컴파일러 설계

Project #2 : Parser

컴퓨터소프트웨어학부
2018008395
박정호

1. 실행 환경

   우분투에서 진행했고, 버전은 16.04.12이다. GCC 버전은 다음과 같다.

   컴파일은 명세에서 안내된 내용 그대로 Makefile을 사용했다.

```
pch@pch-VirtualBox:~$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

2. Implementation

   A. Syntax Tree를 위한 자료구조

   기본적으로 노드의 TYPE은 statement, expression, declaration, parameter의 네 가지로 두었다.

   또한 TreeNode 구조체의 형식도 약간 변경했다. 예제 출력에서 함수나 변수의 정보에 이름과 함께 그 형식도 출력되는 것을 보고 함수/변수 선언이나 매개 변수 선언에 사용할 DeclAttr이라는 구조체를 만들어서 그 함수/변수의 이름과 형식을 함께 저장하도록 했다.

   B. 이름 저장을 위한 처리

   함수 선언이나 배열 접근 등의 경우에서 결과적으로 identifier가 여러 번 쓰이는 경우가 존재할 수 있었다. 이럴 때는 각 token의 tokenString을 저장할 필요가 있었다. 그래서 identifier -> ID; 라는 새로운 grammar를 추가했다. 이 grammar는 단순히 tokenString을 전역 변수 savedName에 저장하는 역할을 한다. 또한 이후의 identifier 파싱으로 인해 savedName이 덮여쓰여지는 것을 막기 위해, identifier를 reduce한 직후, savedName에 저장된 tokenStirng을 노드에 저장하고, 다시 남은 것들을 모두 스택에 넣어서 파싱을 진행했다. 다음 페이지에 보여지는 이미지와 같은 형식이라고 보면 된다.

```
fun_declaration       : INT identifier
                        {
                          $$ = newDeclNode(FunK);
                          $$->lineno = savedLineNo;
                          $$->attr.decl.name = savedName;
                          $$->attr.decl.type = INT;
                        }
                      LPAREN params RPAREN compound_stmt
                        {
                          $$ = $3;
                          //save type, parameters, body
                          $$->child[0] = $1;
                          $$->child[1] = $5;
                          $$->child[2] = $7;
                        }
```

C. Dangling Else Problem의 해결

여타 grammar 들은 기존의 tiny.y의 내용물을 약간 고치거나 응용하는 정도로 해결이 가능했으나, 딱 하나의 문제가 있었다. 바로 Dangling Else Problem 이었는데, 이는 yacc에서 제공하는 명령어 중 하나인 %prec으로 해결했다. NO_ELSE라는 가상의 토큰을 두고, if (expression) stmt 의 형태를 가지는 token stream에 대해서는 NO_ELSE로 지정된 우선순위로 파싱하게 한 것이다. 따라서 nearest if에 else가 붙을 수 있게 되었다.

D. Empty Statement의 처리

문법의 처리에 있어 또 하나 문제가 되는 부분은 stmt : SEMI, 즉 빈 문장의 경우였다. 아무런 동작이 없는 문장도 AST에 추가가 되어야하는지에 대한 고려가 필요했기 때문이다. 결론적으로는 추가하지 않기로 했다. 실제로 컴파일러에서는 이러한 빈 문장은 결국 실행 파일에서 배제할 것이 분명하며, 이 문장이 없어도 기존 프로그램은 제대로 동작할 것이기 때문이다.

즉, 빈 문장의 경우 빈 노드 즉 NULL을 주는 것으로 처리했다. 이로 인해서 출력에도 영향이 있었는데, 이에 대한 것은 다음 단원에서 설명하겠다.

E. util.c, global.h의 수정

global.h의 수정은 앞의 "A. Syntax Tree를 위한 자료구조"에서 설명한 것이 전부이다. util.c에서는 TreeNode를 만드는 함수들의 추가와 printTree의 수정이 이루어졌다. TreeNode를 만드는 newDeclNode, newParamNode, newStmtNode, newExpNode들은 모두 기존함수와 거의 같다. printTree의 경우 과제 명세에서 보여준 예시와 동일한 출력이 이루어지도록 처리했는데, 출력 예시가 없는 문법의 경우 기존 형식과 비슷한 방식으로 어느 정도 통일감 있게 구성했다. 어떻게 출력되었는지에 대해서는 다음 단원에서 설명하겠다.

3. 실행 결과

실행에 사용한 input은 지난 과제에서 사용한 제공된 test1.txt, test2.txt, 그리고 여기서 다루어지지 않은 문법들을 사용한 몇가지 custom test case를 사용했음을 밝힌다. 각 테스트케이스의 내용물과 파싱 결과를 같이 설명하도록 하겠다.

A. test1.txt

기존에 주어진 테스트이다.

과제 명세에서 주어진 출력과 같은 것을 확인할 수 있다.

