Compiler Project #3 Semantic Analysis

Software Engineering

2015004120 Park Seonha

1. Goal of Project#3

- Implement Symbol table and Type Checker
- Traverse syntax tree created by parser

2. Environment

OS : Ubuntu 16.04

Environment: GCC(5.4.0), flex(2.6.0), bison(3.0.4), GNU Make(4.1)

Executable : cminus (./loucomp/cminus)

3. Implementation

* symtab.h

- Move definition of LineList, BucketList from symtab.c. Add treeNode in BucketList struct. To modify symbol table, add Treenode as parameter of st_insert() function, add st_add_lineno() function and st_bucket(), st_exist_top().
- Add Scope struct and function scope_create() to create scope.

Scope struct has name, hash table named bucket, its parent scope address, depth of scope, and scopeLoc to save location of variable in same scope.

- Add functions for generate and manage scope stack (pop, push, get stack top scope)

* symtab.c

- Defined MAX_SCOPE(max number of scope) as 1000
- Add scopeStack and scopeExist (+ stack top variables of each stack) which can store depth of scope and all scope in program.

(scope functions)

- scope_create() : get name of scope as parameter, create scope and insert scopeExist list.
- scope_push() : get scope as parameter, push scope to scopeStack.
- scope_pop() : get endline of scope as parameter. if endline exist(in analyze.c, if scope doesn’t need to store or print its endline, send -1 as parameter of scope_pop), add endline into the name of scope. For example, if the name of scope is “main” and endline is 16, this function changes the name of scope “main:16”. and pop the scope from scopeStack.
- scope_top() : return the top of scopeStack.

(symbol table)

- st_bucket() : get name of scope as parameter, return the bucketlist of scope.
- st_insert() : add treeNode as parameter, if variable is not in symbol table, create BucketList and insert symbol table. If exists in symbol table, run st_add_lineno() instead of st_insert()
- st_lookup() : find the location of variable in scope.
- st_exist_top() : check the variable is in top scope of scopeStack.
- st_add_lineno() : get name of variable and its line number as parameter,
- printSymTab() : change printing format of symbol table to fit project. (add location , variable type)

* analyze.c

- insertIOFunc() - generate TreeNode for input() and and output() function, and insert into symbol table as global scope symbol. Add output function scope in scopeExist list.
- insertNode() – change nodekind to fit cminus grammer, change the way to append line number.
- afterInsertNode() – pop scope at scopeStack if node is compound statement node or function node.
- buildSymtab() : Create global scope, first push global scope and insert input and output into global scope. After traverse, pop global scope.
- beforeCheckNode() : If node is function declaration, change scopename to that function name. if node is compound statements, push scope into scopeStack.
- checkNode() - change nodekind to fit cminus grammer and project description ‘type checker’ page.
- typeCheck() – Like buildSymtab(), create global scope. Push global scope into scopeStack and insert input and output into global scope. After traverse, pop global scope.

* main.c

Change NO_ANALYZE, NO_CODE to semantic analysis, TraceParse, TraceAnalyze to print Symbol table.

* globals.h

Change ScopeRec to ScopeListRec in struct TreeNode.

* Makefile

Add analyze.o to OBJS.

