Compiler Project03_Semantic_Analyzer Report

2015004693_양상헌

Goal:

- **Implementation of C-MINUS Semantic Analyzer**

Design Analysis:

- **Scope & ScopeTable :**

Scope는 C-Minus Semantic Analyzer를 구현함에 있어서 배우 중요한 개념이다. 기존의 tiny 에서의 Default 상태에서는 Scope가 구현되어 있지 않기에 새롭게 구현해야 했다. Scope 내부에는 현재 Scope에서 선언된 함수 혹은 변수의 정보를 저장하는 Bucket 이 존재하고, 이 Bucket 들을 Look up 해서 호출되는 함수 혹은 변수가 미리 선언이 되어있는 지를 확인할 때 사용하게 된다.

함수 선언, 혹은 함수 선언이 아닌 { } 로 이루어진 Compound Statement 가 생성될 때마다 새로운 Scope가 생성되게 되고, Scope 내부에서 또다른 새로운 Scope가 생성이 될 때, 원래의 Scope는 새롭게 생성되는 Scope의 Parent Scope가 된다.

Scope는 기본적으로 Scope의 이름, Scope의 ID(number), BucketList, NestedLevel(Depth), Parent Scope등의 정보를 가지고 있어야 하고, 함수가 선언되거나 { } 로 이루어진 Compound Statement 가 생성될 때마다 다른 Scope로 이동하는 것이 잘 구현돼야 한다.

- **Bucket & BuckeList :**

Bucket은 특정 Scope 에서 선언된 함수 혹은 변수의 정보를 가지고 있는 구조체이다. 함수 혹은 변수의 이름, 선언 혹은 호출된 위치(Line Number) 등의 정보를 가지고 있다. 또한 Linked-List 형태로 구현이 되어있어서 새로운 함수 혹은 변수가 선언될 때마다 Bucket을 새로 생성하고 기존의 BucketList의 Head에 새로운 Bucket을 insert 해주는 방식으로 사용하였다.

Scope 구조체가 가지고 있는 "Bucket"은 해당 Scope내의 BucketList의 head부분을 포인팅한다.

- **Building Symbol Table & Pre-Order.**

심볼 테이블은 PRE-ORDER TRAVESE로 생성한다.

처음 노드를 만났을 때는 그 노드에 대한 정보를 심볼테이블에 기록해주고 Compound Statement를 만났을 경우 Scope을 새로 생성하고 현재의 Scope를 새로 생성한 Scope로 변경해주는 일을 처리하고, 이후 같은 노드를 재방문 했을 때는 아무런 일을 하지 않고 Compound Statement일 경우에만 Scope을 다시 Parent Scope으로 바꿔주는 일만 처리 해주는 방식으로 심볼테이블을 완성한다.

- **Type Checking & Post-Order.**

Type Checking은 POST-ORDER TRAVESE로 진행한다.

일단 앞서서 만들어진 심볼테이블을 사용하여 Type Checking을 진행하는데, 심볼테이블을 만드는 과정과는 반대의 개념으로 처리하는데, 처음 노드를 만났을 때는 Compound Statement에 한해 현재 Scope를 변경된 Scope로 바꿔주는 작업 외에는 아무런 작업을 수행하지 않고, 이후 같

은 노드를 재방문 했을 때, 그 노드의 Type 혹은 노드의 child 노드들의 Type을 확인하여 Type Checking을 진행한다

Implementation:

["globals.h"]

```
/* ExpType is used for type checking */
typedef enum {Void,Integer,Boolean,IntegerArray,VoidArray,TypeError} ExpType;
```

globals.h 에서 type enum 부분에 int array, void array, type error 타입을 추가해주었다.

["symtab.h"]

```
//--- edited for Project 03 ---//
//void st_insert( char * name, int lineno, int loc );
void st_insert( int ScopeIndex, char * name, int lineno, int loc, ExpType type,TreeNode * node );
//int st_lookup ( char * name );
int st_lookup(int ScopeIndex, char * name);
int st_lookup_excluding_parent(int ScopeIndex, char * name);

/* Procedure printSymTab prints a formatted
 * listing of the symbol table contents
 * to the listing file
 */
void printSymTab(FILE * listing);

typedef struct LineListRec
{ int lineno;
  struct LineListRec * next;
} * LineList;

typedef struct BucketListRec
   { char * name;
     LineList lines;
     int memloc ; /* memory location for variable */
     struct BucketListRec * next;
    //----edited ----// for saving node
       TreeNode * node;
       ExpType type;
    //---------------//
   } * BucketList;

typedef struct ScopeListRec{
    char * name;
    BucketList Bucket;
    int NofBuckets;
    int index; //index of hash table;
    int nestedLevel; //for limiting should-be-checked scopes.
    struct ScopeListRec * parent;
    TreeNode * nodeForFunc;
} * ScopeList;

/* the hash table */
ScopeList ScopeTable[500];
```

기존에 "symtab.c"에 선언되어 있었던 Scope, Bucket, Line 구조체를 모두 옮겨와 선언해주고 조금씩 편집해주었다. 또한 st_lookup(), st_insert() 을 지우고 변경된 함수의 형태로 선언해주었다.

