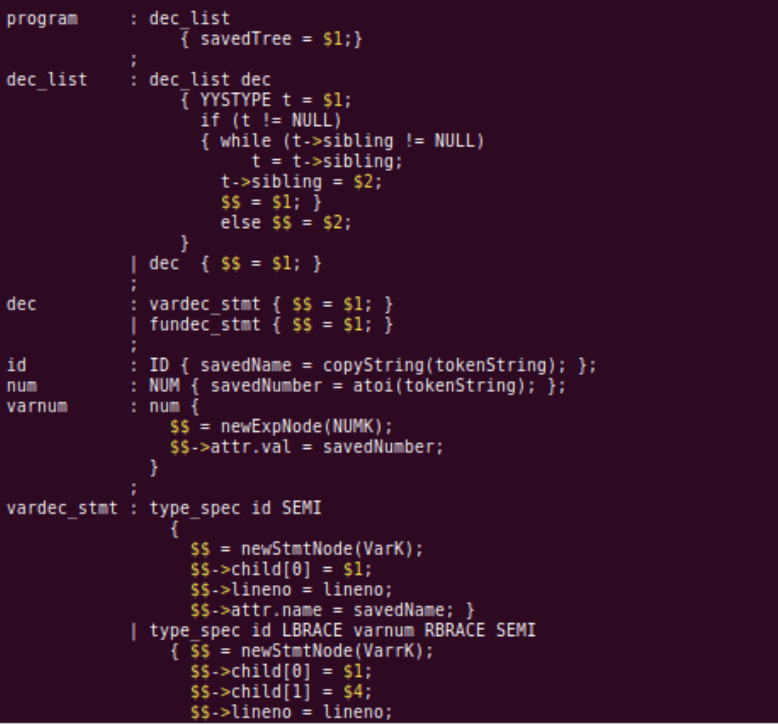


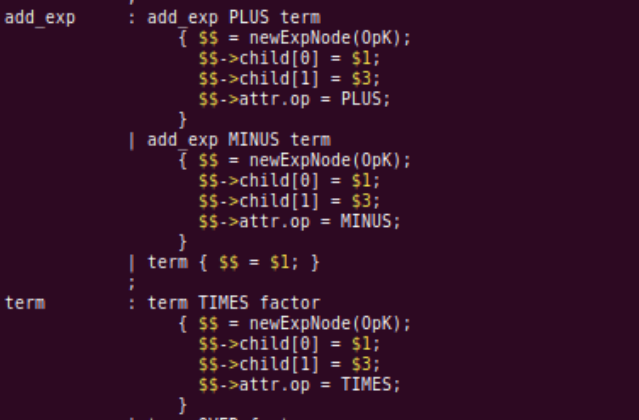
1. cminus.y – 과제 명세 참고하여 cminus grammar, syntax tree에 맞게 rule 정의. NUM에 해당하는 숫자를 담고있기 위해 savedNumber추가. 사용하지 않는 token은 지우고, cminus에서 사용되는 token 추가, if else문에서 else가 우선적으로 처리되기 위해 nonassoc LOWER\_THAN\_ELSE, ELSE를 사용하여 우선순위를 부여.





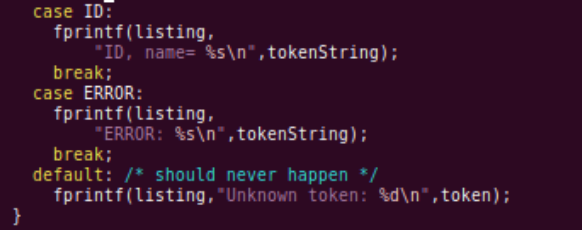
그 이후 cminus grammar, syntax tree에 따라 파싱룰 생성. 보고서 분량 관계상, 일부만 캡쳐하였습니다. 같은 stage에서 계속 파싱가능하면 반복적으로 derivation, 우선순위상 num, id를 먼저 파싱하고 token 인식해야 하는 경우를 처리하고, type에 관한 node 를 추가, conflict, 파싱에러가 생기지 않도록 필요한경우 rule들을 추가 및 변형하여 cminus.y에 구현하였습니다. 대부분은 cminus bnf grammar와 동일합니다. cminus대로 파싱이 되는 부분인 ConstK, OpK등 기존에 있던rule을 그대로 활용한 부분도 있습니다.(ex - WHILE문은 IterK 사용) 일부는 새로운 노드의 종류를 만들어 따로 담기도 했습니다. Ex) TypeK, VarrK, VarK 등