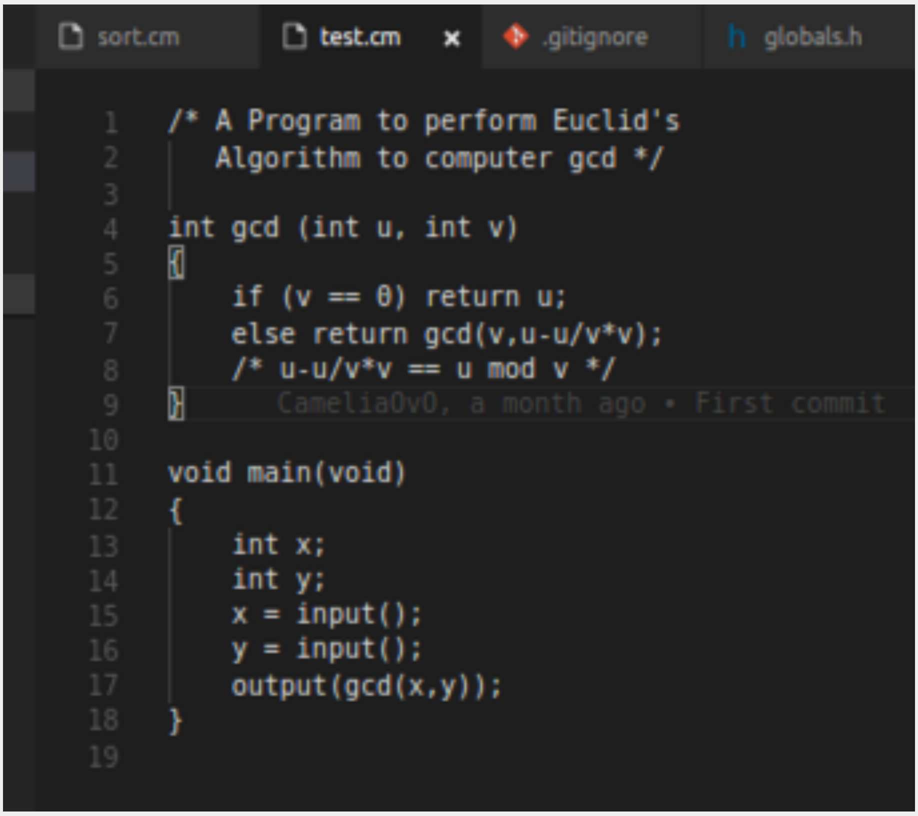
4. Compilation

Makefile is modified for symbol table.

To compile, use ‘**make**’ or ‘**make cminus**’ to generate executable.

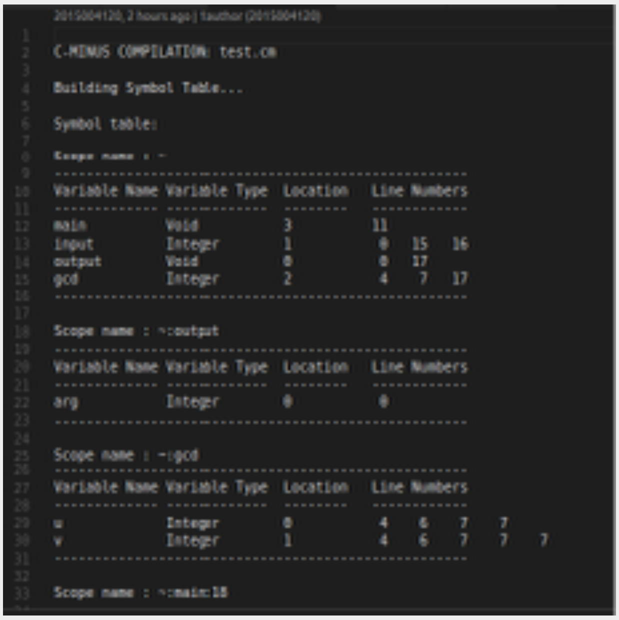
5. Result Screenshot

test.cm

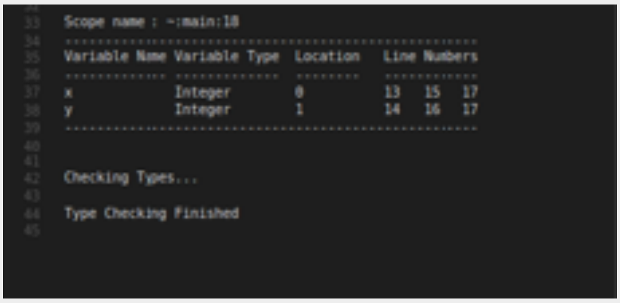


```
/* A Program to perform Euclid's
   Algorithm to computer gcd */

int gcd (int u, int v)
{
    if (v == 0) return u;
    else return gcd(v,u-u/v*v);
    /* u-u/v*v == u mod v */
}

void main(void)
{
    int x;
    int y;
    x = input();
    y = input();
    output(gcd(x,y));
}

