Project #3. Semantic

컴퓨터소프트웨어학부 2017029589 류지범

#0 Environment & Execute

- Environment
  - Ubuntu 18.04.02 LTS
- Execute
  - Makefile로 컴파일한 후 다음과 같이 실행한다.
  - `./cminus_semantic input.txt`

#1 기본 세팅

- `main.c`
  - 테스트할 때는 `TraceAnalyze`를 TRUE로 세팅했고, 제출 코드는 `FALSE`로 설정했다.
- `globals.h`
  - Integer Array 타입을 분석하기 쉽게 하기 위해 `ExpType`에 `IntArray`를 추가했다.

```
typedef enum {Void,Integer, IntArray} ExpType;
```

#2 symtab.c & symtab.h (Print & Manage Symbol Table)

- `symtab.c`와 s`ymtab.h`는 Symbol Table을 만드는데 사용된다.
- 기존의 tiny의 코드의 틀을 유지한채로 필요한 부분을 추가했다.
- scope와 bucket을 관리하기 위한 구조체를 symtab.h에 정의했다.

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

typedef struct ScopeListRec
    { char * scopeName;
      int nestedLevel;
      BucketList bucket[SIZE];
      struct ScopeListRec * parent;
    } * ScopeList;
```

- scope는 hierarchical하게 구성되는데, 항상 현재의 scope정보를 가지고 있어야 하므로 스택으로 관리하도록 했다.
  - scope를 create, push, pop할 수 있는 함수를 각각 만들어주었다.

```
void sc_push(ScopeList scope);

void sc_pop(void);

ScopeList sc_top(void);

ScopeList sc_create(char *scopeName);
```

- 현재 scope를 탐색하는 함수와, scope를 타고 따라 올라가면서 원하는 bucket을 찾는 함수를 추가했다.

```
int st_lookup ( char * name );

BucketList st_lookup_bucket(char *name);
```

- symbol의 위치(line)를 정해주는 함수도 추가했다.

```
void st_add_lineno(char *name, int lineno);
```

- symbol table 출력을 위한 함수들을 추가했다. 각각 < Symbol Table >, < Functions >, < Global Symbols >, < Scopes >를 출력하는 함수이다.

```
void printST(FILE *listing);

void printFuncTab(FILE *listing);

void printGlobalSym(FILE *listing);

void printScope(FILE *listing);
```

#3 analyze.c & analyze.h (Type Checking & Build Symbol Table)

- `analyze.c`와 `analyze.h`는 AST를 기반으로 Symbol Table을 생성하고, 만들어진 Symbol Table과 AST를 이용해서 Type Checking을 수행한다.
- Error Function들을 새로 정의해줬다. (Redifined, Undeclared, voidVariableDeclaration)

```
static void voidValDeclError(TreeNode *t, char *name);

static void undeclaredError(TreeNode *t, char *name);

static void redefinedError(TreeNode *t, char *name);
```

- 각 scope마다 Symbol Table에 insert가 완료되면 scope를 pop해줘야하므로 function pointer로 사용될 함수를 정의해줬다.

