Compiler Project3 Report

1. Compilation method and environment

   Ubuntu 환경에서 실행한다.

   주어진 makefile로 make 명령어를 실행하면 cminus_semantic 실행파일이 생성되는데, ./cminus_semantic [semantic analysis를 수행할 파일 이름] 을 실행하면 실행 결과가 출력된다.

2. Semantic analysis implementation process

   - 명세의 Case1 의 형태로 심볼테이블을 작성하였다.

```
- typedef struct _FuncInfo
- {
-     ExpType returnType;
-     int paramNum;
-     ExpType paramTypes[MAX_PARAM_NUM];
- } FuncInfo;
-
- typedef struct BucketListRec
-    { char * name;
-      char * scope;
-      ExpType type;
-      LineList lines;
-      FuncInfo funcInfo;
-      int memloc ; /* memory location for variable */
-      struct BucketListRec * next;
-    } * BucketList;
-
- typedef struct ScopeListRec
-     {
-       char * name;
-       struct ScopeListRec * parent;
-       struct ScopeListRec * next;
-       BucketList hashTable[SIZE];
-       int location;
-     } * ScopeList;
```

- ScopeList 구조체를 만들어 기존 BucketList의 hashTable 과 같은 형태로 해시테이블을 만들어 놓고, ScopeList 내에 BucketList를 담는 해시테이블을 넣었다. 그리고 해당 테이블 내에는 그 Scope 내의 symbol들이 들어간다.

```
ScopeList ScopeStack[SIZE];
int StackTop = -1;
```

- 새로운 함수나 Compound statement가 생길 때마다 ScopeStack에 해당 Scope를 푸시하고 끝날 때마다 Pop하여 stack의 top을 통해 현재 어떤 Scope에 있는지 구분할 수 있다.

- St_insert에서는 scope 와 type을 추가적으로 매개변수로 받아, 해당 scope에 symbol을 추가한다.

- St_lookup 에서도 scope 매개변수를 추가적으로 받아서 해당 Scope부터 parent로 쭉 따라올라가며 그 안에 찾는 symbol의 name이 존재한다면 해당 버킷을 리턴하고, 없다면 NULL을 리턴한다.

- St_lookup_excluding_parent 에서는 매개변수로 받은 해당 scope에서만 심볼을 찾는다.

- buildSymTab에서 traverse를 하기 전, globalScope를 생성하고 스택에 푸시하고, traverse가 끝나면 globalScope를 pop한다.

```
globalScope = createScopeList("global");
push(globalScope);
...

pop(globalScope);
```

- buildSymTab에서 input과 output 함수를 global Scope에 넣는다.

InsertNode

- insertNode에서 트리노드가 함수선언이거나 Compound stmt 타입인 경우에 새로운 Scope를 만들어 스택에 푸시하고, traverse의 postProc 함수에 popNode 함수를 넣어 해당 Scope가 끝나면 pop한다.

- InsertNode에서 변수선언, 함수선언, 변수사용, 함수호출에서 st_lookup을 통해 심볼을 찾고, 이미 있는 경우, undefined/redefined variable/function 오류를 캐치한다.

- 함수의 인자개수, 인자타입, 리턴타입을 저장하는 FuncInfo 구조체를 BucketList에 넣어 함수 선언 시 심볼을 생성할 때 해당 정보들을 같이 저장해둔다.

CheckNode

- ReturnK에서 현재 함수의 리턴타입과 트리노드의 리턴타입이 맞지 않으면 에러를 캐치한다.
- BinaryK에서 source나 destination의 타입이 Integer가 아니라면 에러를 캐치한다.
- VarK에서 인덱스를 접근하는 경우 인덱스 내부 접근자가 Integer가 아닌 경우 에러를 캐치한다.
- IfK에서 조건부가 Integer가 아닌 경우 에러를 캐치한다.
- CallK에서 인자 개수나 인자 타입이 맞지 않는 경우 에러를 캐치한다.

3. 테스트코드 (소스코드, 실행 결과)

```
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ ./cminus_semantic test.txt

C-MINUS COMPILATION: test.txt
Error: Array Indexing Error at line 3
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ ./cminus_semantic test.3.txt

C-MINUS COMPILATION: test.3.txt
Error: Undefined Variable 'a' at line 2
Error: Undefined Variable 'a' at line 2
Error: Undefined Variable 'a' at line 2
Error: Undefined Variable 'a' at line 2
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ ./cminus_semantic test4.txt

C-MINUS COMPILATION: test4.txt
Error: Type error at line 2: if/while condition is not Integer
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ ./cminus_semantic test5.txt

C-MINUS COMPILATION: test5.txt
Error: Type error at line 7: Number of parameter mismatch
```

```
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ cat test.txt
int main(void) {
    int x[5];
    x[output(5)] = 3 + 5;

    return 0;
}
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ cat test.3.txt
void main(void) {
    if (a < 0) if (a > 3) a = 3; else a = 4;
}
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ cat test4.txt
int main(void) {
    if (output(5)) {}
    return 0;
}
dayun@dayun-VirtualBox:~/2021_ele4029_2019056799/3_Semantic$ cat test5.txt
int x(int y){
    return y + 1;
}

int main(void) {
    int a; int b; int c;
    return x(a, b, c);
}
```