Project 3

Semantic Analyzer

2013011509 오승준

1. 과제 개요

간단한 언어인 C-minus 언어를 기반으로 하는 Semantic Analyzer를 구현

**Code의 길이가 길어 전부 캡쳐하지 못했습니다.

2. 개발환경

1. Linux-16.04 LTS
2. Using GCC

3. 코드분석

- symtab.h -

```
typedef struct BucketListRec
   { char * name;
     LineList lines;
     TreeNode *treeNode;
     int memloc ; /* memory location for variable */
     struct BucketListRec * next;
   } * BucketList;


typedef struct ScopeRec
   { char * funcName;
     int nestedLevel;
     struct ScopeRec * parent;
     BucketList hashTable[SIZE]; /* the hash table */
   } * Scope;
```

사진_1 각 Scope 별로 사용할 구조체

사진_1과 같이 symbol table을 구성하고 scope에 사용되어질 구조체 들을 선언해준다.

```
void st_insert( char * name, int lineno, int loc, TreeNode * treeNode );
int st_lookup ( char * name );
int st_add_lineno(char * name, int lineno);
BucketList st_bucket( char * name );
int st_lookup_top (char * name);

Scope sc_create(char *funcName);
Scope sc_top( void );
void sc_pop( void );
void sc_push( Scope scope );
int addLocation( void );
```

사진_2 symbol table을 만드는데 사용되는 함수들

Hash table을 구성하고 symbol table을 만드는데 사용되는 함수들을 선언해준다.
크게 symbol table을 관리하는 함수 부분과, scope를 관리하는데 사용하는 함수들로 나뉜다.

구성하는데 사용되는 자료구조로는 stack을 이용하여 관리에 용이하도록 하였다.

- analize.c -

```
static void insertNode( TreeNode * t)
{ switch (t->nodekind)
  { case StmtK:
      switch (t->kind.stmt)
      { case CompK:
          if (preserveLastScope) {
            preserveLastScope = FALSE;
          } else {
            Scope scope = sc_create(funcName);
            sc_push(scope);
          }
          t->attr.scope = sc_top();
          break;
        default:
          break;
      }
      break;
    case ExpK:
```

사진_3 insertNode 함수

TreeNode의 구조체의 주소값을 받아 안에 저장되어있는 값들을 기반으로 symbol table에 넣어주는 함수이다.

사진_3 처럼 각각 해당하는 일을 수행하도록 switch문을 사용하여 알맞는 조건의 code를 시행하도록 하였다.

```
static void checkNode(TreeNode * t)
{ switch (t->nodekind)
  { case StmtK:
      switch (t->kind.stmt)
      { case CompK:
          sc_pop();
          break;
        case IterK:
          if (t->child[0]->type == Void)
          /* while test should be void function call */
            typeError(t->child[0],"while test has void value");
          break;
        case RetK:
          { const TreeNode * funcDec =
                st_bucket(funcName)->treeNode;
            const ExpType funcType = funcDec->type;
            const TreeNode * expr = t->child[0];

            if (funcType == Void &&
```

사진_4
chechNode 함수

insertNode의 함수의 수행이 끝나고 symbol table이 다 만들어 진 후, type checking을 할때 사용되는 함수이다. 마찬가지로 TreeNode의 구조체 주소값을 받아 그것을 기반으로 type checking을 하도록 하였다.

4. 실행결과

```
CMINUS COMPILATION: gcd.cm

Building Symbol Table...

Symbol table:

<FUNCTIONS AND GLOBAL VARIABLES>
Scope name : ~:global
nested Lev : 0
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
main           Void           3         11
input          Integer        0         0   15   16
output         Void           1         0   17
gcd            Integer        2         4    7   17
----------------------------------------------------

<FUNCTION PARAMETER AND LOCAL VARIABLES>
Scope name : ~:gcd
nested Lev : 1
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
u              Integer        0         4    6    7    7
v              Integer        1         4    6    7    7    7
----------------------------------------------------

Scope name : ~:main
nested Lev : 1
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
x              Integer        0         13   15   17
y              Integer        1         14   16   17
----------------------------------------------------


Checking Types...

Type Checking Finished
```

사진_5
gcd.cm의 실행결과

처음에 global scope를 감지하여 현재 사용될 함수들과 각 함수들의 타입 그리고 location 및 몇 번째 줄에 있는지를 출력해준다.

그 후, 각 함수들의 parameter과 각 함수 안에서 쓰일 local variable에 대한 타입과 location 및 몇번째 줄에 있는지를 출력해준다.

global은 nested level이 0
Local 함수들은 기본 1로 시작하며 그 안에서 또다른 scope가 생성시, nested level이 증가하게 된다.