["symtab.c"]

-st_insert()

```
105 //inserting the "var" into Symbol Table
106 void st_insert( int ScopeIndex, char* name, int lineno, int loc, ExpType type, TreeNode * node){
107     if(loc!=0){
108         //case of adding new bucket.
109         //when  not added in Symboltable yet
110         BucketList l = ScopeTable[ScopeIndex]->Bucket;
111         while((l != NULL) && (strcmp(name,l->name) != 0) ){
112             l = l->next;
113         }
114         if(l == NULL){ // if the "var" does not exist in  current Scope make a new Bucket
115             l = (BucketList)malloc(sizeof(struct BucketListRec));
116             l->name = name;
117             l->lines = (LineList) malloc(sizeof(struct BucketListRec));
118             l->lines->lineno = lineno;
119             l->lines->next = NULL;
120             l->memloc = loc;
121             l->type = type;
122             l->node = node;
123             l->next = ScopeTable[ScopeIndex]->Bucket;
124             ScopeTable[ScopeIndex]->Bucket = l;
125             ScopeTable[ScopeIndex]->NofBuckets++;
126         }
127         else { // if the "var" already exist in current scope, just add line number
128             //this will not happen;;
129             LineList t = l->lines;
130             while (t->next != NULL)
131                 t = t->next;
132             t->next = (LineList) malloc(sizeof(struct LineListRec));
133             t->next->lineno = lineno;
134             t->next->next = NULL;
135         }
136     }
137     else{
138         ScopeList S = ScopeTable[ScopeIndex];
139         BucketList l;
140         int n=0;
141         while(S!=NULL){
142             l = S->Bucket;
143             while(n < S->NofBuckets){
144                 if(l!=NULL && strcmp(name, l->name) != 0){
145                     l = l->next;
146                     n++;
147                 }
148                 else if (l!=NULL && strcmp(name, l->name) == 0){
149                     LineList t = l->lines;
150                     while (t->next != NULL)
151                         t = t->next;
152                     t->next = (LineList) malloc(sizeof(struct LineListRec));
153                     t->next->lineno = lineno;
154                     t->next->next = NULL;
155                     //fprintf(listing,"line is well added\n");
156                     return;
157                 }
158             }
159             S = S->parent;
160             n=0;
161         }
162     }
163 }
```

기존의 st_insert() 함수와는 다르게 Scope의 개념을 추가하여 알맞은 Scope에 bucket이 추가되도록 하는 부분을 추가하였다. 또한 Bucket에서 Bucket이 선언되었을 때의 노드 정보를 가질 수 있게 하여 추후에 함수 변수 구분, 타입 구분이 용이하게끔 만들어 주었다.

-st_lookup() & st_lookup_excluding_parent

```
// look up all the scope from current to all ancestors to find whether "var" exists or not
// when using or calling variable;
int st_lookup(int ScopeIndex, char * name){
    ScopeList S = ScopeTable[ScopeIndex]; //set current scope to start lookup
    BucketList l;
    int n = 0;
    while(S != NULL){
        l = S->Bucket;
        while(n < S->NofBuckets){
            if(l!=NULL && strcmp(name, l->name) != 0){
                l = l->next;
                n++;
            }
            else if (l!=NULL && strcmp(name, l->name) == 0){
                return 1;//variable is declared
            }
        }
        S = S->parent;
        n = 0;
    }
    return -1;//variable is not declared
}

//lookup only current scope
//when variable declaration or parameter adding
int st_lookup_excluding_parent(int ScopeIndex, char * name){
    ScopeList S = ScopeTable[ScopeIndex];
    BucketList l = S->Bucket;
    if(l == NULL)
        return -1;
    while( l != NULL && strcmp(name, l->name) != 0){
       l = l->next;
    }
    if(l == NULL) return -1;//variable is not declared

   else return 1;// variable is declared
}
```

Lookup 함수는 같은 이름을 가지는 변수 혹은 함수의 Bucket을 탐색하는 함수이다.

"st_lookup()" 함수는 현재의 Scope에서 탐색을 시작하여 parent Scope들을 모두 탐색하는 함수이다. 이 함수는 변수나 함수를 호출할 때 사용되는데 이때 호출하는 변수나 함수가 미리 선언되어있는지를 확인하여 선언 되어있으면 1, 아니면 -1을 반환한다. "st_lookup_excluding_parent()" 함수는 위 함수와는 다르게 오로지 현재 Scope안에서만 탐색을 진행한다. 이 함수는 변수나 함수 선언, 지역 변수나 함수의 매개변수 선언 시에 사용되고, 위와 마찬가지로 찾는 이름과 같은 변수 혹은 함수가 선언 되어있으면 1, 아니면 -1을 반환한다.

["analyze.c"]

이 소스코드 파일에서 가장 중요한 함수는 insertNode() 와 checkNode() 함수이다. insertNode() 함수는 심볼테이블을 만들 때 Bucket들을 알맞은 Scope에 추가해주는 중요한 역할을 하고, checkNode()는 심볼테이블이 다 만들어지고 나서 각각의 Node들의 타입을 체크해주는 중요한 역할을 한다.

-insertNode()

```
// adding buckets to ScopeTable;
static void insertNode( TreeNode * t){
    switch(t->nodekind){
        case StmtK:
            switch (t->kind.stmt){
                case CompndK: //Scope is changing here
                    if(isInFuncKpre == 1 && funcCompound == 0){//if in funcDecl
                        funcCompound = 1; //just flague on;
                    }
                    else{//just Compund without func Decl
                        //add new scope
                        just_compound++;
                        TotalScopes++;
                        ScopeLevel++;
                        CurScopeIndex = TotalScopes;
                        ScopeTable[CurScopeIndex] = (ScopeList)malloc(sizeof(struct ScopeListRec));
                        ScopeTable[CurScopeIndex]->name = NULL;
                        ScopeTable[CurScopeIndex]->index = CurScopeIndex;
                        ScopeTable[CurScopeIndex]->nodeForFunc = CurrentScope->nodeForFunc;
                        ScopeTable[CurScopeIndex]->nestedLevel = ScopeLevel;
                        ScopeTable[CurScopeIndex]->Bucket = NULL;
                        ScopeTable[CurScopeIndex]->NofBuckets = 0;
                        ScopeTable[CurScopeIndex]->parent = CurrentScope;
                        CurrentScope = ScopeTable[CurScopeIndex];
                        //CurrentScope->returnType = CurrentScope->parent->returnType;
                    }
                    break;
```

