컴파일러 #2

—— 1
Compilation method
make all

Usage
./cminus_flex <target_file>

Environment
ubuntu 18.04.1 LTS, x86_64
flex 2.6.4
bison 3.0.4
/* 로 시작하는 곳에서 input() 함수로 EOF가 들어오지 않는 부분이 있는데, 조교님 환경에 맞춰서 if(c == EOF) break; 로 넣어놨습니다.

—— 2
Explanation
- Commit 기록에 기반

1. yacc -d cminus.y 로 만든 y.tab.o를 사용하여 Makefile을 하도록 makefile 수정
2. cminus.y 에 필요없는 토큰을 지우고, 새롭게 적용시킬 토큰을 추가 (NE, LT 등등)
3. 기존 cminus.y에 있는 문법을 지우고 Appendix에 맞게 정의만 추가
   (ex program : declaration-list {/* empty */ };)
4. 정의만 해둔 상태로 make file을 시도 후 에러나는 부분을 수정 (analyze, cgen 토큰 삭제)
5. appendix 의 정의에서 - 가 bison에서 에러가 나서 _로 수정
6. 새롭게 필요한 Kind를 globals.h에 정의 (ex VarK, FunctionK ..)
7. id, num 값을 저장하는 문법 생성
8. syntax rule을 지정
9. scan.c 를 수정해서 tree print를 수정
10. 에러 수정

——3
modified code

Makefile의 OBJS에 y.tab.o를 추가하고, y.tab.o를 만든다

```
+ OBJS = y.tab.o lex.yy.o main.o util.o symtab.o analyze.o code.o cgen.o

  cminus: $(OBJS)
          $(CC) -o $@ $(CFLAGS) $(OBJS)
  @@ -39,25 +38,22 @@ code.o: code.c code.h globals.h
  cgen.o: cgen.c globals.h symtab.h code.h cgen.h
          $(CC) $(CFLAGS) -c cgen.c

+ y.tab.o: cminus.y globals.h
+         yacc -d cminus.y
+         $(CC) $(CFLAGS) -c y.tab.c
+
```

사용하는 토큰으로 yacc수정

```
- %token IF THEN ELSE END REPEAT UNTIL READ WRITE
- %token ID NUM
- %token ASSIGN EQ LT PLUS MINUS TIMES OVER LPAREN RPAREN SEMI
- %token ERROR
+ %token IF ELSE WHILE RETURN INT VOID
+ %token ID NUM
+ %token ASSIGN EQ NE LT LE GT GE PLUS MINUS TIMES OVER LPAREN RPAREN LBRACE RBRACE LCURLY
  RCURLY SEMI COMMA
+ %token ERROR ENDFILE
```

주석으로 Appendix 추가

```
-            ;
+ /* Appendix A.2 */
+ /* 1. program -> declaration-list */
+ /* 2. declaration-list -> declaration-list declaration | declaration */
+ /* 3. declaration -> var-declaration | fun-declaration */
+ /* 4. var-declaration -> type-specifier ID ; | type-specifier ID [NUM] ; */
+ /* 5. type-specifier -> int | void */
```

실제 문법 틀 만들기

```
  /* 3. declaration -> var-declaration | fun-declaration */
+ declaration :
+ var-declaration {
+
+ }
+ | fun-declaration {
+
+ };
+
  /* 4. var-declaration -> type-specifier ID ; | type-specifier ID [NUM] ; */
+ var-declaration :
+ type-specifier ID SEMI {
+
+ }
+ | type-specifier ID LBRACE NUM RBRACE SEMI {
+
+ };
+
  /* 5. type-specifier -> int | void */
```

globals 에 사용하는 Kind 추가

```
- typedef enum {IfK,RepeatK,AssignK,ReadK,WriteK} StmtKind;
- typedef enum {OpK,ConstK,IdK} ExpKind;
+ // STMT -> rectangle
+ // EXP  -> elipse
+ typedef enum {StmtK, ExpK, TokenTypeK} NodeKind;
+ typedef enum
+ {VarK, VarArrayK, NUMK, SingleParamK,
+ ArrayParamK, AssignK, IdK, OpK, ConstK, CallK} ExpKind;
+ typedef enum
+ {FunctionK, ParamsK, CompoundK, LocalDeclarationsK,
+ StatementLisstK, WhileK, IfK, ReturnK, ArgsK} StmtKind;
```