Stmbol table이 다 만들어지면 이어서 type checking을 하게되고 이때 별 문제가 없으면 Type Checking Finished란 문구를 출력하고 종료한다.

```
1  /* A program to perform Euclid's
2  Algorithm to computer gcd */
3
4  void a;
5
6  int gcd (int u, int v)
7  {
8      here = 1;
9      if (v == 0) return u;
10     else return gcd(v,u-u/v*v);
11     /* u-u/v*v == u mod v */
12 }
13 void main(void)
14 {
15     int x; int y;
16     x = input(); y = input();
17     output(gcd(x,y));
18     dummy();
19 }
```

사진_6
오류가 있는 테스트 코드 - gcd.cm

에러 체크는 크게 두가지로 나뉘는데,
선언시 잘못했을 경우와,
Symbol table을 만들고 난 후
Type checking에서 문제가 발생하는 경우이다.

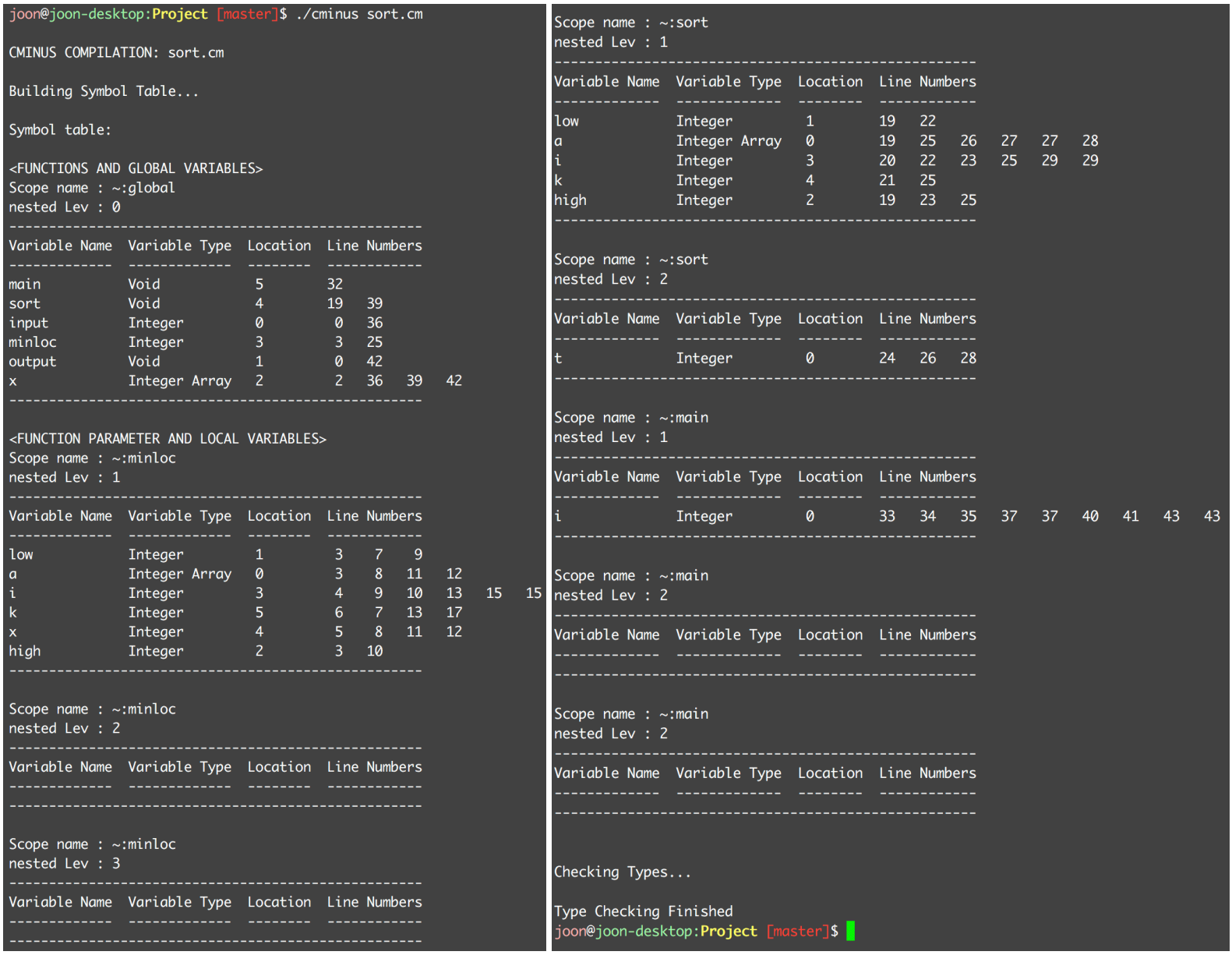
사진_6은 사진_5의 gcd.cm의 코드에서 선언시 잘못 선언한 경우로써
4번줄에 void 형 변수선언
8번줄에 선언을 안한 변수 사용
18번줄에 선언을 안한 함수 사용
의 잘못된 선언을 하였다.

```
CMINUS COMPILATION: test.cm

Building Symbol Table...
Symbol error at line 4: variable should have non-void type
Symbol error at line 8: undelcared symbol
Symbol error at line 18: undelcared symbol
```

