Compiler Project2 Report

1. Compilation method and environment

   Ubuntu 환경에서 실행한다.

   주어진 makefile로 make 명령어를 실행하면 cminus_parser 실행파일이 생성되는데, ./cminus_parser [syntax analysis를 수행할 파일 이름] 을 실행하면 Parser의 실행 결과가 출력된다.

2. Explanation about how to implement and how to operate

   Cminus.y의 rules section에서 c minus 의 BNF 문법에 따라 rule들을 작성해주면 작성된 문법에 따라 sibling과 child를 생성하여 AST가 생성되고, 생성된 AST를 따라가며 util.c의 printTree 함수가 해당 함수의 내용을 output 포맷에 맞추어 출력해준다. Child로 연결된 노드들끼리는 print될 때 indent가 들어가고, sibling 노드들은 포함된 코드가 아닌 그냥 이어지는 코드를 의미하므로 indent가 들어가지 않고 같은 열 상에 출력된다.

3. Some explanation about the modified code

   Main.c) NO_PARSER 를 FALSE로, NO_ANALYZE를 TRUE로 변경한다.

   TraceScan을 FALSE로, TraceParse를 TRUE로 변경한다.

   Globals.h) yacc 디렉토리 내의 globals.h 파일로 기존 globals.h 파일을 대체한다.

   Cminus 명세에 맞게 Node의 종류를 추가하고 모든 노드를 NodeKind 타입으로 합친다.

   ExpType에서 Boolean은 cminus에서 사용하지 않으므로 제거한다.

   모든 노드가 하나의 타입으로 합쳐졌으므로 TreeNode에서 kind는 제거한다.

   TreeNode에 isArray라는 변수를 추가한다. 이는 해당 노드가 TypeK, ParamK나 VarK, VarDecK일 때 그 매개변수나 일반 변수의 타입이 배열인지 아닌지를 나타낸다.

   Util.c) newStmtNode와 newExpNode를 없애고 newNode라는 새로운 함수를 만든다.

   printTree 함수에서는 tree->kind에 따라 switch문을 실행하며 각 노드종류에 따라 출력되어야할 값들을 출력한다.

   Cminus.y) deifinition섹션에 cminus compiler의 token들을 추가해준다. C reference에 따라 token들의 priority와 associativity를 설정해주었고, dangling else problem의 conflict를 해결하기 위해 ELSE의 priority를 PAREN 보다 높게 설정해주고 associativity를 left로 설정해주었다.

   Rules 섹션에서는 C-Minus 의 BNF Grammar에 맞게 rule들을 작성해준다.

-_list 와 local_declaration의 경우에는 tiny.y의 stmt_seq 부분과 동일하게 sibling을 추가해주는 방식으로 구현한다. Type, val, op를 저장하는 방식은, type_specifier의 INT, VOID와 factor의 NUM, relop의 EQ, NE, LE, LT, GE, GT 에서 새 트리노드를 생성해 알맞은 값을 트리노드에 저장해두면 (NUM의 경우 tokenString에서 값을 가져옴), $$->attr.op = $1->attr.op 나 $$->attr.val = $1->attr.val, $$->type = $1->type과 같은 방식으로 그 값이 필요한 트리노드에 가져와서 저장하는 방식으로 구현했다.

Id의 경우 _id rule을 만들어서 ID 토큰이 나왔을 때, tokenString에서 copyString을 통해 ID 값을 attr.name에 저장해두고, ID 토큰이 사용되는 rule들에는 ID 토큰을 직접 사용하는 대신 _id rule을 사용하도록 하였다. 그리고 해당 rule이 reduce될 때, _id tree node의 attr.name에서 id 값을 가져와서 해당 rule의 포인터 ($$)의 attr.name에 저장하는 방식으로 구현하였다.

4. Example and Result Screenshot (input은 주어진 test파일 두개와 과제 명세 슬라이드의 Dangling Else Problem에 있는 예제입니다.)

```
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/2_Parser$ ./cminus_parser test.1.txt

C-MINUS COMPILATION: test.1.txt

Syntax tree:
  Function Declaration: name = gcd, return type = int
    Parameter: name = u, type = int
    Parameter: name = v, type = int
    Compound Statement:
      If-Else Statement:
        Op: ==
          Variable: name = v
          Const: 0
        Return Statement:
          Variable: name = u
        Return Statement:
          Call: function name = gcd
            Variable: name = v
            Op: -
              Variable: name = u
              Op: *
                Op: /
                  Variable: name = u
                  Variable: name = v
                Variable: name = v
  Function Declaration: name = main, return type = void
    Void Parameter
    Compound Statement:
      Variable Declaration: name = x, type = int
      Variable Declaration: name = y, type = int
      Assign:
        Variable: name = x
        Call: function name = input
      Assign:
        Variable: name = y
        Call: function name = input
      Call: function name = output
        Call: function name = gcd
          Variable: name = x
          Variable: name = y
```

```
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/2_Parser$ ./cminus_parser test.2.txt

C-MINUS COMPILATION: test.2.txt
Files
Syntax tree:
  Function Declaration: name = main, return type = void
    Void Parameter
    Compound Statement:
      Variable Declaration: name = i, type = int
      Variable Declaration: name = x, type = int[]
        Const: 5
      Assign:
        Variable: name = i
        Const: 0
      While Statement:
        Op: <
          Variable: name = i
          Const: 5
        Compound Statement:
          Assign:
            Variable: name = x
              Variable: name = i
            Call: function name = input
          Assign:
            Variable: name = i
            Op: +
              Variable: name = i
              Const: 1
      Assign:
        Variable: name = i
        Const: 0
      While Statement:
        Op: <=
          Variable: name = i
          Const: 4
        Compound Statement:
          If Statement:
            Op: !=
              Variable: name = x
                Variable: name = i
              Const: 0
            Compound Statement:
              Call: function name = output
                Variable: name = x
                  Variable: name = i
```

```
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/2_Parser$ ./cminus_parser test.3.txt

C-MINUS COMPILATION: test.3.txt

Syntax tree:
  Function Declaration: name = main, return type = void
    Void Parameter
    Compound Statement:
      If Statement:
        Op: <
          Variable: name = a
          Const: 0
        If-Else Statement:
          Op: >
            Variable: name = a
            Const: 3
          Assign:
            Variable: name = a
            Const: 3
          Assign:
            Variable: name = a
            Const: 4
```