Project 3: Semantic Analysis

2016024875 손정우

1. Compilation methods and environment

To compile the project, I used Makefile example given by the TA. The environment that I build this project was as below:

- Ubuntu 16.04.7 (64-bits, on VMware Workstation)
- gcc 5.4.0 20160609
- flex 2.6.0
- GNU Bison 3.0.4

2. Implementation of C-minus Semantic Analyzer

1) Makefile

I used the Makefile given as an example in the project 2 specification. However, source files not used in this project were excluded from the creation of executable file *cminus*.

```
OBJS = main.o util.o lex.yy.o y.tab.o symtab.o analyze.o #code.o cgen.o
```

2) symtab.c & symtab.h

In TINY language, there are only one scope exists whereas C-minus can have multiple scopes. Thus, I had to create a new structure that wraps the *BucketListRec* structure. The following is the declaration of *ScopeListRec*. To manage this structure, I used the concept of stack.

```
/* symtab.h
 * The record for each scope (including name, its bucket, and parent scope)
 */
typedef struct ScopeListRec
  { char * scopeName;
    int nestedLevel;
    BucketList hashTable[SIZE];
    struct ScopeListRec * parent;
    struct ScopeListRec * next;
  } * ScopeList;

/* Miscellaneous functions for handling ScopeListRec
 * Same features in stack implementation (scTop for getting top elements,
 * scPush for pushing elements in the stack, scPop for popping up the element, and
 * scCreate for making new stack)
 */
ScopeList scCreate( char * scopeName );
ScopeList scTop( void );
```

```
void scPush( ScopeList scope );
void scPop( void );
```

Also, as mentioned earlier, it was necessary to modify the existing *st_insert()* and *st_lookup()* functions to handle multiple scopes. The following are the implementation of these functions.

```
void st_insert( char * name, int lineno, int loc, TreeNode * treeNode )
{ int h = hash(name);
  ScopeList top = scTop();
  BucketList l = top->hashTable[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 = treeNode;
    l->lines = (LineList) malloc(sizeof(struct LineListRec));
    l->lines->lineno = lineno;
    l->memloc = loc;
    l->lines->next = NULL;
    l->next = top->hashTable[h];
    top->hashTable[h] = l; }
  else /* found in table, so just add line number */
  { // ERROR!
    printf("Error occured in st_insert()\n");
  }
}

BucketList st_lookup_bucket( char * name )
{ int h = hash(name);
  ScopeList sc = scTop();
  while(sc)
  { BucketList l = sc->hashTable[h];
    while ((l != NULL) && (strcmp(name,l->name) != 0))
      l = l->next;
    if (l != NULL) return l;
    sc = sc->parent;
  }
  return NULL;
}

int st_lookup( char * name )
{ BucketList l = st_lookup_bucket(name);
  if (l != NULL)
    return l->memloc;
  return -1;
}

int st_lookup_sc( char * name )
{ int h = hash(name);
  ScopeList sc = scTop();
  if (sc)
  { BucketList l = sc->hashTable[h];
```

```
    while ((l != NULL) && (strcmp(name,l->name) != 0))
      l = l->next;
    if (l != NULL)
      return TRUE;
  }
  return FALSE;
}
```

3) Analyze.c & Analyze.h

The part that actually builds the symbol table and performs type checking through *ScopeListRec* implemented in *symtab.c* and *symtab.h* is *analyze.c*. Implementing the stack structure was as follows: whenever a new compound statement was encountered, a new scope was created and pushed onto the stack, and each time the compound statement was exited, it popped off the stack. (Note that the scope of the declaration and arguments must be the same.)

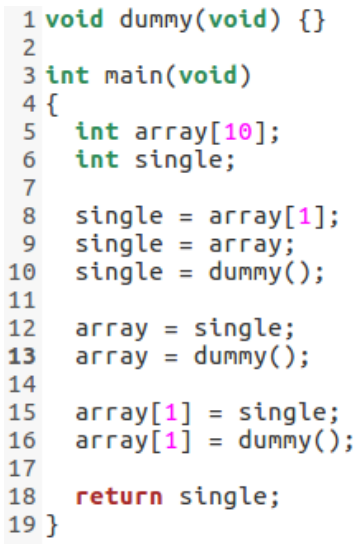
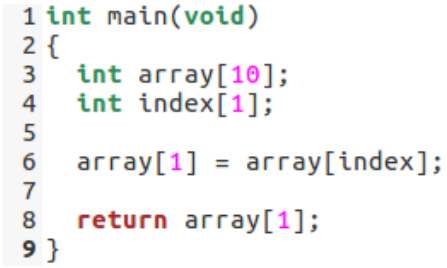
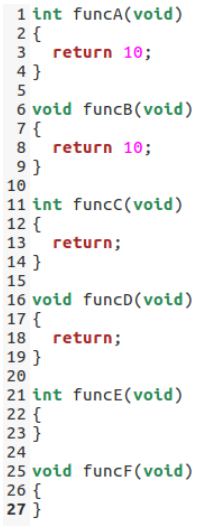
The most important functions in *analyze.c* are *insertNode()* and *checkNode()*. *insertNode()* plays the role of inserting data into the symbol table when building the symbol table. At this time, it may catch semantic errors such as redundant declaration of a variable or function or the use of an undeclared identifier. *checkNode()* plays the role of checking the codes line by line based on the symbol table created by the *buildSymtab()*.