Array Type 정의

```
+ typedef struct ArrayType {
+   char * name;
+   int length;
+ } ArrayType;
+
  #define MAXCHILDREN 3

  typedef struct treeNode
  @@ -79,7 +89,9 @@ typedef struct treeNode
       union { StmtKind stmt; ExpKind exp;} kind;
       union { TokenType op;
               int val;
-              char * name; } attr;
+              char * name;
+              ArrayType arr;
+              } attr;
       ExpType type; /* for type checking of exps
```

appendix 실제 문법 구현

```
  var_declaration :
  type_specifier _id SEMI {
-
+   $$ = newExpNode(VarK);
+   $$->child[0] = $1;
+   $$->lineno = lineno;
+   $$->attr.name = savedName;
  }
  | type_specifier _id LBRACE _num RBRACE SEMI {
-
+   $$ = newExpNode(VarArrayK);
+   $$->child[0] = $1;
+   $$->lineno = lineno;
+   $$->attr.arr.name = savedName;
+   $$->attr.arr.length = savedNumber;
  };

  /* 5. type_specifier -> int | void */
```

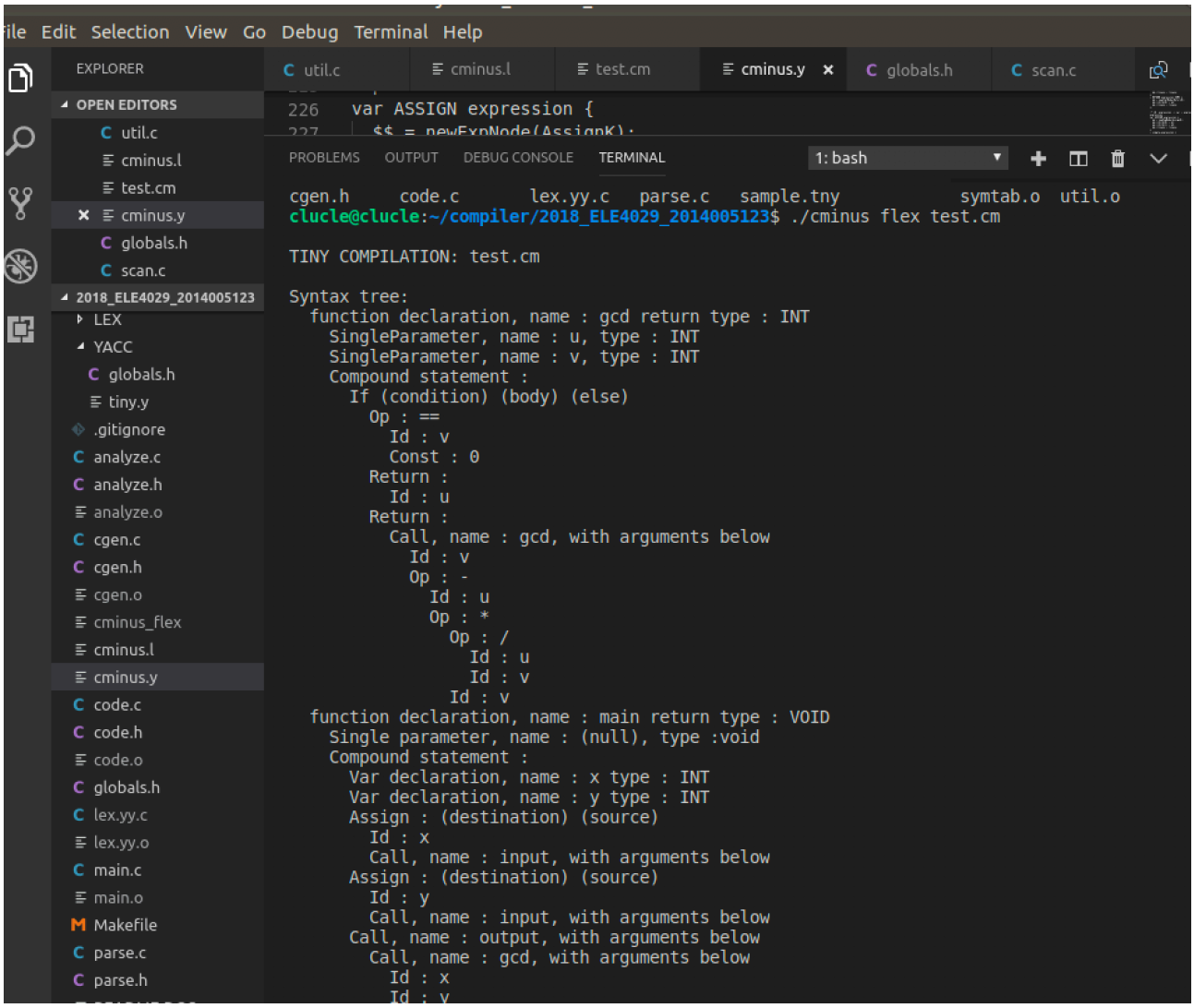
printTree구현

```
void printTree( TreeNode * tree )
{
  // TODO : c-minus syntax tree
@@ -167,19 +217,34 @@ void printTree( TreeNode * tree )
    { switch (tree->kind.stmt) {
        // 6
        case FunctionK:
-         fprintf(listing,"Function\n");
+         fprintf(listing,"function declaration, name : return type : "
+         printTreeToken(tree->child[0]);
+         printTree(tree->child[1]);
+         printTree(tree->child[2]);
          break;
        // 10
```