FuncK 의 child가 아닌 CompoundK일 때(ex: if, while, 단순 local Scope), 새로운 Scope을 생성해 주고 현재 Scope을 새로 생성된 Scope로 바꿔준다.

```
                case CallK:
                    if(st_lookup(CurScopeIndex, t->attr.name) == 1){//called function is declared
                        //insert line no.
                        st_insert(CurScopeIndex, t->attr.name, t->lineno, 0, t->type, t);
                    }
                    else{//called function is not declared
                        SemanticError(t, "Calling Undeclared Function that does not exist in Symbol Table\n");
                    }
                    break;
```

CallK 일 땐 lookup함수를 통해 함수가 선언되어 있는지를 확인한 후 해당 함수 Bucket에 line no. 를 추가해주고, 선언 되어있지 않을 경우 에러 메세지를 띄운다. .

```
                case VarK:
                    /*
                    if(t->type == Void){
                        // void type variable error
                         SemanticError(t, "Declaration of [Void] type Variable");
                    }*/
                    if(st_lookup_excluding_parent(CurScopeIndex, t->attr.name) == 1){
                        //variable with the same name already exists in Declaring scope
                        //error "duplicated Declaration of variables with same name"
                        SemanticError(t, "Duplicated Declaration of Variables with the Same Name; It will not be added into Symbol
                            Table!");
                    }
                    else{
                        //insert into symtab
                        st_insert(CurScopeIndex, t->attr.name, t->lineno, location++, t->type, t);
                    }
                    break;

                case ArrVarK:
                    /*
                    if(t->type == Void){
                        //voidArray Type variable error
                        SemanticError(t, "Declaration of [Void] type Variable");
                    }*/
                    if(st_lookup_excluding_parent(CurScopeIndex, t->attr.name) == 1){
                        //variable with the same name already exists in Declaring scope
                        //error "duplicated Declaration of variables with same name"
                        SemanticError(t, "Duplicated Declaration of Variables with the Same Name; It will not be added into Symbol
                            Table!");
                    }
```

처음엔 Void 형의 변수 혹은 배열 변수 선언을 심볼테이블을 만드는 과정에서 확인하여 즉시 에러메세지를 띄우고 심볼테이블에 추가하지 않으려 했지만 이렇게 하면 Type체크 과정에서 세그멘테이션 폴트가 자주 일어나게 될 수 있어서 일단 Void 형의 변수를 선언할 수 있게 해주었고, 이후에 Type Checking을 하면서 이를 처리하는 방식으로 구현하였다. 변수 선언은 현재 Scope내에서만 같은 이름의 변수가 선언 되어 있는지만 확인하면 된다.

```
228                 case FuncK:
229                     if(st_lookup_excluding_parent(CurScopeIndex, t->attr.name)==1){
230                         //Function with the same name already exists in Declaring scope
231                         //error "duplicated Declaration of function with same name"
232                         SemanticError(t, "Duplicated Declaration of Function with the Same Name; It will not be added into Symbol
                                Table!");
233                     }
234                     else{ //can declare;
235                         //insert into symtab;
236                         st_insert(CurScopeIndex, t->attr.name, t->lineno, location++, t->type, t);
237                         //add new scope
238                         TotalScopes++;
239                         CurScopeIndex = TotalScopes;
240                         ScopeLevel++;
241                         ScopeTable[CurScopeIndex] = (ScopeList)malloc(sizeof(struct ScopeListRec));
242                         ScopeTable[CurScopeIndex]->name = t->attr.name;
243                         ScopeTable[CurScopeIndex]->index = CurScopeIndex;
244                         ScopeTable[CurScopeIndex]->nodeForFunc = t;
245                         ScopeTable[CurScopeIndex]->nestedLevel = ScopeLevel;
246                         ScopeTable[CurScopeIndex]->Bucket = NULL;
247                         ScopeTable[CurScopeIndex]->NofBuckets = 0;
248                         ScopeTable[CurScopeIndex]->parent = CurrentScope;
249                         CurrentScope = ScopeTable[CurScopeIndex];
250                         //flague on;
251                         isInFuncKpre = 1;
252                         funcCompound = 0;
253                     }
254                     break;
```

함수가 선언될 때는 여러 중요한 정보를 새롭게 생성되는 Scope에 저장해주고, 현재의 Scope을 새로 생성한 Scope로 바꿔준다.

```
256                 case ParamK:
257                     if(t->type == Void && t->attr.name == NULL){
258                         // do nothing (void) OK!
259                     }
260                     /*else if (t->type == Void && t->attr.name != NULL){
261                         // void type parameter error
262                         SemanticError(t, "Parameter's type should not be [Void]");
263                     }*/
264                     else if ( st_lookup_excluding_parent(CurScopeIndex, t->attr.name) == 1 ){
265                         //error, duplicated parameter name;
266                         SemanticError(t, "Duplicated Parameter with the Same Name; It will not be added into Symbol Table, so this
                                Parameter will be disregarded");
267                     }
268                     else if ( st_lookup_excluding_parent(CurScopeIndex, t->attr.name) == -1 ){
269                         //insert into symtab;
270                         st_insert(CurScopeIndex, t->attr.name, t->lineno, location++, t->type, t);
271                     }
272                     break;
273
274                 case ArrParamK:
275                     /*if(t->type == Void){
276                         //error void[] type parameter;
277                         SemanticError(t, "Parameter's type should not be [Void]");
278                     }*/
279                     if (st_lookup_excluding_parent(CurScopeIndex, t->attr.name)==1){
280                          //error, duplicated parameter name;
281                         SemanticError(t, "Duplicated Parameter with the Same Name; It will not be added into Symbol Table, so this
                                Parameter will be disregarded");
282                     }
283                     else if(st_lookup_excluding_parent(CurScopeIndex, t->attr.name)!=1){
284                         //insert into symtab
285                         st_insert(CurScopeIndex, t->attr.name, t->lineno, location++, t->type, t);
286                     }
287                     break;
```