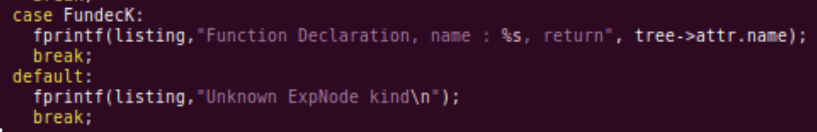
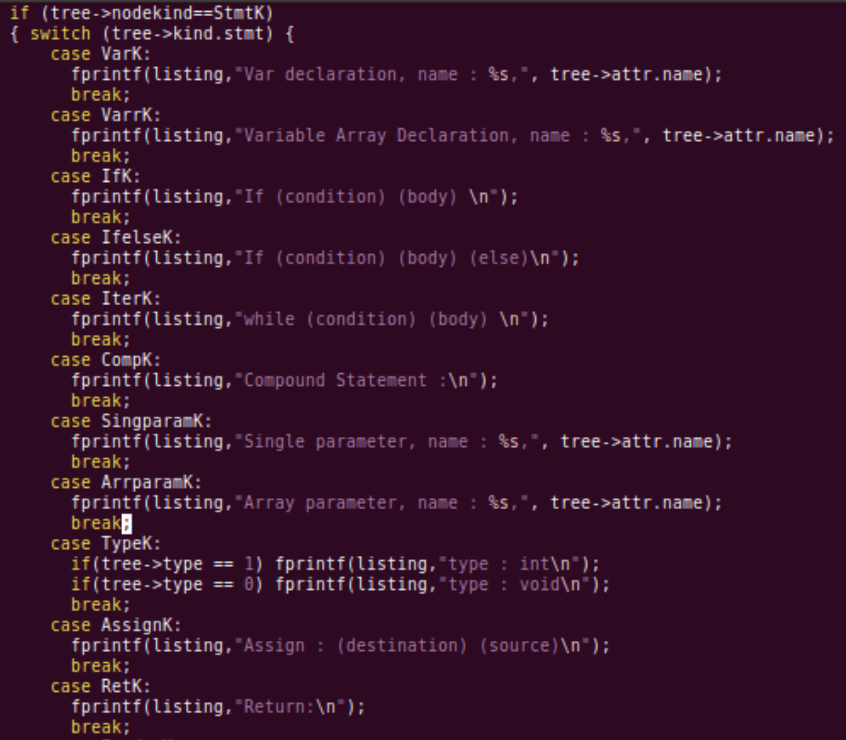




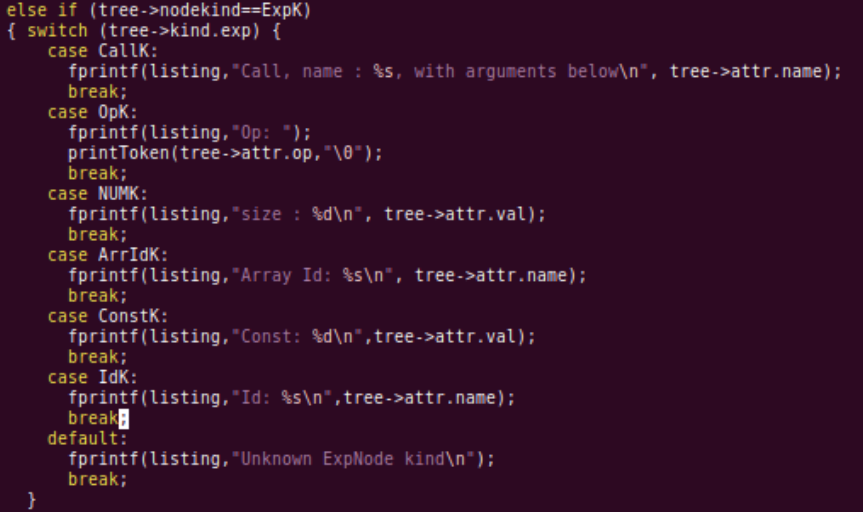
* 이렇게 여러 stage로 derivation해야 하는 부분을 줄여서 한번에 되도록 변형하기도 했습니다.

1. util.c – printToken함수 > parsing token에서 따로출력되어야 할 token들 출력형식 지정.