```

semantic analysis result

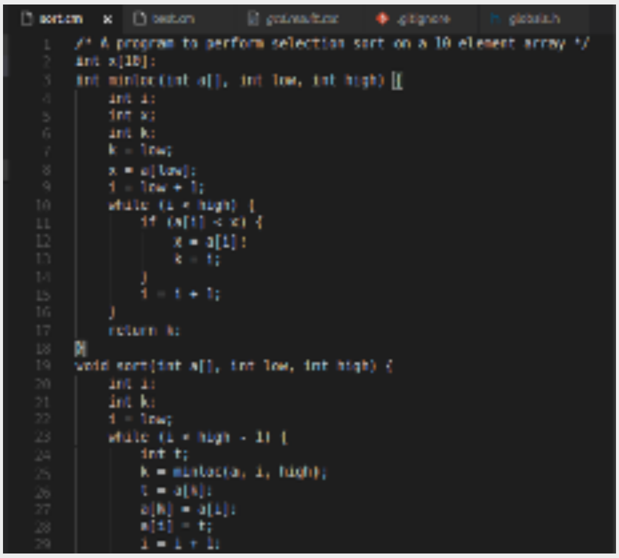


```

C-MINUS COMPILATION: test.cm

Building Symbol Table...

Symbol table:

Scope name : ~
------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
main          Void           3          11
input         Integer        1           0   15   16
output        Void           0           0   17
gcd           Integer        2           4    7   17
------------------------------------------------------

Scope name : ~:output
------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
arg           Integer        0           0
------------------------------------------------------

Scope name : ~:gcd
------------------------------------------------------
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:18
```

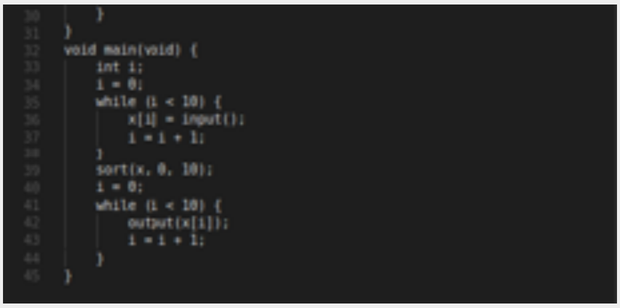


```
Scope name : ~:main:18
------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
x             Integer        0          13   15   17
y             Integer        1          14   16   17
------------------------------------------------------


Checking Types...

Type Checking Finished

```

sort.cm

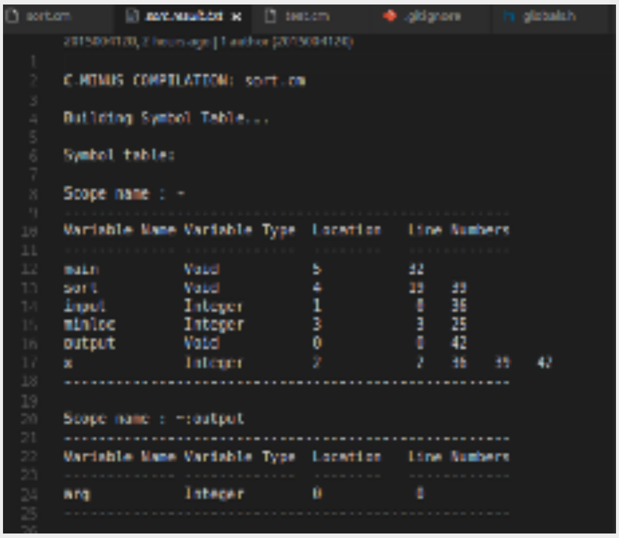


```
/* A program to perform selection sort on a 10 element array */
int x[10];
int minloc(int a[], int low, int high) {
    int i;
    int x;
    int k;
    k = low;
    x = a[low];
    i = low + 1;
    while (i < high) {
        if (a[i] < x) {
            x = a[i];
            k = i;
        }
        i = i + 1;
    }
    return k;
}
void sort(int a[], int low, int high) {
    int i;
    int k;
    i = low;
    while (i < high - 1) {
        int t;
        k = minloc(a, i, high);
        t = a[k];
        a[k] = a[i];
        a[i] = t;
        i = i + 1;
```

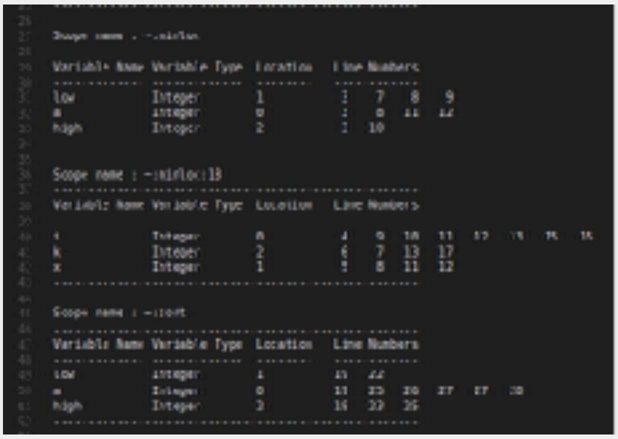


```
        }
}
void main(void) {
    int i;
    i = 0;
    while (i < 10) {
        x[i] = input();
        i = i + 1;
    }
    sort(x, 0, 10);
    i = 0;
    while (i < 10) {
        output(x[i]);
        i = i + 1;
    }
}
```

