Project3 Semantics

2019094511 김준표

1. Compilation method and environment

- Environment: Windows 10 WSL2 – Ubuntu 18.04

- Run:

make → 컴파일 실시, cminus_semantic 생성

execute → ./cminus_semantic test.1.txt

2. Explanation about how to implement and how to operate + Code

- symtab.c

sc_insert() 함수를 통해 symbol을 table에 넣어주고 st_lookup()을 통해 symbol entry의 위치를 출력하는 부분을 구현하는 것이 핵심이다.

```
/* Procedure st_insert inserts line numbers and
 * memory locations into the symbol table
 * loc = memory location is inserted only the
 * first time, otherwise ignored
 */
void st_insert( char * name, int lineno, int loc, TreeNode * node )
{
  int h = hash(name);
  ScopeList curScope = scTop();
  BucketList l = curScope->bucket[h];
  while ((l != NULL) && (strcmp(name,l->name) != 0))
    l = l->next;
  if (l == NULL) /* variable not yet in table */
  { l = (BucketList) malloc(sizeof(struct BucketListRec));
    l->name = name;
    l->treeNode = node;
    l->lines = (LineList) malloc(sizeof(struct LineListRec));
    l->lines->lineno = lineno;
    l->memloc = loc;
    l->lines->next = NULL;
    l->next = curScope->bucket[h];
    curScope->bucket[h] = l; }
  else /* found in table, so just add line number */
  { //LineList t = l->lines;
    //while (t->next != NULL) t = t->next;
    //t->next = (LineList) malloc(sizeof(struct LineListRec));
    //t->next->lineno = lineno;
    //t->next->next = NULL;
  }
} /* st_insert */

/* Function st_lookup returns the memory
 * location of a variable or NULL if not found
 */
```

위의 코드를 통해 table에 symbol을 넣어주고 이를 바탕으로 lookup을 실시한다.

또한 해당 부분에서는 각각의 Symbol table을 출력하는 함수를 구현한다.

```
void printSymTab(FILE * listing)
{
  printSymbolTab(listing);
  fprintf(listing,"\n");
  printFunTab(listing);
  fprintf(listing,"\n");
  printFunGlob(listing);
  fprintf(listing,"\n");
  printFunParam(listing);
}

void printSymbolTab(FILE * listing)
{
  int sc, bk;
  fprintf(listing,"< Symbol Table >\n");
  fprintf(listing,"Variable Name  Variable Type  Scope Name  Location   Line Numbers\n");
  fprintf(listing,"-------------  -------------  ----------  --------   ------------\n");
  for (sc = 0; sc < scopeIdx; sc++)
  { ScopeList curScope = scopes[sc];
    BucketList * hashTable = curScope->bucket;
    for (bk = 0; bk < SIZE; bk++){
      if (hashTable[bk] != NULL)
      { BucketList l = hashTable[bk];
        TreeNode * node = l->treeNode;
        while (l != NULL)
        { LineList t = l->lines;
          fprintf(listing,"%-14s ",l->name);
          switch (node->nodekind)
          {
            case DeclK:
              switch (node->kind.decl) {
                case VarK:
                  switch (node->type)
                  {
                    case Void:
                      fprintf(listing, "%-15s", "Void");
                      break;
                    case Integer:
                      fprintf(listing, "%-15s", "Integer");
                      break;
                    default:
                      break;
```

이와 같이 hashtable 마다 순회하며 노드 별로 출력을 확인해준다.