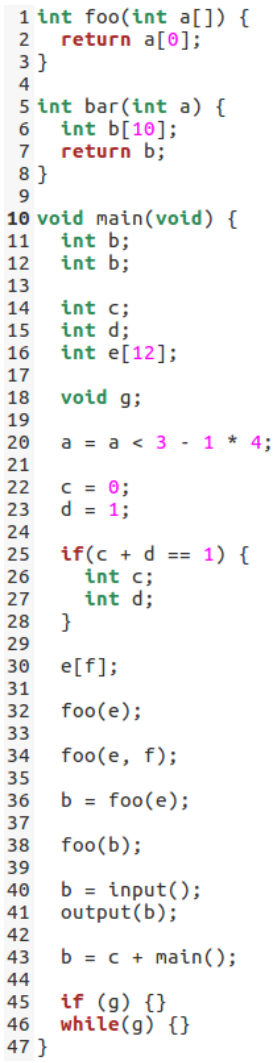
I also implemented the *insertBuiltinFunc()* function that inserts builtin functions, *input()* and *output()*, into the symbol table, as given in the specification. This function is executed before *buildSymtab()* calls traverse().

3. Example Results

I wrote a test case containing various semantic errors and tested the code I implemented. The following is a part of the result, and through the test cases, I was able to confirm that the code works without any issues.

Description	Source Code	Semantic Analysis Result
Invalid if/while condition	1 void dummy(void) {} 2 int array[10]; 3 int x; 4 5 int main(void) 6 { 7 if (dummy()) 8 return 1; 9 10 if (array) 11 return 2; 12 13 while (dummy()) 14 x = x + 1; 15 16 while (array) 17 x = x + 1; 18 }	C-MINUS COMPILATION: ./case_01.cm Type error at line 7: invalid expression (if-condition must be Integer type) Type error at line 10: invalid expression (if-condition must be Integer type) Type error at line 13: invalid expression (while-condition must be Integer type) Type error at line 16: invalid expression (while-condition must be Integer type)

Invalid assign operation		C-MINUS COMPILATION: ./case_02.cm Type error at line 9: type conflict in assignment Type error at line 10: type conflict in assignment Type error at line 12: type conflict in assignment Type error at line 13: type conflict in assignment Type error at line 16: type conflict in assignment
Invalid array index		C-MINUS COMPILATION: ./case_03.cm Type error at line 6: expected integer type index, got IntegerArray type index Type error at line 6: type conflict in assignment
Invalid return type		C-MINUS COMPILATION: ./case_04.cm Type error at line 8: invalid return type (non-void return value in void type function) Type error at line 6: return statement is missing or not properly stated in this funciton Type error at line 13: invalid return type (return value should be Integer) Type error at line 11: return statement is missing or not properly stated in this funciton Type error at line 21: return statement is missing or not properly stated in this funciton

Multiple semantic errors		C-MINUS COMPILATION: ./case_mult.cm Symbol error at line 12: symbol already declared for current scope Symbol error at line 18: variable should have non-void type Symbol error at line 20: using undeclared symbol Symbol error at line 30: using undeclared symbol Symbol error at line 45: using undeclared symbol Type error at line 5: return statement is missing or not properly stated in this function Type error at line 7: invalid return type (return value should be Integer) Type error at line 18: declaration of void or void array type variable is invalid Type error at line 20: operand of Relop should be Integer type Type error at line 20: type conflict in assignment Type error at line 34: invalid function call (# of arguments does not match) Type error at line 38: invalid function call (argument type mismatched) Type error at line 43: operand of Op should be Integer type Type error at line 43: type conflict in assignment Type error at line 45: invalid expression (if-condition must be Integer type) Type error at line 46: invalid expression (while-condition must be Integer type)