```
static void afterInsertNode(TreeNode *t);
```

- Type checking을 하기 전 scope를 넣고, function이름과, return 상태를 체크하기 위해 function pointer로 사용될 함수를 정의해줬다.

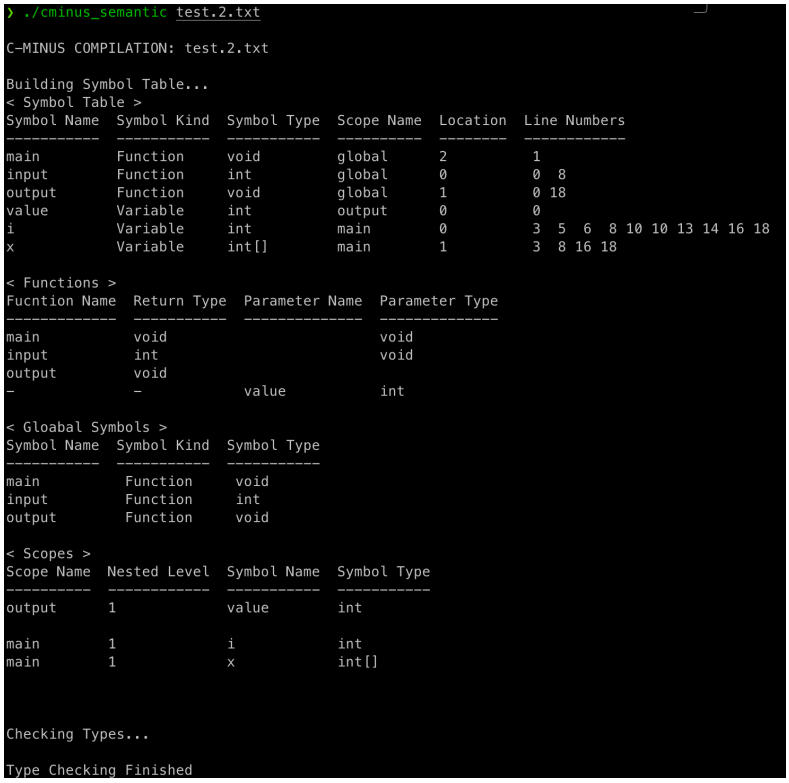
```
static void beforeCheckNode(TreeNode *t);
```

- Built-In function인 int input(void)와 void output(int value)를 추가해주기 위한 함수 `pushBuiltInFunc`을 정의해줬다.
    - 각각 타입에 맞는 정보를 할당해주고 symbol table에 insert했다.
    - output의 경우 value를 parameter로 가지기 때문에 output이라는 scope를 정의해준 후 output scope를 집어넣고, value를 집어넣은 후 scope를 pop해주었다.
- Symbol table를 만드는 작업은 `static void insertNode( TreeNode * t)`로 수행했다. scope를 만들고 스택에 넣는 작업을 하도록 했다. symbol table을 찾아보며 Redefined error를 잡도록했다.
- Type Checking은 `static void checkNode(TreeNode * t)`로 수행했다. checking을 진행한 부분은 다음과 같다.
    - Void type은 선언할 수 없도록 했다. (Void array, void variable)
    - void type은 assign될 수 없도록 했다.
    - int array는 int variable에 assign될 수 없고, int variable 역시 int array에 assign될 수 없다. (indexing을 통해서만 가능하도록 했다)
    - operater의 operand가 type이 다를 경우 assign될 수 없도록 했다.
    - Function call에서 argument와 paremter의 개수가 다를 경우 error를 출력하도록 했다.
    - function call에서 argument의 type과 parameter의 type이 불일치할 경우 error를 출력하도록 했다.
    - argument는 void가 될 수 없도록 했다.
    - 조건문의 condition type에는 int만 들어올 수 있도록 했다.
    - array의 index에는 integer type만 들어올 수 있도록 했다.
    - Void function은 return이 없어도 되지만 int function은 return문이 존재하도록 했다.
    - function의 return type과 return문의 type이 일치하지 않으면 error를 출력하도록 했다.
        - Array type의 function은 구현하지 않았으므로 array type을 return하지 못하도록 했다.

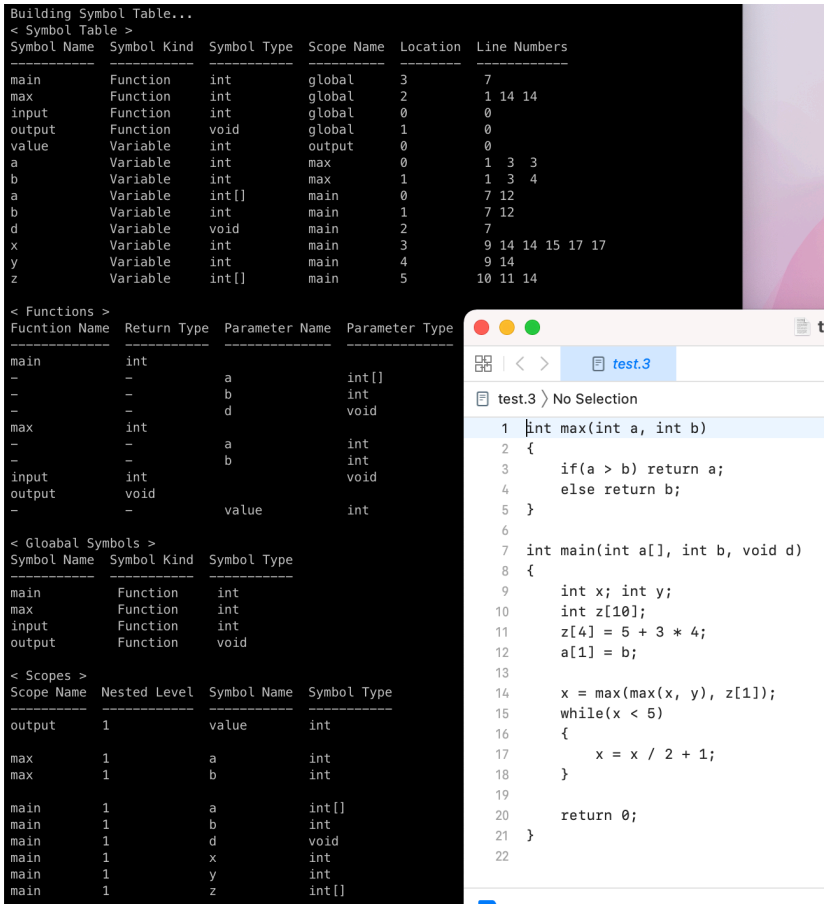
#4 Test Case & Result

- 주어진 test case에 대한 symbol table은 다음과 같다.
- test.1.txt
  -