ID, NUM 값 뒤에 expression이나 다른것이 와서 saveId가 지워지는 에러를 수정

```
- | _id LBRACE expression RBRACE {
+ | _id {
    $$ = newExpNode(ArrayIdK);
    $$->attr.name = savedName;
-   $$->child[0] = $3;
    $$->lineno = lineno;
+ }
+ LBRACE expression RBRACE {
+   $$ = $2;
+   $$->child[0] = $4;
  };
```

result


File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
util.c
cminus.l
test.cm
cminus.y
globals.h
scan.c
2018_ELE4029_2014005123
LEX
YACC
globals.h
tiny.y
.gitignore
analyze.c
analyze.h
analyze.o
cgen.c
cgen.h
cgen.o
cminus_flex
cminus.l
cminus.y
code.c
code.h
code.o
globals.h
lex.yy.c
lex.yy.o
main.c
main.o
Makefile
parse.c
parse.h
226 var ASSIGN expression {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
cgen.h code.c lex.yy.c parse.c sample.tny symtab.o util.o
clucle@clucle:~/compiler/2018_ELE4029_2014005123$ ./cminus flex test.cm
TINY COMPILATION: test.cm
Syntax tree:
function declaration, name : gcd return type : INT
SingleParameter, name : u, type : INT
SingleParameter, name : v, type : INT
Compound statement :
If (condition) (body) (else)
Op : ==
Id : v
Const : 0
Return :
Id : u
Return :
Call, name : gcd, with arguments below
Id : v
Op : -
Id : u
Op : *
Op : /
Id : u
Id : v
Id : v
function declaration, name : main return type : VOID
Single parameter, name : (null), type :void
Compound statement :
Var declaration, name : x type : INT
Var declaration, name : y type : INT
Assign : (destination) (source)
Id : x
Call, name : input, with arguments below
Assign : (destination) (source)
Id : y
Call, name : input, with arguments below
Call, name : output, with arguments below
Call, name : gcd, with arguments below
Id : x