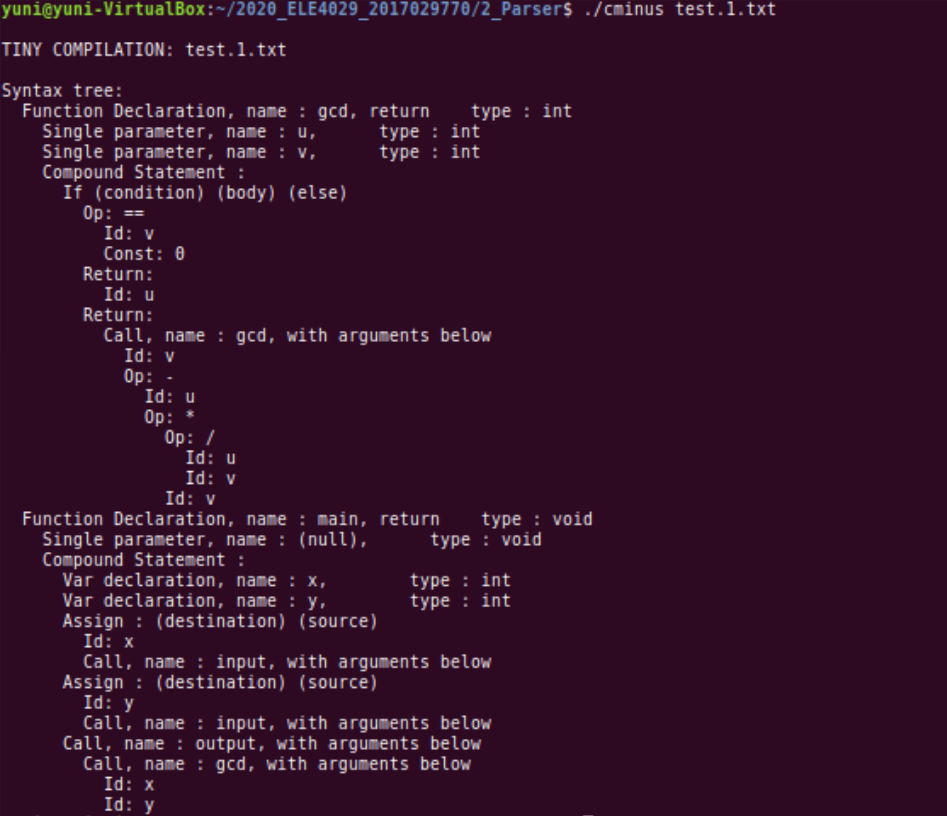




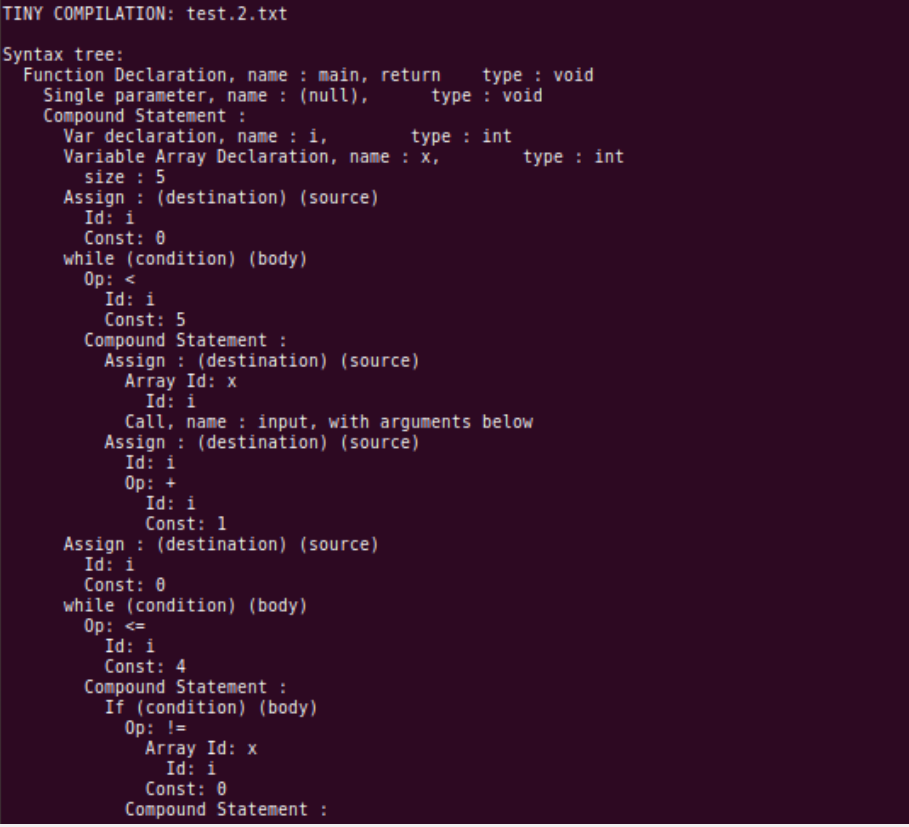
* StmtKind Tree Node 출력, 제가 추가한 treenode 종류들이 각 형식에 맞게 print 될 수 있도록 코드 추가. Type의 경우 기존에있던 enum 활용
* ExpKind 도 마찬가지, 특히 NUMK 타입의 노드는 attr.val에 var array declaration의 사이즈를 담도록 지정해두었음.(cminus.y 에 rule 지정해둠) 그래서 출력도 size : %d 형식으로 하게함.



Test.1.txt 실행결과



Test.2.txt 실행결과





Test.txt > 추가적으로 파싱 예제 직접 만들어 테스트

