南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMPILIER @Liu Yepang 2019

for SUSTech CSE

Project 2

Edited by

汪至圆

11610634

2019

SHENZHEN

1 Lab Demand

In last project, I have finished the parser for the SPL source code and generated the syntex tree. In this project, I will do the semantic analysis for the SPL source code based on the syntex generated by the last project.

2 Environment

The environment for my project is g++ 9.1, flex 2.6, bison 3.4. All the work of coding and test was finished in the Manjaro 18.1 which based on Linux 4.19.

3 Run

My project was compiled by the make. If you want to compile the project, you just need run 'make splc'. And then it will output a executable file splc.out. The splc.out will receive a argument which is the path of SPL source code file, and the output of the program will at the same folder of the source code and the extend name of the file will be ".out". The context of the output file will be the error in the source code file.

4 Data Structures To Store Message

To store the message of the variable, function, scope and the structure, I design some data structure.

- Variable Information:

```
struct VarInfo {
    string name; //Variable name
    int type; //Variable type
    bool array; //If the Variable is array
    int dimension; //The dimension of the Variable
};
```

- Function Information:

```
struct FuncInfo {
    string name; //Function name
    int returnType; //Return Type of the Function
    vector<VarInfo> argSet; //The arguments of the Function
};
```

- Structure Information:

```
struct StructInfo {
    string name; //The name of the Structure
    vector<VarInfo> varSet; //The fields of the Structure
};
```

- Scopes Information:

```
struct ScopeInfo {
    map<string, VarInfo> varSet; //The variables in current scope.
    ScopeInfo *parentScope = nullptr; // The parent Scope of current scope.
    vector<ScopeInfo> childList; //The children Scopes of current scope.
};
```

5 Definition Process

All the variable, function, structure must be defined before use it. When define the variable, I will check if the type of it exists, and whether the variable name exists in the current scope. When define the function, I will check the return type and the arguments type exists or not. And the function name exists or not. The SPL not support overload the function now. When define the structure, I will check the structure name haven't been defined, and the types of the fields is existed.

6 Executing Process

After definition, there are also some executing codes. For these codes, I will check the types of the expressions next the operators, the return type of the function, the existence of the variables, functions and structures.

7 Optional Rules

For the two optional rules, I implement all of them.

For the first optional rules, I define a global scope tree, the root of the tree is at the root of the syntex tree, while define a new function or create a new If/While statement, I will create a new scope as a child scope of current scope. While define the variable, I will only check the current scope, but when finding a variable which is defined, I will find it in the scope tree recursively until find it or arrive the root scope.

For the second optional rules, I define the fields of the structure in a array, and when check the type of two variables, if they are strcuture, I will check the type of fields of them.

8 bonus

8.1 Add a new type bool

Revoke Assumption 2, I add a new type bool, and set only bool type value can be used as the condition of the If/While statement.

The test file is test/test_2_b01.spl

8.2 Orderd structural equivalence

For the two strcuture, I will check their fields in order. For example:

```
struct a{
    int c;
    float d
}
strcut b{
    float d;
    int c
}
```

a and b are different.

The test file is test/test_2_b02.spl

8.3 Recursive structural equivalence

For two structures, if they have structure variables in their fields, I will check the fields recursively. For example:

```
struct a{
    int c;
    int d;
};
struct b{
    int c;
    int d;
};
struct a1{
    struct a aa;
    int i;
};

struct b1{
    struct b bb;
    int i;
};

struct a2{
    struct b1 bb;
    int i;
};
```

a1 and b1 are the same, a1 and a2 are different.

The test file is test/test_2_b03.spl

8.4 Assignment/Operation Between INT and FLOAT Variables

Revoke Assumption 1, when do operation between int and the float, the return value will be float. And we can assign int value to float variable, but float value can't be assigned to int variable.

The test file is test/test_2_b04.spl