C-Minus Compiler Implementation - 3_Semantic

2018008531 송연주

Compilation method and environment

실행환경 : VMWare Workstation 16 Player, Ubuntu 18.04.5 LTS

컴파일 및 실행 방법

```
$ make
$ ./cminus ./test/simple/[파일이름]
```

Semantic analysis implementation process and source code description of principle parts

1. `symtab.c`

   스택의 개념을 사용해 scope를 구현하였으며 필요한 함수들을 추가적으로 구현하였다.

   - `st_insert()`에서 line number만 add하는 부분을 분리하였다. 또한 기존의 `hashTable`을 삭제하고 `currScope->bucket[h]`를 hash table로 사용하였다.

```
void st_insert( char * name, TreeNode * node, int lineno, int loc ){
  int h = hash(name);
  ScopeList currScope = curr_scope();
  BucketList l =  currScope->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->lines = (LineList) malloc(sizeof(struct LineListRec));
    l->lines->lineno = lineno;
    l->node = node;
    l->memloc = loc;
    l->lines->next = NULL;
    l->next = currScope->bucket[h];
    currScope->bucket[h] = l;
  }
}
```

- `bucket_lookup()`과 `lookup()`은 조건에 맞는 BucketList를 찾아주는 것으로 `bucket_lookup()`은 scope의 parent를 iteration하면서 BucketList를 찾고 `lookup()`은 현재 scope에서 BucketList를 찾는다. 이 두 함수를 통해 다른 함수들을 구현하였다.

```
BucketList bucket_lookup (char * name)
{
  int h = hash(name);
  ScopeList currScope = curr_scope();

  while(currScope != NULL){
    BucketList l = currScope->bucket[h];

    while((l != NULL) && (strcmp(name,l->name) != 0))
      l = l->next;

    if(l != NULL) return l;

    currScope = currScope->parent;
  }
  return NULL;
}
```

```
BucketList lookup ( char * name, ScopeList scope )
{
  int h = hash(name);
  ScopeList currScope = scope;
  BucketList l = currScope->bucket[h];
  while((l != NULL) && (strcmp(name,l->name) != 0))
    l = l->next;
  if(l != NULL) return l;
  return NULL;
}
```

- scope를 추가하고 제거하는 것은 stack을 기반으로 구현하였다.

```
void push_scope(ScopeList scope)
{
  stack[nStack] = scope;
  location[nStack++] = 0;
}

void pop_scope()
{
  if(nStack>0)
  {
    nStack--;
  }
}
```

- symbol table을 출력하는 함수들을 구현하였다. `printSymTab()`이외에도 `printFuncTable()`, `printFuncGlobalVar()` 등을 추가적으로 구현하였다.

```c
void printSymTab(FILE * listing)
{
  int sc, bk;

  fprintf(listing,"< Symbol Table >\n");
  fprintf(listing,"Variable Name  Variable Type    Location      Scope Name    Line Numbers  \n");
  fprintf(listing,"-------------  -------------  -------------  -------------  ------------- \n");

  for(sc = 0; sc < nScope; sc++)
  {
    ScopeList currScope = scopes[sc];
    BucketList * l =  currScope->bucket;

    for(bk =0; bk < SIZE; bk++)
    {
      if (l[bk] != NULL){

        BucketList currBK = l[bk];

        while (currBK != NULL){
          fprintf(listing,"%-15s",currBK->name);

          TreeNode * node = currBK->node;
          switch (node->type)
          {
            case Void:
              fprintf(listing,"%-15s","Void");
              break;
            case Integer:
              fprintf(listing,"%-15s","Integer");
              break;
            default:
              break;
          }

          fprintf(listing,"%-15d",currBK->memloc);
          fprintf(listing,"%-15s",currScope->name);
          LineList linelist = currBK->lines;
          while(linelist != NULL){
            fprintf(listing,"%4d",linelist->lineno);
            linelist = linelist->next;
          }
```

2. `analyze.c`
   - `insertIOFunction()`을 통해 built in function인 `input()`과 `output()`을 추가하였다.

```
static void insertIOFunction()
{
  TreeNode * function;
  TreeNode * type;
  TreeNode * parameter;
  TreeNode * parameter_child;
  TreeNode * comp;

  function = newDeclNode(FunK);
  function->type = Integer;
  function->lineno = 0;
  function->attr.name = "input";
  function->child[0] = type;
  function->child[1] = NULL;
  function->child[2] = comp;

  type = newDeclNode(TypeK);
  type->attr.type = INT;

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

  st_insert("input", function, 0, add_location());

  function = newDeclNode(FunK);
  function->type = Void;
  function->lineno = 0;
  function->attr.name = "output";
  function->child[0] = type;
  function->child[1] = parameter;
  function->child[2] = comp;

  type = newDeclNode(TypeK);
  type->attr.type = VOID;

  parameter = newParamNode(SingleParamK);
```