함수의 인자 역시 처음엔 Void 형의 변수 혹은 배열 변수 선언을 심볼테이블을 만드는 과정에서 확인하여 즉시 에러메세지를 띄우고 심볼테이블에 추가하지 않으려 했지만 그렇게 하면 함수의 인자의 갯수가 선언과는 다르게 심볼테이블에 저장되어서 함수가 호출될 때 문제가 생길 수 있어 일단 허용했다.

```
289                 case IdK:
290                     if( st_lookup(CurScopeIndex, t->attr.name) == -1){
291                         //Does not exists in Symtab;
292                         //error Semantic Error: undeclared variable;
293                         SemanticError(t, "Using Undeclared Variable that does not exist in Symbol Table");
294                     } else{
295                         //insert line no.
296                         st_insert(CurScopeIndex, t->attr.name, t->lineno, 0, t->type, t);
297                     }
298                     break;
```

변수의 참조를 하는 부분에선 변수가 이미 선언 되어있는 변수인지를 확인해주었다.

-checkNode()

```
static void checkNode(TreeNode * t){
    switch (t->nodekind){
        case ExpK:
            switch(t->kind.exp){

                case VarK:
                    if(t->type == Void)
                        typeError(t,"Declaration of [Void] type Variable is Invaild");
                    break;

                case ArrVarK:
                    if(t->type == VoidArray)
                         typeError(t,"Declaration of [VoidArray] type Variable is Invaild");
                    break;
```

Void 타입으로 선언된 변수는 에러메세지를 띄운다.

```
                case FuncK:
                    if(isWellReturned == 1 && ReturnTypeErr != 1){
                        isWellReturned = 0;
                        ReturnTypeErr = 0;
                        isInFuncKpre = funcCompound = 0;
                        CurrentScope = CurrentScope->parent;
                        ScopeLevel--;
                    }
                    else{
                        isWellReturned = 0;
                        ReturnTypeErr = 0;
                        ScopeLevel--;
                        CurrentScope = CurrentScope->parent;

                        typeError(t, "Return-Statement is [Missing] or [NOT properly stated] in this function");
                    }
                    //fprintf(listing, "scope up form function scope\n");
                    break;
```

함수 부분에서는 함수 내부에서 return 과정이 함수의 return type에 알맞게 잘 일어 났는지를 확인하여 에러메세지를 출력하도록 해주었다.

```
                case ParamK:
                    if(t->type == Void){
                        if(t->sibling != NULL){
                            typeError(t,"[Void] Type Parameter is Invalid");
                        }
                    }
                    break;

                case ArrParamK:
                    if(t->type == VoidArray)
                        typeError(t,"[VoidArray] Type Parameter is Invalid!");
                    break;
```

함수선언에서 void 타입의 변수가 인자로 들어가 있는 경우도 모두 에러 메세지를 띄운다.

```
                case RelOpK:
                    if(t->child[0]->type==Integer && t->child[1]->type==Integer){
                        t->type = Boolean;
                    }
                    else{
                        t->type = TypeError;
                        typeError(t,"Operand of RelOp should be [Interger] type");
                    }
                    break;

                case OpK:
                    if(t->child[0]->type==Integer && t->child[1]->type==Integer){
                        t->type = Integer;
                    }
                    else{
                        t->type = TypeError;
                        typeError(t,"Operand of Op should be [Interger] type");
                    }
                    break;
```

일반 사칙연산 Operation의 Operand들은 모두 int 타입이 아닌 경우 에러 메세지를 띄우고 OpK 노드의 타입을 TypeError로 바꿔주어 이후에 이 연산 결과를 사용하게 될 시에도 에러가 출력되게끔 해주었다. 만약 두 Operand가 모두 int type일 경우 OpK의 타입을 int 로 해주었다. RelOpK 도 OpK 와 비슷하게 두 Operand의 타입이 모두 int 가 아닌경우 TypeError 타입으로 바꿔주었고, 에러 메세지를 출력했다. 만약 두 Operand가 모두 int type일 경우 RelopK의 타입을 Boolean으로

