编译原理第三次实验“语义分析”报告

181250086 刘佳月 181250086@smail.nju.edu.cn

实现功能

在词法分析和语法分析基础上对 C--源码进行语义分析和类型检查，完成对错误类型 1-17 和要求 2.3 的检查，而且只检查最本质的错误。

关键数据结构

符号表：使用 STL 中的 unordered_map。维护三个哈希表，map 用于存放所有变量名（变量包括基本数据类型、数组类型、结构体类型、函数形参、结构体的域）及其类型，functionMap 存放函数名及其类型，structureMap 存放结构体名及其类型。

变量类型：

```
struct Type_{
    enum{BASIC,ARRAY,STRUCTURE,FUNCTION,ERROR} kind;
    union{
        int basic;
        struct {Type elem; int size;} array;
        FieldList structure;
        FuncList myfunc;
        int error_type;
    }u;
};
```

```
struct FieldList_{
    string name;//域名
    Type type;//域的类型
    FieldList tail;//下一个域的指针
};
struct FuncList_{
    string name;//函数名&参数名
    Type type;//返回类型&参数类型
    FuncList next;
};
```

其中函数类型是一个链表，第一个节点是返回值的类型，之后的所有节点是函数参数的类型。

实现步骤

在实验二语法分析中构建好的语法分析树上，通过为每个非终结符写一个函数，使得子节点与父节点之间可以做必要的信息传递。

所有的产生式可以分为定义部分和语句部分。

在定义部分，父节点会层层向下传递 Specifier 的类型，在 VarDec 的产生式中对该变量进行定义。对于 VarDec，需要 VarDec_in_FuncParams, VarDec_in_Struct, VarDec 三个函数，分别处理函数形参定义、结构体中域的定义、全局或函数中变量的定义这三种不同情况。对于 A.1.6 中的所有 Local Definitions，需要分 DefList_in_Func 与 DefList_in_Struct 两种情况，因为函数中定义与结构体中定义时加入符号表前所做的操作不一样。

在语句部分，最重要的 Exp 函数，向上层返回表达式类型，上层负责检查 17 种错误类型。子节点也可以向上返回 ERROR 类型。

难点

1) 类型比较 isSameType 函数的实现
2) 高维数组的 Type 的实现
3) 结构体域连成链表、函数返回值+形参连成链表

印象深刻的 bug

- 实验二中构建的树节点的 str_constant 的类型是 char 数组，这次实验中符号表所有的名字都是 string。将 char[]赋值给 string（比如 FieldList->name=n->str_constant 或者 FieldList->name=string(n->str_constant)）时，以及比较 char[]与某个 string 是否相等（比如 n->child->name==”WHILE”）时，在 Debian10 上时不时出现段错误，但是在其他同学环境上不会触发错误。其实没搞明白具体的原因害，总之改了三处后就解决了，害。
  一，用 strcmp，strcmp(n->child->name,”WHILE”)==0;
  二，用 string str(char[])，先声明一个 string target(n->child->name)，然后赋值 FieldList->name=target（没有全部都改成这样，有些地方还是直接赋值）；
  三，将所有 malloc Type_, FuncList_和 FieldList_都改成 new。
- 对于产生式 StructSpecifier->STRUCT OptTag LC DefList RC，OptTag->空串，我将空串“”作为结构体名加入 structureMap，所以再次碰到这类产生式就会报结构体名重复。
- 段错误 1：处理函数和结构体的链表时，新增一个节点但忘连到原链表上
- 段错误 2：判断等号左边是否是左值时，分三种情况，由于等号左边的产生式的右部符号个数不同，写出了对 NULL 取成员变量的代码。
- int a; float b=a;这种情况也要将 b 加入符号表。