```
❯ ./cminus_semantic test.1.txt

C-MINUS COMPILATION: test.1.txt

Building Symbol Table...
< Symbol Table >
Symbol Name  Symbol Kind  Symbol Type  Scope Name  Location  Line Numbers
-----------  -----------  -----------  ----------  --------  ------------
main         Function     void         global      3         11
input        Function     int          global      0          0 14 14
output       Function     void         global      1          0 15
gcd          Function     int          global      2          4  7 15
value        Variable     int          output      0          0
u            Variable     int          gcd         0          4  6  7  7
v            Variable     int          gcd         1          4  6  7  7  7
x            Variable     int          main        0         13 14 15
y            Variable     int          main        1         13 14 15

< Functions >
Fucntion Name  Return Type  Parameter Name  Parameter Type
-------------  -----------  --------------  --------------
main           void                         void
input          int                          void
output         void
-              -            value           int
gcd            int
-              -            u               int
-              -            v               int

< Gloabal Symbols >
Symbol Name  Symbol Kind  Symbol Type
-----------  -----------  -----------
main          Function     void
input         Function     int
output        Function     void
gcd           Function     int

< Scopes >
Scope Name  Nested Level  Symbol Name  Symbol Type
----------  ------------  -----------  -----------
output      1             value        int

gcd         1             u            int
gcd         1             v            int

main        1             x            int
main        1             y            int



Checking Types...

Type Checking Finished
```

- test.2.txt
  - 

```
❯ ./cminus_semantic test.2.txt

C-MINUS COMPILATION: test.2.txt

Building Symbol Table...
< Symbol Table >
Symbol Name  Symbol Kind  Symbol Type  Scope Name  Location  Line Numbers
-----------  -----------  -----------  ----------  --------  ------------
main         Function     void         global      2          1
input        Function     int          global      0          0  8
output       Function     void         global      1          0 18
value        Variable     int          output      0          0
i            Variable     int          main        0          3  5  6  8 10 10 13 14 16 18
x            Variable     int[]        main        1          3  8 16 18

< Functions >
Fucntion Name  Return Type  Parameter Name  Parameter Type
-------------  -----------  --------------  --------------
main           void                         void
input          int                          void
output         void
-              -            value           int

< Gloabal Symbols >
Symbol Name  Symbol Kind  Symbol Type
-----------  -----------  -----------
main          Function     void
input         Function     int
output        Function     void

< Scopes >
Scope Name  Nested Level  Symbol Name  Symbol Type
----------  ------------  -----------  -----------
output      1             value        int

main        1             i            int
main        1             x            int[]



Checking Types...

Type Checking Finished
```

- 개인적으로 만든 test case에 대한 symbol table은 다음과 같다.
- test.3.txt
  - 

```
Building Symbol Table...
< Symbol Table >
Symbol Name  Symbol Kind  Symbol Type  Scope Name  Location  Line Numbers
-----------  -----------  -----------  ----------  --------  ------------
main         Function     int          global      3          7
max          Function     int          global      2          1 14 14
input        Function     int          global      0          0
output       Function     void         global      1          0
value        Variable     int          output      0          0
a            Variable     int          max         0          1  3  3
b            Variable     int          max         1          1  3  4
a            Variable     int[]        main        0          7 12
b            Variable     int          main        1          7 12
d            Variable     void         main        2          7
x            Variable     int          main        3          9 14 14 15 17 17
y            Variable     int          main        4          9 14
z            Variable     int[]        main        5         10 11 14

< Functions >
Fucntion Name  Return Type  Parameter Name  Parameter Type
-------------  -----------  --------------  --------------
main           int
-              -            a               int[]
-              -            b               int
-              -            d               void
max            int
-              -            a               int
-              -            b               int
input          int                          void
output         void
-              -            value           int

< Gloabal Symbols >
Symbol Name  Symbol Kind  Symbol Type
-----------  -----------  -----------
main          Function     int
max           Function     int
input         Function     int
output        Function     void

< Scopes >
Scope Name  Nested Level  Symbol Name  Symbol Type
----------  ------------  -----------  -----------
output      1             value        int

max         1             a            int
max         1             b            int

main        1             a            int[]
main        1             b            int
main        1             d            void
main        1             x            int
main        1             y            int
main        1             z            int[]
```