해주었다.

```
539                 case ConstK:
540                     t->type = Integer;
541                     break;
542 
543                 case IdK://lookup sympad;
544                 {
545                     ScopeList S = CurrentScope;
546                     BucketList l;
547                     int wasFunc = 0;
548                     int i=0;
549                     while(S!=NULL){
550                         l=S->Bucket;
551                         while(i < S->NofBuckets  &&  l != NULL){
552                             if(l!=NULL && strcmp(t->attr.name,l->name)!=0){
553                                 l=l->next;
554                             }
555                             else if(l!=NULL && strcmp(t->attr.name, l->name) == 0){
556                                 if(l->node->nodekind == ExpK && l->node->kind.exp == FuncK){//funciton name을 찾은경우
557                                     // search more
558                                     wasFunc = 1;
559                                 }
560                                 else if(l->type == IntegerArray && t->child[0]==NULL){
561                                     t->type = IntegerArray;
562                                     return;
563                                 }
564                                 else if(l->type == IntegerArray && t->child[0]!=NULL){
565                                     t->type = Integer;
566                                     return;
567                                 }
568                                 else if(l->type == Integer && t->child[0]!=NULL){
569                                     t->type = TypeError;
570                                     typeError(t, "Using [Integer] Type Variable as [IntegerArray]; Accessing by index is Invalid");
571                                     return;
572                                 }
573                                 else{
574                                     t->type = l->type;
575                                     return;
576                                 }
577                             }
578                             i++;
579                         }
580                         S = S->parent;
581                         i=0;
582                     }
583                     if(wasFunc){
584                         t->type = TypeError;
585                         typeError(t, "Using Function name as a variable is Invalid!!");
586                         return;
587                     }
588                     t->type = TypeError;
589                     typeError(t, "Using Undeclared Variable");
590                 }
591                     break;
```

상수의 경우 타입을 int 로 해주었고, 일반적인 변수의 사용일 경우 가장먼저 이 변수가 미리 선언이 되어있어서 심볼테이블에 존재하는지를 확인해주었고, 선언 되었을 때의 타입과 비교를 하여 타입을 지정해 주었다. 또한 Array 변수인 경우 노드의 child의 존재 유무에 따라 int 혹은 int array 형태로 저장하였다. 또한 함수의 이름을 변수처럼 호출하는 경우도 예외처리하여 에러를 출력하게 했다.

```
596                 case AssignK://
597                     if (t->child[0]->type != Integer || t->child[1]->type != Integer)
598                         typeError(t->child[0],"Type Conflict in Assignment; LHS's type and RHS's type should be all Integer");
599                     break;
```

Assign의 경우 OpK 나 RelOpK 와 마찬가지로 Operand들의 타입이 모두 같은지를 확인해주었다.

```
601                 case CompndK:
602                     if(just_compound > 0){
603                         ScopeLevel--;
604                         CurrentScope = CurrentScope->parent;
605                         //fprintf(listing, "scope up from juct copmstmt\n");
606                         just_compound--;
607                     }
608                     else if(isInFuncKpre==1 && funcCompound==1){
609                         funcCompound=0;
610                     }
611                     break;
```

함수가 아닌 compound statement일 때는 현재의 Scope을 현재 Scope의 parent로 변경해 주었다.

```
613                 case SelectK://
614                     if(t->child[0]->type != Boolean)
615                         typeError(t->child[0],"Invalid Expression, Conditional of If() should be [Boolean] Type");
616                     break;
617 
618                 case IterK://
619                     if(t->child[0]->type != Boolean)
620                         typeError(t->child[0],"Invalid Expression, Conditional of While() should be [Boolean] Type");
621                     break;
```

If 문이나 while문일 때, conditional 부분의 타입이 Boolean 인지를 확인한다.

```
623                 case ReturnK://함수의 리턴타입이랑 맞는지 확인해야함.
624                     if(CurrentScope->nodeForFunc->type == Void){ // should be return;
625                         if(t->child[0] != NULL){
626                             typeError(t, "Invalid Return Type!, Return Type should be [Void]");
627                             ReturnTypeErr = 1;
628                         }
629                         else{
630                             isWellReturned = 1;
631                         }
632                     }
633                     else if(CurrentScope->nodeForFunc->type == Integer){ //should return somthing;
634                         if(t->child[0]==NULL){
635                             typeError(t, "Invalid Return Type!, Return Type should be [Interger]");
636                             ReturnTypeErr = 1;
637                         }
638                         else if(t->child[0]->type != Integer){
639                             typeError(t, "Invalid Return Type!, Return Type should be [Interger]");
640                             ReturnTypeErr = 1;
641                         }
642                         else{
643                             isWellReturned = 1;
644                         }
645                     }
646                     break;
```

Return 구문에서는 현재 스코프에서의 올바른 return type이 어떤 타입인지를 확인하여 알맞게 Return이 되었는지를 확인해주었다. 또한 함수 Scope내에서 잘못된 형식의 Return 이 하나라도 존재하게 되면 함수 선언 부분에서 에러를 출력하게 해주었다. 또한 int 형 함수 내에서 return 구문이 없이 함수 선언이 종료되는 경우도 예외처리를 해주어 에러를 출력하게끔 해주었다.

```
648                 case CallK://
649                     if(t->child[0] == NULL){//void parameter//
650                         BucketList l = ScopeTable[0]->Bucket;
651                         while(l != NULL && strcmp(l->name,t->attr.name) != 0){
652                             l = l->next;
653                         }
654                         if(l!=NULL){//found!
655                             t->type = l->type;
656                             if(l->memloc == -1 ){//case of input fuc
657                                 //No error
658                                 return;
659                             }
660                             else if(l->memloc == -2){//case of calling output without parameter Error
661                                 typeError(t, "Need Proper Parameter for Function Call");
662                             }
663                             else if(l->node->child[0]->type == Void && l->node->child[0]->sibling == NULL){
664                                 //good NO ERror
665                                 return;
666                             }
667                             else{
668                                 typeError(t, "Need Proper Parameter for Function Call");
669                             }
670                         }
671                         else{
672                             t->type = TypeError;
673                             typeError(t, "Calling Undeclared Function");
674                         }
675                     }
```

