Compiler Term Project

3. Semantic Report

2014004211

컴퓨터소프트웨어학부

표영권

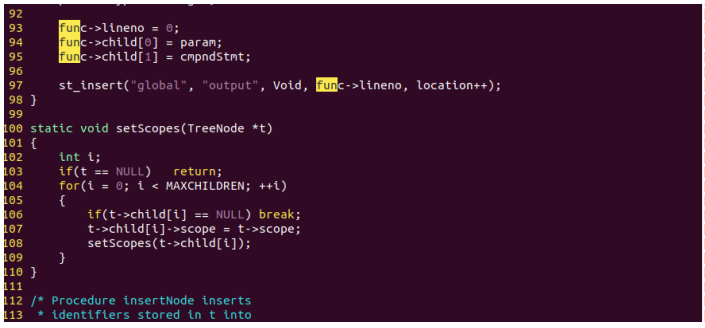
과제 요구사항

1. C-Minus 언어의 Symbol Table을 작성
2. input(), output()을 global area에 먼저 정의
3. C- Minus 언어의 Semantic analysis를 진행
    a. type check
    b. duplicate declaration check
    c. undefined check

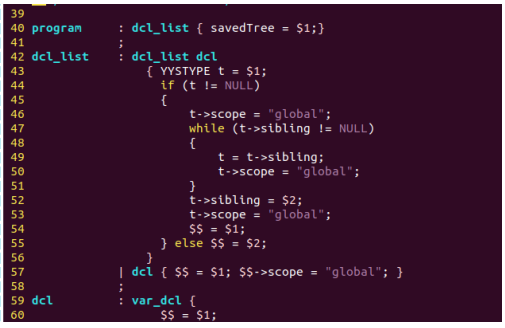
실행 방법

- Build : $make
- Run : $ ./cminus [테스트할 파일명]
- Remove output files : $make clean

구현 과정



```
92
93      func->lineno = 0;
94      func->child[0] = param;
95      func->child[1] = cmpndStmt;
96
97      st_insert("global", "output", Void, func->lineno, location++);
98  }
99
100 static void setScopes(TreeNode *t)
101 {
102     int i;
103     if(t == NULL)   return;
104     for(i = 0; i < MAXCHILDREN; ++i)
105     {
106         if(t->child[i] == NULL) break;
107         t->child[i]->scope = t->scope;
108         setScopes(t->child[i]);
109     }
110 }
111
112 /* Procedure insertNode inserts
113  * identifiers stored in t into
```



```
39
40 program     : dcl_list { savedTree = $1;}
41              ;
42 dcl_list    : dcl_list dcl
43                 { YYSTYPE t = $1;
44                   if (t != NULL)
45                   {
46                       t->scope = "global";
47                       while (t->sibling != NULL)
48                       {
49                           t = t->sibling;
50                           t->scope = "global";
51                       }
52                       t->sibling = $2;
53                       t->scope = "global";
54                       $$ = $1;
55                   } else $$ = $2;
56                 }
57              | dcl { $$ = $1; $$->scope = "global"; }
58              ;
59 dcl         : var_dcl {
60                   $$ = $1;
```

analye.c 파일에서 setScope와 traversal을 이용하여 AST의 scope를 forward propagation의 원리로 지정하도록 하였다. 그리고 함수의 경우에는 global area에서만 선언이 가능하므로, cminus.y 파일의 yacc 코드에서 이를 구현하였다.

```
51 static void insertIOfunc()
52 {
53     TreeNode *prime;
54     TreeNode *func;
55     TreeNode *param;
56     TreeNode *cmpndStmt;
57 
58     prime = newPrimeNode();
59     prime->attr.name = "input";
60     prime->type = Integer;
61 
62     cmpndStmt = newStmtNode(CmpndK);
63     cmpndStmt->child[0] = NULL;       // no local var
64     cmpndStmt->child[1] = NULL;       // no stmt
65 
66     func = prime;
67     func->nodekind = DclK;
68     func->kind.dcl = FdclK;
69 
70     func->lineno = 0;
71     func->child[0] = param;
72     func->child[1] = cmpndStmt;
73     st_insert("global", "input", Integer, func->lineno, location++);
74 
75 
76     prime = newPrimeNode();
77     prime->attr.name = "output";
78     prime->type = Void;
79 
80     cmpndStmt = newStmtNode(CmpndK);
81     cmpndStmt->child[0] = NULL;       // no local var
82     cmpndStmt->child[1] = NULL;       // no stmt
83 
84     func = prime;
85     func->nodekind = DclK;
86     func->kind.dcl = FdclK;
87 
88     param = newPrimeNode();
89     param->attr.name = "arg";
90     param->is_param = TRUE;
91     param->type = Integer;
92 
93     func->lineno = 0;
94     func->child[0] = param;
95     func->child[1] = cmpndStmt;
96 
97     st_insert("global", "output", Void, func->lineno, location++);
98 }
99 
```

