Syntax Analysis (Parser) Report

- 주하진, 2024062806

Compilation Environment and Method

주어진 `Makefile`을 이용해 C파일과 `cminus.l`, `cminus.y`를 변환함.
C파일은 `gcc`를 이용하고 `*.l`은 `flex`, `*.y`는 `yacc`을 이용함.

`Makefile`의 빌드 결과물 `cminus_parser`를 만들기 위해서 `main.c`, `util.c`, `lex.yy.o`, `y.tab.o`를 필요로 한다.

C-Minus Parser Implementation

C-Minus Parser 구현을 위해 다음 세 파일의 큰 수정이 필요했다.

- `globals.h`
- `util.c`, `util.h`
- `cminus.y` (Important)

`globals.h`

여러개의 Kind Enum을 추가하였다.

- NodeKind(큰 분류)
- StmtKind(Statement의 종류)
- ExpKind(Expression의 종류)
- DeclKind(Declaration의 종류)
- TypeKind(Declaration에서 Type을 구분하기 위해 사용, 실제로 파스트리에 들어가진 않음, var_declaration에서 참조하기 위한 목적.)

StmtKind

- IfK: if문
- IterK: while문
- ReturnK: return문
- CompK: 여러개 있는 중괄호(복합) 문

ExpKind

- AssignK: 할당문
- OpK: 연산자가 포함된 표현식
- ConstK: 상수
- IdK: 식별자
- ArrIdK: 배열 식별자
- CallK: 함수 호출

DeclKind

- FuncK: 함수 선언
- VarK: 변수 선언
- ArrVarK: 배열 변수 선언
- ArrParamK: 배열 매개변수
- NonArrParamK: 매개변수

TypeKind

- TypeNameK: 선언의 타입

`TreeNode`를 추가하였다.

```
typedef struct treeNode {
    struct treeNode *child[MAXCHILDREN];
    struct treeNode *sibling;
    int lineno;
    NodeKind nodekind;
    union {
        StmtKind stmt;
        ExpKind exp;
        DeclKind decl;
        TypeKind type;
    } kind;
    union {
        TokenType op;
        int val;
        char *name;
    } attr;
    ExpType type; /* for type checking of exps */
} TreeNode;
```

TreeNode 타입은 ParseTree의 노드를 나타내며, 자식 노드와 형제 노드를 가리키는 포인터 그리고 노드의 kind와 attr, type을 가진다.

`util.c`, `util.h`

`newStmtNode`, `newExpNode`, `newDeclNode`, `newTypeNode` 함수를 추가 및 수정했다. 각각 Statement, Expression, Declaration, Type 노드를 생성하는 함수이다.

Type을 출력하기 위해 `printType` 함수를 추가하였다.

printTree는 TreeNode를 출력하는 함수이다. nodeKind에 따라 구분하여 출력한다. 이때 type이 필요한 node이면 type도 같이 출력한다.

`cminus.y` (Important)

cminus.y에서 토큰의 선언은 다음과 같이 했다.

```
%token IF ELSE WHILE RETURN INT VOID
%token EQ NE LT LE GT GE LPAREN RPAREN LBRACE LCURLY RBRACE RCURLY SEMI
%token ID NUM


%left PLUS MINUS
%left TIMES OVER
%right ASSIGN

%nonassoc THEN
%nonassoc ELSE

%token ERROR
```

나머지 부분은 제공된 grammar와 tiny.y의 많은 부분을 참고하여 작성했다.

이때 중요한 부분은 dangling-else 부분이다.

```
selection_stmt : IF LPAREN expression RPAREN statement %prec THEN {
  ...
} | IF LPAREN expression RPAREN statement ELSE statement {
  ...
};
```

`single-if` 문의 우선순위를 `ELSE` 보다 낮은 `THEN` 으로 지정하여 Shift/Reduce Conflict를 해결했다.

Results

다음은 테스트 C-Minus 프로그램과 그에 대한 파스트리 출력 결과이다.

C-Minus Test Program	Parse Tree Output

```
void main(void)
{
        int i; int x[5];

        i = 0;
        while( i < 5 )
        {
                x[i] = input();

                i = i + 1;
        }

        i = 0;
        while( i <= 4 )
        {
                if( x[i] != 0 )
                {
                        output(x[i]);
                }
        }
}
```

```
C-MINUS COMPILATION: test.2.txt

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
```

```
int main(void){
    int a;
    int b;
    a = (b = 4) + 3;
    if(a==b+(c*b+d))
        while(1)
            if(1)
                a=2;
            else a=3;
}
```

```
C-MINUS COMPILATION: test.cm

Syntax tree:
  Function Declaration: name = main, return type = int
    Void Parameter
    Compound Statement:
      Variable Declaration: name = a, type = int
      Variable Declaration: name = b, type = int
      Assign:
        Variable: name = a
        Op: +
          Assign:
            Variable: name = b
            Const: 4
          Const: 3
      If Statement:
        Op: ==
          Variable: name = a
          Op: +
            Variable: name = b
            Op: +
              Op: *
                Variable: name = c
                Variable: name = b
              Variable: name = d
        While Statement:
          Const: 1
          If-Else Statement:
            Const: 1
            Assign:
              Variable: name = a
              Const: 2
            Assign:
              Variable: name = a
              Const: 3
```