함수를 호출하는 부분이 가장 복잡한 구조로 되어있다. 위 사진은 Parameter 없이 함수를 호출하는 경우이다[ex: "function()"]. 가장먼저 "input()"함수와 "output()"함수의 예외처리를 해주었고, 선언이 되어있는 함수인지를 확인하여 선언된 함수의 인자가 void로 선언되어 있는지를 확인하여 에러를 검사한다.

```
676                     else{//parameter list 존재하는 경우//
677                         TreeNode * tempnode;
678                         TreeNode * params;
679                         BucketList l = ScopeTable[0]->Bucket;
680                         while(l != NULL && strcmp(l->name,t->attr.name) != 0){
681                             l = l->next;
682                         }
683                         if(l!=NULL){//found!
684                             t->type = l->type;
685                             if(l->memloc == -1 ){//case of calling input with parameter
686                                 //error
687                                 typeError(t, "Need Proper Parameter for Function Call");
688                             }
689                             else if(l->memloc == -2){//case of calling output without parameter Error
690                                 //No error
691                                 if(t->child[0]->type == Integer && t->child[0]->sibling == NULL){
692                                     //no Error
693                                     return;
694                                 }
695                                 else{
696                                     typeError(t, "Need Proper Parameter for Function Call");
697                                 }
698                             }
699                             else if(l->node->child[0]->type==Void && l->node->child[0]->sibling==NULL){
700                                 typeError(t, "Need Proper Parameter for Function Call");
701                             }
702                             else{
703                                 tempnode = l->node->child[0];//decl param
704                                 params = t->child[0];//call param
705                                 while(tempnode!=NULL && params!=NULL){
706                                     if(tempnode->type == params->type){
707                                         tempnode = tempnode->sibling;
708                                         params = params->sibling;
709                                     }
710                                     else{
711                                         typeError(t, "Need Proper Parameter for Function Call");
712                                         return;
713                                     }
714                                 }
715                                 if(tempnode==NULL && params==NULL){
716                                     return;
717                                 }
718                                 else{
719                                     typeError(t, "Need Proper Parameter for Function Call");
720                                 }
721                             }
722                         }
723                         else {
724                             t->type = TypeError;
725                             typeError(t, "Calling Undeclared Function");
726                         }
727                     }
728 
729                     break;
```

위사진은 함수 호출에서 인자가 있는 호출은 하는 경우이다[ex: function(a,b,c)]. 이러한 경우 함수가 선언이 되어있는지 확인을 한 후에 선언된 함수의 이자 부분은 현재의 호출할 때 사용된 인자들과 1대1로 타입을 비교하여 인자의 갯수가 다르거나, 인자의 타입이 다른 경우를 검사하여 에러메세지를 출력한다.

Results & Discuss:

****** 과제 진행 환경 *******

-코드 작성 및 변경: **MAC OS Mojave 10.14.6 ; IDE: Xcode**

-테스트 및 실행결과 캡쳐(REPORT 첨부 사진): **MAC OS TERMINAL** (**Makefile** 수정 후 실행)

-테스트[2] (Ubuntu호환검사): **Ubuntu 16.04 LTS VirtualBox in MacBook Pro Retina 2015**

-git push 환경: **Ubuntu 16.04 LTS VirtualBox in MacBook Pro Retina 2015**

****** 실행 결과 출력 양식 *******

<1. 심볼 테이블 생성 과정>

- **같은 Scope 내에 중복된 이름의 함수, 변수, 함수선언시 사용되는 인자가 선언되는 경우** 이를 확인하여 **에러메세지를 출력**하고 **중복된 이름의 변수, 함수, 인자 들은 심볼테이블에 추가되지 않는다.**

<2.심볼 테이블 출력 과정>

- 생성된 심볼테이블을 출력한다.

<3.Type-Checking 과정>

- 생성된 심볼테이블을 바탕으로 Type Checking을 진행하여 모든 Type Error를 출력한다.
- **Void 타입의 변수, Assign 에서의 타입 체크, 연산과정에서 피연산자들의 타입 체크, 리턴 타입 체크와 함수 전체에서 return 과정이 타입에 맞게 올바르게 이뤄졌는지 확인, 함수 호출시 인자들의 갯수와 타입을 체크, 변수 사용시 타입과 미리 선언 되어 있는지를 체크, if나 while 문에서 conditional 부분의 타입이 Boolean인지를 체크, ... (이외에도 더 많은 타입 에러 종류를 처리)**

****위와같은 과정 순서로 인해 에러메세지가 심볼테이블을 중간에 두고 위 아래 모두 출력될 수 있다.***

****** TEST CASE *******

<1. test.cm> - 과제 명세에 첨부됨

<2. sort.cm> - 포탈에 첨부됨

<3. p3test01.cm> - 과제 명세에 첨부됨

<4. p3test02.cm > - 과제 명세에 첨부됨

<5. p3test03.cm > - 과제 명세에 첨부됨

<6. p3test04.cm > - 과제 명세에 첨부됨

(1). test.cm

test.cm

```
/*A program to perform Euclid's
  Algorithm to compute 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));
}
```

```
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$ ./cminus test.cm

C-MINUS COMPILATION: test.cm

Building Symbol Table...

>> Input() function & Output() function are added into symbol table <<


Building Symbol Table is done!!

[1].<Symbol table>

Scope Name & Number   Symbol Name   Location   Line Numbers
--------------------  ------------  --------   ----------------------
    global      0           main        4       11
    global      0            gcd        1        4   7  15
    global      0         output       -2       -1  15
    global      0          input       -1       -1  14  14
       gcd      1              v        3        4   6   7   7   7
       gcd      1              u        2        4   6   7   7
      main      2              y        6       13  14  15
      main      2              x        5       13  14  15


 [2].<Scope Table>

Scope Name & Number  Parant name & No.  nested Level
-------------------  -----------------  ------------
    global      0          (NULL)            0
       gcd      1       global      0        1
      main      2       global      0        1


 [3].<Function and Global variables>

    ID name     ID type      DATAType
------------   ---------    ------------
      main      Function       Void
       gcd      Function     Integer
    output      Function       Void
     input      Function     Integer


 [4].<Function Parameter and local variables>

 Scope Name&no.     nestedLevel    ID name        DataType
-----------------   -----------    -----------    ---------
     gcd      1            1              v       Integer
     gcd      1            1              u       Integer

    main      2            1              y       Integer
    main      2            1              x       Integer


Checking Types...

Type Checking Finished
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$
```