```
1  /* A program to perform Euclid's
2     Algorithm to computer gcd*/
3
4  int gcd (int u, int v)
5  {
6      if (v == 0) return u;
7      else return gcd(v, u-u/v*v);
8      /*u-u/v*v == u mod v*/
9  }
10
11 void main(void)
12 {
13     int x; int y;
14     x = input(); y = input();
15     output(gcd(x,y));
16 }
```

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/2_Parser$ ./cminus test/test1.cm

C-MINUS COMPILATION: test/test1.cm

Syntax tree:
  Function declaration, name : gcd, return type : int
    Single Parameter, name : u, type : int
    Single Parameter, name : v, type : int
    Compound statement :
      If (condition) (body) (else)
        Op : ==
          Id : v
          Const : 0
        Return :
          Id : u
        Return :
          Call, name : gcd, with argument 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 argument below
      Assign : (destination) (source)
        Id : y
        Call, name : input, with argument below
      Call, name : output, with argument below
        Call, name : gcd, with argument below
          Id : x
          Id : y
```

B. test2.txt

이전 과제에서 사용한 테스트 케이스 중 하나이다.

배열 선언 시 child로 사이즈가 함께 출력되고, 배열 접근 시 child로 인덱스에 해당하는 expression이 있는 것을 볼 수 있다. 또한 반복문도 잘 처리되는 것을 볼 수 있다.

```
1  void main(void)
2  {
3      int i; int x[5];
4
5      i = 0;
6
7      while( i < 5 )
8      {
9          x[i] = input();
10         i = i + 1;
11     }
12
13     i = 0;
14     while( i <= 4 )
15     {
16         if( x[i] != 0 )
17         {
18             output(x[i]);
19         }
20     }
21 }
```

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/2_Parser$ ./cminus test/test2.cm

C-MINUS COMPILATION: test/test2.cm

Syntax tree:
  Function declaration, name : main, return type : void
    Single Parameter, name : (null), type : void
    Compound statement :
      Var declaration, name : i, type : int
      Array declaration, name : x, type : int, with size below
        Const : 5
      Assign : (destination) (source)
        Id : i
        Const : 0
      While (condition) (body)
        Op : <
          Id : i
          Const : 5
        Compound statement :
          Assign : (destination) (source)
            ArrId : x, with index below
              Id : i
            Call, name : input, with argument below
          Assign : (destination) (source)
            Id : i
            Op : +
              Id : i
              Const : 1
      Assign : (destination) (source)
        Id : i
        Const : 0
      While (condition) (body)
        Op : <=
          Id : i
          Const : 4
        Compound statement :
          If (condition) (body)
            Op : !=
              ArrId : x, with index below
                Id : i
              Const : 0
            Compound statement :
              Call, name : output, with argument below
                ArrId : x, with index below
                  Id : i
```

C. test3.txt

과제 명세에서 보여준 예시이다.

분명 선언되지 않은 변수 c가 사용되었지만, syntax 자체의 에러는 없기 때문에 파싱이 제대로 되는 것을 볼 수 있다.

