2. Parser

2019027192 김현수

2021년 11월 28일

차 례

1 cminus.y

기존 프로젝트에 있던 tiny.y 파일을 기반으로, 과제 명세에 있는 BNF Grammar에 따라 코드를 작성했습니다. 과제 명세에 나온 트리 구조를 참고해서 대부분은 간단하게 구현이 가능했습니다. 보고서 분량 문제로 구현에 특이점이 있던 몇 가지만 아래에에 작성했습니다.

A) 재귀 문법

declaration_list, param_list와 같이 재귀적인 문법은 tiny.y 코드를 참고해 아래와 같이 처리했습니다.

```
param_list : param_list COMMA param
{
YYSTYPE t = $1;
if (t != NULL)
{
while (t->sibling != NULL)
{
t = t->sibling;
}
t->sibling = $3;
$$ = $1;
}
else
{
$$ = $3;
}
}
```

B) ID, NUM

tokenString와 lineno는 token을 읽어오며 계속 변합니다. 따라서 등장 시점과 트리에 들어가는 시점이 다른 ID와 NUM은 tokenString과 lineno를 따로 저장해 두어야 합니다.

```
saveName            : ID
{
savedName = copyString(tokenString);
savedLineNo = lineno;
}
;

saveNum             : NUM
{
savedNum = atoi(tokenString);
savedLineNo = lineno;
}
;
```

ID, NUM이 필요한 쪽에서는 대신 saveName과 saveNum을 사용하고, 추후에 트리 노드를 생성할 때 savedName과 savedNum을 이용해 값에 접근하면 됩니다.

C) Array Type

'int a[10];'나 'int func(int a[])'에서 a의 type은 int[]가 됩니다. 이렇게 Array Type을 나타내기 위해 TreeNode 구조체에 isarray 플래그를 추가했습니다.

isarray 플래그는

1. var_declaration : type_specifier saveName LBRACE saveNum RBRACE SEMI
2. param : type_specifier saveName LBRACE RBRACE

이렇게 두 패턴에서 참이 됩니다.

D) Dangling Else

BNF Grammar 대로 cminus.y를 구현하면, shift/reduce conflict가 발생해 warning이 발생합니다. ELSE를 IF_REDUCE보다 우선시하고, ELSE가 없는 IF 문의 우선순위를 IF_REDUCE와 같게 해 shift/reduce conflict를 해결했습니다.

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

위처럼 구현하면 selection_stmt를 유도할 때 항상 IF-ELSE (shift)를 우선시하므로, 과제명세대로 else가 가장 가까운 if와 연결됩니다.

2 util.c

과제 명세에 맞게 printTree를 수정했습니다. 그리고 getExpTypeString과 newDeclarationNode 함수를 추가했습니다.

A) getExpTypeString

node의 type에 해당하는 문자열을 반환하는 함수입니다. node-¿isarray가 true라면 int[]/void[], false라면 int/void를 반환하게 했습니다.

B) newDeclarationNode

선언과 정의를 구분하기 위해 DeclarationKind를 생성했으므로, DeclarationK 타입의 TreeNode를 생성하는 newDeclarationNode 함수를 구현했습니다. newStmtNode, newExpNode와 큰 차이점은 없습니다.

3 compilation environment

Ubuntu 20.04 LTS

gcc (Ubuntu 9.3.0-17ubuntu1 20.04) 9.3.0

flex 2.6.4

bison (GNU Bison) 3.5.1

4 test

2_Parser 폴더 내에서 make 명령어를 실행하면 cminus_parser 파일이 생성됩니다.

```
 dodo@dodo ~/github/2021_ele4029_2019027192/2_Parser master make
yacc -d -v cminus.y
gcc -W -Wall -c main.c
gcc -W -Wall -c util.c
flex cminus.l
gcc -W -Wall -c lex.yy.c
lex.yy.c:1274:17: warning: ‘yyunput’ defined but not used [-Wunused-function]
 1274 |     static void yyunput (int c, char * yy_bp )
      |                 ^~~~~~~
gcc -W -Wall -c y.tab.c
gcc -W -Wall main.o util.o lex.yy.o y.tab.o -o cminus_parser -lfl
```

제공된 테스트 파일들을 cminus_parser의 인자로 넘기고, 결과물로 출력된 파일을 제공된 result 파일과 비교했습니다.

```
 dodo@dodo ~/github/2021_ele4029_2019027192/2_Parser master ./cminus_parser ./test.1.txt > program.1.txt; d
iff result.1.txt program.1.txt
 dodo@dodo ~/github/2021_ele4029_2019027192/2_Parser master ./cminus_parser ./test.2.txt > program.2.txt; d
iff result.2.txt program.2.txt
```

비교 결과를 보면, parser의 출력과 제공된 result의 내용이 완전히 동일함을 알 수 있습니다.