의도한대로 심볼 테이블이 잘 만들어졌다.

(2). sort.cm

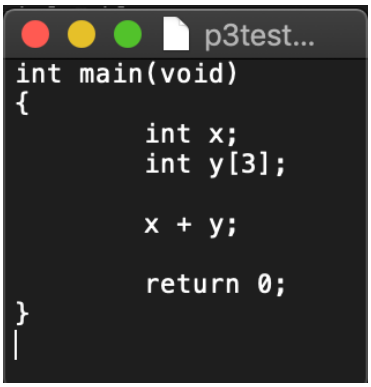
sort.cm

```
/* 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;
        }
}
```

```
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$ ./cminus sort.cm

C-MINUS COMPILATION: sort.cm

Building Symbol Table...

>> Input() function & Output() function are added into symbol table <<


Building Symbol Table is done!!


 [1].<Symbol table>

Scope Name & Number   Symbol Name   Location   Line Numbers
--------------------  ------------  --------   ------------------------
    global      0           main       16       35
    global      0           sort        9       21  42
    global      0         minloc        2        4  27
    global      0              x        1        3  39  42  45
    global      0         output       -2       -1  45
    global      0          input       -1       -1  39
    minloc      1              k        8        7   8  14  18
    minloc      1              x        7        6   9  12  13
    minloc      1              i        6        5  10  11  12  13  14  16  16
    minloc      1           high        5        4  11
    minloc      1            low        4        4   8   9  10
    minloc      1              a        3        4   9  12  13
      sort      4              k       14       23  27  28  29
      sort      4              i       13       22  24  25  27  29  30  31  31
      sort      4           high       12       21  25  27
      sort      4            low       11       21  24
      sort      4              a       10       21  27  28  29  29  30
    (null)      5              t       15       26  28  30
      main      6              i       17       36  37  38  39  40  40  43  44  45  46  46


 [2].<Scope Table>

Scope Name & Number  Parant name & No.  nested Level
--------------------  ------------------  ------------
    global      0          (NULL)           0
    minloc      1       global      0       1
    (null)      2       minloc      1       2
    (null)      3       (null)      2       3
      sort      4       global      0       1
    (null)      5         sort      4       2
      main      6       global      0       1
    (null)      7         main      6       2
    (null)      8         main      6       2

 [3].<Function and Global variables>

    ID name     ID type      DATAType
------------  ---------    ------------
      main      Function        Void
      sort      Function        Void
    minloc      Function     Integer
         x      Variable    IntegerArray
    output      Function        Void
     input      Function     Integer


 [4].<Function Parameter and local variables>

 Scope Name&no.     nestedLevel    ID name        DataType
-----------------  ------------    -----------    ---------
  minloc      1           1              k       Integer
  minloc      1           1              x       Integer
  minloc      1           1              i       Integer
  minloc      1           1           high       Integer
  minloc      1           1            low       Integer
  minloc      1           1              a      IntegerArray


    sort      4           1              k       Integer
    sort      4           1              i       Integer
    sort      4           1           high       Integer
    sort      4           1            low       Integer
    sort      4           1              a      IntegerArray

  (null)      5           2              t       Integer

    main      6           1              i       Integer



Checking Types...

Type Checking Finished
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$
```

의도한 대로 심볼테이블이 잘 만들어 졌다.

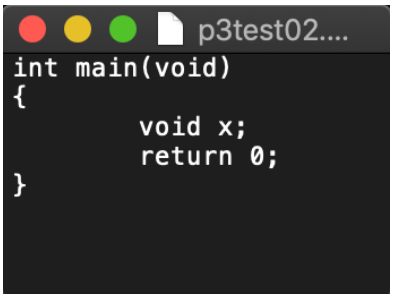
(3). p3test01.cm




```
[yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$ ./cminus p3test01.cm

C-MINUS COMPILATION: p3test01.cm

uilding Symbol Table...

>> Input() function & Output() function are added into symbol table <<


Building Symbol Table is done!!


 [1].<Symbol table>

Scope Name & Number   Symbol Name   Location   Line Numbers
-------------------  -------------  --------   ------------------------
    global      0           main        1        1
    global      0         output       -2       -1
    global      0          input       -1       -1
      main      1              y        3        4   6
      main      1              x        2        3   6


 [2].<Scope Table>

Scope Name & Number  Parant name & No.  nested Level
-------------------  -----------------  ------------
    global      0          (NULL)            0
      main      1       global      0        1


 [3].<Function and Global variables>

    ID name     ID type      DATAType
------------   ---------    ------------
      main      Function     Integer
    output      Function       Void
     input      Function     Integer


 [4].<Function Parameter and local variables>

 Scope Name&no.     nestedLevel    ID name        DataType
-----------------   -----------    -----------    ---------
    main      1           1              y      IntegerArray
    main      1           1              x       Integer


Checking Types...
Error(while type-checking) at line 6: Operand of Op should be [Interger] type

Type Checking Finished
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$
```

의도한 대로 심볼 테이블이 잘 만들어졌고, 6번째 줄에서 타입 에러가 잘 출력된다.