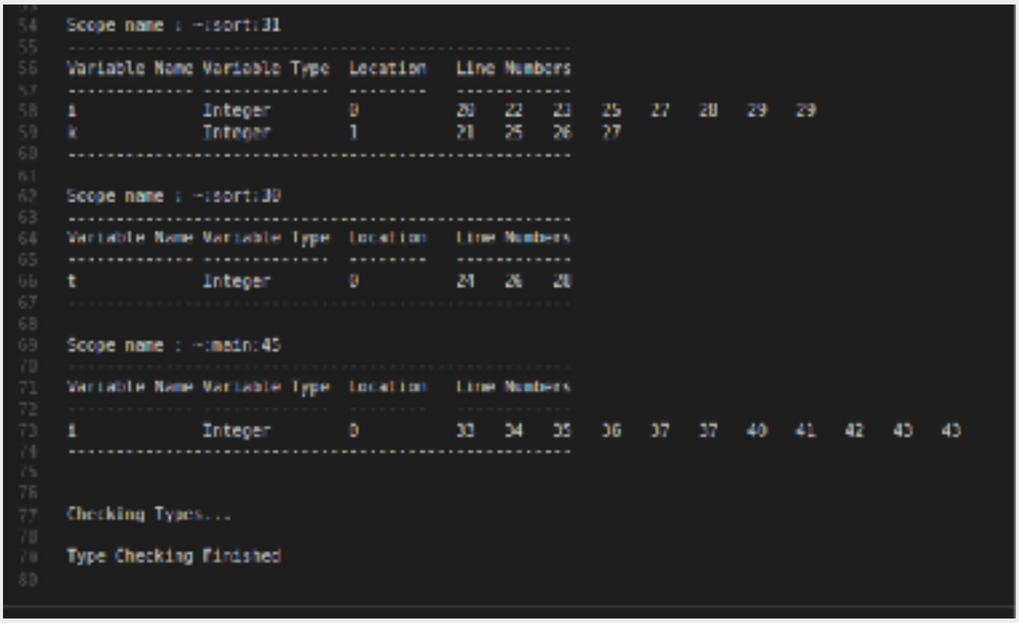
semantic analysis result (sort.cm)



```

C-MINUS COMPILATION: sort.cm

Building Symbol Table...

Symbol table:

Scope name : ~
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
main          Void           5          32
sort          Void           4          19   39
input         Integer        1           0   36
minloc        Integer        3           3   25
output        Void           0           0   42
x             Integer        2           2   36   39   42
------------------------------------------------------------

Scope name : ~:output
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
arg           Integer        0           0
------------------------------------------------------------
```



```
------------------------------------------------------------

Scope name : ~:minloc

Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
low           Integer        1           3    7    8    9
a             Integer        0           3    8   11   12
high          Integer        2           3   10


Scope name : ~:minloc:18
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers

i             Integer        0           4    9   10   11   12   13   15   15
k             Integer        2           6    7   13   17
x             Integer        1           5    8   11   12
------------------------------------------------------------

Scope name : ~:sort
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
low           Integer        1          19   22
a             Integer        0          19   25   26   27   27   28
high          Integer        2          19   23   25
------------------------------------------------------------
```



```
Scope name : ~:sort:31
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
i             Integer        0          20   22   23   25   27   28   29   29
k             Integer        1          21   25   26   27
------------------------------------------------------------

Scope name : ~:sort:30
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
t             Integer        0          24   26   28
------------------------------------------------------------


Scope name : ~:main:45
------------------------------------------------------------
Variable Name Variable Type  Location   Line Numbers
------------- -------------  --------   ------------
i             Integer        0          33   34   35   36   37   37   40   41   42   43   43
------------------------------------------------------------



Checking Types...

Type Checking Finished

```