```
test.3
test.3 〉 No Selection
 1  int max(int a, int b)
 2  {
 3      if(a > b) return a;
 4      else return b;
 5  }
 6
 7  int main(int a[], int b, void d)
 8  {
 9      int x; int y;
10      int z[10];
11      z[4] = 5 + 3 * 4;
12      a[1] = b;
13
14      x = max(max(x, y), z[1]);
15      while(x < 5)
16      {
17          x = x / 2 + 1;
18      }
19
20      return 0;
21  }
22
```

- semantic error에 대한 예시는 다음과 같다.
  - redefined error & different number of arg and param
    -

```
int foo(int n)
{
    return n + 1;
}
int a;
int a;
/* wrong # of parameter <-> argument */
int main(void)
{
    int a;
    int b;

    return foo(a,b);
}
```

```
❯ ./cminus_semantic tc1.txt
                ⎵┘


C-MINUS COMPILATION: tc1.txt
Error: Redefined Symbol "a" at line 6
Type error at line 13: invalid function call: not mathing # of args <->
params
```

  - different type of arg and param
    -

```
int foo(int n)
{
    return n + 1;
}
/* wrong type of parameter <-> argument */
int main(void)
{
    int a[5];

    return foo(a);
}
```

```
❯ ./cminus_semantic tc2.txt
                ⎵┘


C-MINUS COMPILATION: tc2.txt
Type error at line 10: invalid function call: not mathing type(arg <->
param))
```

- void variable & void array
  - ```
    /* void var & arr */
    void a;
    void A[5];
    int main(void)
    {
        return 0;
    }
    ```

    ```
    ❯ ./cminus_semantic tc3.txt
                    ⎵┘

    C-MINUS COMPILATION: tc3.txt
    Error: Variable Type cannot be Void at line 2 (name : a)
    Error: Variable Type cannot be Void at line 3 (name : A)
    Error: Variable Type cannot be Void at line 2 (name : a)
    Error: Variable Type cannot be Void at line 3 (name : A)
    ```

- incompatible type error
  - ```
    int main(void)
    {
        int a;
        int b[5];
        int arr[2];

        /* incompatible type */
        int c;
        c = a + b;
        b = a;
        a = b;

        /* Ok */
        a = b[1];

        /* incompatible type */
        a = b + b;

        return 0;
    }
    ```

    ```
    ❯ ./cminus_semantic tc4.txt
                    ⎵┘

    C-MINUS COMPILATION: tc4.txt
    Type error at line 9: array can't be operand
    Type error at line 9: void can't be assigned
    Type error at line 10: var can't assigned array, array can't assigned var
    Type error at line 11: var can't assigned array, array can't assigned var
    Type error at line 17: array can't be operand
    Type error at line 17: void can't be assigned
    ```

- wrong loop condition
  -

```
1  void foo(int a)
2  {
3  }
4
5  int main(void)
6  {
7      int A[5];
8      int b;
9
10     /* wrong loop condition */
11     if (foo(5)) return 1;
12
13     if (A) return 2;
14
15     while(A) b = b + 1;
16     return 0;
17 }
```

```
❯ ./cminus_semantic tc5.txt
                ⌟


C-MINUS COMPILATION: tc5.txt
Type error at line 11: condition type must be int
Type error at line 13: condition type must be int
Type error at line 15: condition type must be int
```

- invalid assign & invalid return
  -

```
1  void foo(int n)
2  {
3  }
4  int main(void)
5  {
6      int a;
7
8      /* invalid assign */
9      a = foo(5);
10
11     /* invalid return */
12     return;
13 }
```

```
❯ ./cminus_semantic tc6.txt
                ⌟


C-MINUS COMPILATION: tc6.txt
Type error at line 9: void can't be assigned
Type error at line 12: non-void function has return val
```

- no return statement
  -
```
int main(void)
{
    int a;
}
```

```
❯ ./cminus_semantic tc7.txt
                ⎵┘


C-MINUS COMPILATION: tc7.txt
Type error at line 1: int function need return stmt
```

- Indice type error (not integer)
  -
```
int main(void)
{
    /* void variable */
    void c;
    int a[5];
    int b[4];
    int arr[2];
    /* array type index & void type index error */
    arr[1] = b[c];
    arr[2] = b[a];
    return 0;
}
```

```
❯ ./cminus_semantic tc8.txt
                      ⎵┘


C-MINUS COMPILATION: tc8.txt
Error: Variable Type cannot be Void at line 4 (name : c)
Error: Undelcared Symbol "c" at line 9
Error: Variable Type cannot be Void at line 4 (name : c)
Type error at line 9: indices must be integer type
Type error at line 9: void can't be assigned
Type error at line 10: indices must be integer type
Type error at line 10: void can't be assigned
```