(4). p3test02.cm




```
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$ ./cminus p3test02.cm

C-MINUS COMPILATION: p3test02.cm

Building Symbol Table...

>> Input() function & Output() function are added into symbol table <<


Building Symbol Table is done!!


 [1].<Symbol table>

Scope Name & Number   Symbol Name   Location   Line Numbers
--------------------  ------------  --------   ----------------------
    global      0           main        1        1
    global      0         output       -2       -1
    global      0          input       -1       -1
      main      1              x        2        3


 [2].<Scope Table>

Scope Name & Number  Parant name & No.  nested Level
-------------------  -----------------  ------------
    global      0          (NULL)            0
      main      1       global      0        1


 [3].<Function and Global variables>

    ID name      ID type       DATAType
------------   ---------    ------------
      main      Function     Integer
    output      Function       Void
     input      Function     Integer


 [4].<Function Parameter and local variables>

 Scope Name&no.      nestedLevel    ID name        DataType
-----------------   -----------    -----------    ---------
    main      1            1               x         Void


Checking Types...
Error(while type-checking) at line 3: Declaration of [Void] type Variable is Inv

Type Checking Finished
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$
```

의도한 대로 심볼 테이블이 잘 만들어졌고, 3번째 줄에서 void 형 변수 선언에 대한 에러가 잘 출력된다.

(5). p3test03.cm

p3test03.cm

```
int x(int y)
{
        return y + 1;
}

int main(void)
{
        int a;
        int b;
        int c;

        return x(a,b,c);
}
```

```
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$ ./cminus p3test03.cm

C-MINUS COMPILATION: p3test03.cm

Building Symbol Table...

>> Input() function & Output() function are added into symbol table <<


Building Symbol Table is done!!


 [1].<Symbol table>

Scope Name & Number   Symbol Name   Location   Line Numbers
--------------------  ------------  --------   ----------------------
    global      0           main        3        6
    global      0              x        1        1  12
    global      0         output       -2       -1
    global      0          input       -1       -1
         x      1              y        2        1   3
      main      2              c        6       10  12
      main      2              b        5        9  12
      main      2              a        4        8  12


 [2].<Scope Table>

Scope Name & Number  Parant name & No.  nested Level
-------------------  -----------------  ------------
    global      0          (NULL)            0
         x      1       global      0        1
      main      2       global      0        1


 [3].<Function and Global variables>

    ID name     ID type      DATAType
------------   ---------    ------------
      main      Function     Integer
         x      Function     Integer
    output      Function       Void
     input      Function     Integer


 [4].<Function Parameter and local variables>

 Scope Name&no.     nestedLevel    ID name        DataType
-----------------   -----------    -----------    ---------
       x      1           1              y       Integer

    main      2           1              c       Integer
    main      2           1              b       Integer
    main      2           1              a       Integer


Checking Types...
Error(while type-checking) at line 12: Need Proper Parameter for Function Call

Type Checking Finished
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$
```

의도한 대로 심볼 테이블이 잘 만들어졌고, 12번째 줄에서 올바르지 않은 인자로 함수를 호출하는 것에 대한 에러가 잘 출력된다.

(6). P3test04.cm

p3test04.cm

```
int main(void)
{
        return x;
}
```

```
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$ ./cminus p3test04.cm

C-MINUS COMPILATION: p3test04.cm

Building Symbol Table...

>> Input() function & Output() function are added into symbol table <<

Error(while building SymTab) at line 3: Using Undeclared Variable that does not exist in Symbol Table

Building Symbol Table is done!!


 [1].<Symbol table>

Scope Name & Number   Symbol Name   Location   Line Numbers
-------------------  -------------  --------   ------------------------
    global      0           main        1        1
    global      0         output       -2       -1
    global      0          input       -1       -1


 [2].<Scope Table>

Scope Name & Number  Parant name & No.  nested Level
-------------------  -----------------  ------------
    global      0          (NULL)            0
      main      1       global      0        1


 [3].<Function and Global variables>

    ID name     ID type      DATAType
------------   ---------    ------------
      main      Function     Integer
    output      Function       Void
     input      Function     Integer


 [4].<Function Parameter and local variables>

 Scope Name&no.     nestedLevel    ID name        DataType
-----------------   -----------    -----------    ---------


Checking Types...
Error(while type-checking) at line 3: Using Undeclared Variable
Error(while type-checking) at line 3: Invalid Return Type!, Return Type should be [Interger]
Error(while type-checking) at line 1: Return-Statement is [Missing] or [NOT properly stated] in this function

Type Checking Finished
yangsangheonui-MacBook-Pro:P3_prac12 SangheonY$
```

의도한 대로 심볼 테이블이 잘 만들어졌고, 심볼테이블을 만드는 과정에서 선언되지 않는 변수의 사용 에러를 감지한다. 또한 심볼 테이블 출력 후 타입체킹에서 3번째 줄에서 선언되지 않은 변수 사용에 대한 에러를 감지하고 이로 인해 x 노드의 타입은 "typeError"가 되고, main 함수의 return type과 다른 타입의 변수를 return 하기에 return type 에러가 감지되고, 함수 내에서 하나라도 return type 에러가 감지되면, 함수 선언이 시작되는 부분에서 return type 에러를 한번 더 감지하도록 의도한부분이 잘 출력된다.