사진_7
사진_6의 code를 실행한 결과

사진_7은 사진_6의 결과로써 symbol table을 만드는 과정중에 4번 8번 18번에 문제가 있음을 알려주고 어떤 문제인지 출력해준다.



```
joon@joon-desktop:Project [master]$ ./cminus sort.cm

CMINUS COMPILATION: sort.cm

Building Symbol Table...

Symbol table:

<FUNCTIONS AND GLOBAL VARIABLES>
Scope name : ~:global
nested Lev : 0
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
main           Void           5        32
sort           Void           4        19   39
input          Integer        0         0   36
minloc         Integer        3         3   25
output         Void           1         0   42
x              Integer Array  2         2   36   39   42
----------------------------------------------------

<FUNCTION PARAMETER AND LOCAL VARIABLES>
Scope name : ~:minloc
nested Lev : 1
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
low            Integer        1         3    7    9
a              Integer Array  0         3    8   11   12
i              Integer        3         4    9   10   13   15   15
k              Integer        5         6    7   13   17
x              Integer        4         5    8   11   12
high           Integer        2         3   10
----------------------------------------------------

Scope name : ~:minloc
nested Lev : 2
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
----------------------------------------------------

Scope name : ~:minloc
nested Lev : 3
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
----------------------------------------------------
Scope name : ~:sort
nested Lev : 1
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
low            Integer        1         19   22
a              Integer Array  0         19   25   26   27   27   28
i              Integer        3         20   22   23   25   29   29
k              Integer        4         21   25
high           Integer        2         19   23   25
----------------------------------------------------

Scope name : ~:sort
nested Lev : 2
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
t              Integer        0         24   26   28
----------------------------------------------------

Scope name : ~:main
nested Lev : 1
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
i              Integer        0         33   34   35   37   37   40   41   43   43
----------------------------------------------------

Scope name : ~:main
nested Lev : 2
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
----------------------------------------------------

Scope name : ~:main
nested Lev : 2
----------------------------------------------------
Variable Name  Variable Type  Location  Line Numbers
-------------  -------------  --------  ------------
----------------------------------------------------


Checking Types...

Type Checking Finished
joon@joon-desktop:Project [master]$
```

사진_9 sort.cm

사진_9는 sort.cm의 실행결과로써, nested하게 짜여져있는 scope들을 잘 찾아내어 각 scope 별로 local variable을 저장하여 symbol table을 성공적으로 만드는것을 볼 수 있다.

```
1  int x;
2  int a[5];
3
4  void funcA(void) {
5      int a;
6      a = input();
7      output(a);
8      return a;
9  }
10
11 void funcB(int a) {
12     int x[4];
13     output(funcA(a,x));
14     x = a;
15 }
16
17 int funcC(int a, int b[]) {
18     b = 2;
19     a = funcB(a);
20     funcB(b);
21     return b;
22 }
23
24 int funcD(int a, int b, int c[]) {
25     funcB(a,b);
26     funcA(a);
27     funcB(funcD(a,c));
28     dum();
29     return a;
30 }
```

사진_10
다양한 오류가 있는 test code

사진_10은 다양한 오류가 존재하는 test code로써

28번째 줄에 선언을 안한 변수 사용

Type checking 에서

8번째 줄에 return 에러(void)
13번째 줄에 parameter 에러
14번째 줄에 array 에러
18번째 줄에 parameter 에러
19번째 줄에 array 에러
21번째 줄에 return 에러(array 반환불가)
25번째 줄에 parameter 에러
27번째 줄에 parameter 에러
29번째 줄에 return 에러

과 같은 에러가 존재한다.

```
Building Symbol Table...
Symbol error at line 28: undelcared symbol
```

사진_11
28번째 줄 에러

```
Checking Types...
Type error at line 8: return value should not exist
Type error at line 13: the number of parameters is wrong
Type error at line 13: void value cannot be passed as an argument
Type error at line 14: assignment to array variable
Type error at line 18: assignment to array variable
Type error at line 19: assignment to array variable
Type error at line 21: wrong return value type
Type error at line 25: the number of parameters is wrong
Type error at line 27: the number of parameters is wrong
Type error at line 29: wrong return value type
```

사진_12
나머지 type checking 에러

사진_11과 사진_12에서 보이는것 처럼 에러들을 검출하였다.