- analyze.c

해당 부분에서는 buildSymtab()을 통해 Symtab을 생성하고 typechecking을 수행한다. buildSymtab()에서는 built-in function인 input(), output()을 추가한다. 또한 semantic errors에 대한 에러 메시지를 출력한다.

```
/* Procedure insertNode inserts
 * identifiers stored in t into
 * the symbol table
 */
static void insertNode( TreeNode * t)
{ switch (t->nodekind)
  { case DeclK:
      switch (t->kind.decl)
      {
        case FunK:
          funcName = t->attr.name;
          if (st_lookup(t->attr.name) >= 0) {
            redefinedError(t);
            break;
          }
          if (scTop() != globalScope) {
            funcDeclNotGlobal(t);
            break;
          }
          st_insert(funcName, t->lineno, addLocation(), t);
          scPush(createScope(funcName));
          inScope = TRUE;
        case VarK:
        case ArrVarK:
          name = t->attr.name;
          t->type = Integer;
          if (st_lookup(name) < 0) st_insert(name, t->lineno, addLocation(), t);
          else redefinedError(t);
          break;
        case ParamK:
          if (t->child[0]->attr.type == VOID) break;
          if (st_lookup(t->attr.name) == -1) {
            st_insert(t->attr.name, t->lineno, addLocation(), t);
            t->type = Integer;
          }
          break;
        default:
          break;
      }
      break;
```

```
    case StmtK:
      switch (t->kind.stmt)
      { case CompK:
          if (inScope) inScope = FALSE;
          else {
            ScopeList scope = createScope(funcName);
            scPush(scope);
            location++;
          }
          t->attr.scope = scTop();
          break;
        default:
          break;
      }
    case ExpK:
      switch (t->kind.exp)
      { case IdK:
        case IdxK:
        case CallK:
          if (st_lookup(t->attr.name) == -1)
          /* not yet in table, so treat as new definition */
            undeclaredError(t);
          else
          /* already in table, so ignore location,
             add line number of use only */
            st_line_add(t->attr.name, t->lineno);
          break;
        default:
          break;
      }
      break;
    default:
      break;
  }
}
```

<insert function>

```c
static void insertBuiltIn(void)
{
  TreeNode * func;
  TreeNode * param;
  TreeNode * typeSpec;
  TreeNode * compStmt;

  func = newDeclNode(FunK);
  typeSpec = newDeclNode(TypeK);
  typeSpec->attr.type = INT;
  func->type = Integer;

  compStmt = newStmtNode(CompK);
  compStmt->child[0] = NULL;
  compStmt->child[1] = NULL;

  func->lineno = 0;
  func->attr.name = "input";
  func->child[0] = typeSpec;
  func->child[1] = NULL;
  func->child[2] = compStmt;

  st_insert("input", 0, addLocation(), func);
```

<Built-in function>

input과 output에 대해 동일하게 실시해준다.

3. Screenshots

test 1, test 2에 대해서는 에러가 symbol table 생성에서 발생했으며 test 3의 경우 정상적으로 출력되었다. 에러의 원인을 찾는 중에 있다.

Test.1.txt

```
/mnt/c/Users/pyo99/CS/Compiler/2021_ele4029_2019094511/3_Semantic | master ?44  ./cminus_semantic test.1.txt

TINY COMPILATION: test.1.txt

Building Symbol Table...
[1]    21431 segmentation fault  ./cminus_semantic test.1.txt
```

Test.2.txt

```
/mnt/c/Users/pyo99/CS/Compiler/2021_ele4029_2019094511/3_Semantic | master ?44  ./cminus_semantic test.2.txt

TINY COMPILATION: test.2.txt

Building Symbol Table...
[1]    21460 segmentation fault  ./cminus_semantic test.2.txt
```

Test.3.txt

```
/mnt/c/Users/pyo99/CS/Compiler/2021_ele4029_2019094511/3_Semantic | master ?44  ./cminus_semantic test.3.txt

TINY COMPILATION: test.3.txt

Building Symbol Table...
Error: Undeclared Variable "d" at line 10
< Symbol Table >
Variable Name  Variable Type  Scope Name  Location   Line Numbers
-------------  -------------  ----------  --------   ------------
main           Function       global      2          6
input          Function       global      0          0
output         Function       global      1          0
a              IntegerArray   main        0          6
b              Integer        main        1          8 10
c              Integer        main        2          9 10

< Function Table >
Function Name  Scope Name  Return Type  Parameter Name   Parameter Type
-------------  ----------  -----------  --------------   --------------
main           global      Void                          Void
input          global      Integer                       Void
output         global      Void
                                                         Integer
               Void
               Void
               Void

< Function and Global Variables >
   ID Name      ID Type    Data Type
-------------  ---------  -----------
main           Function    Void
input          Function    Integer
output         Function    Void

< Function Parameters and Local Variables >
  Scope Name    Nested Level     ID Name      Data Type
--------------  ------------  -------------  -----------
global          0
global          0
global          0

Checking Types...

Type Checking Finished
```