```
1 int main(void)
2 {
3       int a;
4       int b;
5       c = a + b;
6 }
```

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/2_Parser$ ./cminus test/test3.cm

C-MINUS COMPILATION: test/test3.cm

Syntax tree:
  Function declaration, name : main, return type : int
    Single Parameter, name : (null), type : void
    Compound statement :
      Var declaration, name : a, type : int
      Var declaration, name : b, type : int
      Assign : (destination) (source)
        Id : c
        Op : +
          Id : a
          Id : b
```

D. test4.txt

dangling else problem이 발생할 수 있는 소스코드이다.

명세에서 안내된 대로 nearest if 에 else가 속하는 것을 볼 수 있다.

(else 부분이 없는 if문은 출력 시에도 if (condition) (body) 까지만 출력된다.)

```
1 int main(void){
2     int a; int b; int c;
3     if ( a == b )
4         if ( b == c )
5             output(1);
6         else
7             output(y);
8 }
```

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/2_Parser$ ./cminus test/test4.cm

C-MINUS COMPILATION: test/test4.cm

Syntax tree:
  Function declaration, name : main, return type : int
    Single Parameter, name : (null), type : void
    Compound statement :
      Var declaration, name : a, type : int
      Var declaration, name : b, type : int
      Var declaration, name : c, type : int
      If (condition) (body)
        Op : ==
          Id : a
          Id : b
        If (condition) (body) (else)
          Op : ==
            Id : b
            Id : c
          Call, name : output, with argument below
            Const : 1
          Call, name : output, with argument below
            Id : y
```

E. test5.txt

배열 접근 시 조금 복잡한 expression이 인덱스로 사용이 가능한지 확인하고, 배열을 매개 변수로 사용하는 함수의 선언, 전역 변수의 선언이 제대로 파싱되는지 확인하는 테스트이다. 문제 없는 것을 확인했다.

```
1 void novalue;
2 void noarray[50];
3 int value;
4 int array[10];
5
6 int size(void arr[]){
7     return strlen(arr)/sizeof(arr[0]);
8 }
9
10
11 int main(void){
12     {
13         output(array[10+value/3]);
14     }
15 }
```

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/2_Parser$ ./cminus test/test5.cm

C-MINUS COMPILATION: test/test5.cm

Syntax tree:
  Var declaration, name : novalue, type : void
  Array declaration, name : noarray, type : void, with size below
    Const : 50
  Var declaration, name : value, type : int
  Array declaration, name : array, type : int, with size below
    Const : 10
  Function declaration, name : size, return type : int
    Array Parameter, name : arr, type : void
    Compound statement :
      Return :
        Op : /
          Call, name : strlen, with argument below
            Id : arr
          Call, name : sizeof, with argument below
            ArrId : arr, with index below
              Const : 0
  Function declaration, name : main, return type : int
    Single Parameter, name : (null), type : void
    Compound statement :
      Compound statement :
        Call, name : output, with argument below
          ArrId : array, with index below
            Op : +
              Const : 10
              Op : /
                Id : value
                Const : 3
```

F. test6.txt

empty statement의 처리를 중점적으로 본 테스트이다. 여기서 분명 if else 구문인데도 else가 처리되지 않은 것을 볼 수 있는데, 필자는 단일 if와 if else를 2번 인덱스의 자식이 NULL이냐 아니냐에 따라서 판단했으므로, 빈 문장은 결국 NULL이기에 단일 if문이 쓰였다고 판단한 것이다. 또한 빈문장은 아예 printTree에서 출력되지 않는 것을 확인할 수 있다.

```
void main(void)
{
   if (a)
      while (a)
      if (a)
        ;
   else
      ;
}
```

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/2_Parser$ ./cminus test/test6.cm

C-MINUS COMPILATION: test/test6.cm

Syntax tree:
  Function declaration, name : main, return type : void
    Single Parameter, name : (null), type : void
    Compound statement :
      If (condition) (body)
        Id : a
        While (condition) (body)
          Id : a
          If (condition) (body)
            Id : a
```