또한 과제 요구 사항 중 하나인 input() 함수와 output() 함수를 symbol table에 넣는 과정이다. 고정적인 global area이므로 st_insert를 통해 symbal table을 추가할 때 "global" const string 형태로 저장해주었다.

```
47 void st_insert( char * scope, char * name, ExpType type, int lineno, int loc )
48 { int h1 = hash(scope), h2 = hash(name);
49   ScopeList sl = hashTable[h1];
50
51   while ((sl != NULL) && (strcmp(scope,sl->name) != 0))
52     sl = sl->parent;
53   if (sl == NULL)
54   {
55       sl = (ScopeList)malloc(sizeof(struct ScopeListRec));
56       sl->name = scope;
57       sl->parent = hashTable[h1];
58       hashTable[h1] = sl;
59   }
60
61   /* found scope or inserted new scope */
62   BucketList bl = sl->bucket[h2];
63   while((bl != NULL) && (strcmp(name, bl->name) != 0))
64       bl = bl->next;
65   if (bl == NULL) /* variable not yet in table */
66   { bl = (BucketList) malloc(sizeof(struct BucketListRec));
67     bl->name = name;
68     bl->type = type;
69     bl->lines = (LineList) malloc(sizeof(struct LineListRec));
70     bl->lines->lineno = lineno;
71     bl->memloc = loc;
72     bl->lines->next = NULL;
73     bl->next = sl->bucket[h2];
74     sl->bucket[h2] = bl;
75   }
76   else
77   {
78      st_add_lineno(&bl, lineno);
79   }
80 } /* st_insert */
81
82 void st_add_lineno(BucketList *blp, int lineno)
83 {
84     //printf("add lineno\n");
85     BucketList bl = *blp;
86     LineList t = bl->lines;
87     while(t->next != NULL) t = t->next;
88     t->next = (LineList) malloc(sizeof(struct LineListRec));
89     t->next->lineno = lineno;
90     t->next->next = NULL;
91 }
```

symbal table을 삽입하는 함수의 구현부분이다. 하단의 st_add_lineno()의 경우, 기존에 선언되어 있는 상태에서 새로 use를 할 때, 새롭게 use된 곳의 line number를 저장하기 위해 기록하는 함수이다.

실행 화면

현재 구현이 모두 되어 있지 않고, 아직 symbol table 구현에서 이슈가 있어 지금까지 성공한 케이스 일부를 올려 두겠다.

```
young@young-VirtualBox:~/바탕화면/workspace/Compiler/2020_ELE4029_2014004211/3_Semantic/src$ ./cminus t1.tm
C-MINUS COMPILATION: t1.tm

Building Symbol Table...

Symbol table:


Name           type      Scope Name  Location   Line Numbers
-------------  -------   ----------  --------   ------------
input          int       global      0          0
a              int       global      2          1
output         void      global      1          0

Checking Types...

Type Checking Finished
```

intput(), output() 그리고 임의로 정의한 global variable a도 정상적으로 symbol table에 삽입되어 있음을 알 수 있다.

이슈와 진행 상황

```
    case StmtK:
     switch (t->kind.stmt)
     {
         case CmpndK:
             t->child[0]->scope = t->child[1]->scope = copyString(t->scope);
             break;
         case ExpSK:
             switch(t->kind.exp)
             {
                 case IdK:
                 case AssignK:
                 case CallK:
                   bl = st_lookup(t->scope, t->attr.name);
                   if(bl == NULL)
                   {
                       st_insert(t->scope, t->attr.name, t->type, t->lineno, location++);
                       //semanticError(t, "undeclared symbol.");
                   }
                   else
                   {
```

현재 과제 진행 상황은 symbol table의 작성 중에 있다. 가장 성공한 케이스는 함수의 선언과 local variable까지 symbol table 작성이 성공한 것을 보였으나, 여러 가지 코드를 수정하던 중 어디선가 segmentation fault error가 발생하고 있다. scope를 propagation하는 과정에서 발생하는 것으로 추측되며, 이를 고치고 정상적으로 table을 작성한 후 type checking, undeclared error, duplicate declaration error 등을 analysis할 예정이다.