- 각 error에 대해 출력할 수 있는 함수들을 정의하였다.

```c
static void typeError(TreeNode* t, char* message)
{
  fprintf(listing,"Error: Type error at line %d: %s\n",t->lineno,message);
  Error = TRUE;
}

static void invalidError(TreeNode* t, char* message)
{
  fprintf(listing,"Error: Invalid array Indexing at line %d: %s\n",t->lineno,message);
  Error = TRUE;
}

static void symbolError(TreeNode* t, char* message)
{
  fprintf(listing,"Error: Symbol error at line %d: %s\n",t->lineno,message);
  Error = TRUE;
}

static void undeclaredError(TreeNode* t)
{
  if (t->kind.exp == CallK)
    fprintf(listing,"Error: Undeclared Function \"%s\" at line %d\n",t->attr.name,t->lineno);
  else if (t->kind.exp == IdK || t->kind.exp == ArrIdK)
    fprintf(listing,"Error: Undeclared Variable \"%s\" at line %d\n",t->attr.name,t->lineno);
  Error = TRUE;
}

static void redefinedError(TreeNode* t)
{
  if (t->kind.decl == VarK)
    fprintf(listing,"Error: Redefined Variable \"%s\" at line %d\n",t->attr.name,t->lineno);
  else if (t->kind.decl == ArrVarK)
    fprintf(listing,"Error: Redefined Variable \"%s\" at line %d\n",t->attr.arr.name ,t->lineno);
  else if (t->kind.decl == FunK)
    fprintf(listing,"Error: Redefined Function \"%s\" at line %d\n",t->attr.name,t->lineno);
  Error = TRUE;
}
```

- `checkNode()`에서 error를 detect하고 error의 종류에 따라 알맞은 내용을 출력하는 부분을 구현하였다.

```c
static void checkNode(TreeNode * t)
{ switch (t->nodekind)
  { case StmtK:
      switch (t->kind.stmt)
      {
        case IfK:
        case IfEK:
          if (t->child[0] == NULL)
            typeError(t,"expected expression");
          else if (t->child[0]->type == Void)
            typeError(t->child[0],"invalid if condition type");
          break;
        case WhileK:
          if (t->child[0] == NULL)
            typeError(t,"expected expression");
          else if (t->child[0]->type == Void)
            typeError(t->child[0],"invalid loop condition type");
          break;
        case ReturnK:
        {
```

3. `main.c`
   - symbol table을 출력하기 위해 관련된 부분의 코드를 수정하였다. 또한 과제 명세에 따라 flag를 조정하였다.

```
#if !NO_ANALYZE
  if (! Error)
  {
    if(TraceAnalyze) fprintf(listing,"\nBuilding Symbol Table...\n");
    buildSymtab(syntaxTree);
    if(TraceAnalyze) fprintf(listing,"\nChecking Types...\n");
    typeCheck(syntaxTree);
    if(TraceAnalyze) fprintf(listing,"\nType Checking Finished\n");

    if (TraceAnalyze && !Error){
      printSymTab(listing);
      fprintf(listing,"\n\n");
      printFuncTable(listing);
      fprintf(listing,"\n\n");
      printFuncGlobalVar(listing);
      fprintf(listing,"\n\n");
      printFuncParamLocalVar(listing);
    }
  }
```

Result Screenshot

```
coral@ubuntu:~/src$ ./cminus ./test/simple/type_error.cm

C-MINUS COMPILATION: ./test/simple/type_error.cm
coral@ubuntu:~/src$ ./cminus ./test/simple/void_var.cm

C-MINUS COMPILATION: ./test/simple/void_var.cm
Error: Variable Type cannot be Void at line 3 (name : x)
coral@ubuntu:~/src$ ./cminus ./test/simple/func.cm

C-MINUS COMPILATION: ./test/simple/func.cm
Error: Type error at line 12: invalid function call
coral@ubuntu:~/src$ ./cminus ./test/simple/undeclare.cm

C-MINUS COMPILATION: ./test/simple/undeclare.cm
Error: Undeclared Variable "x" at line 3
Error: Type error at line 3: invalid return type
coral@ubuntu:~/src$ ./cminus ./test/simple/indexing.cm

C-MINUS COMPILATION: ./test/simple/indexing.cm
Error: Type error at line 3: invalid function call
coral@ubuntu:~/src$ ./cminus ./test/simple/condition.cm

C-MINUS COMPILATION: ./test/simple/condition.cm
Error: Type error at line 2: invalid function call
Error: Type error at line 2: invalid if condition type
```