CS323 Project Phase 2 Sematic Analyze Report

姓名	学号
李昕	12012138
廖铭骞	12012919

如何运行

1. 先通过`make clean`清除上次编译的多余文件
2. 通过`make`编译运行
3. 使用`sh github_test.sh`或者`sh student_test.sh`跑老师以及学生的相应测试，测试的结果会生成在`myOutput`文件夹当中
4. 可以通过运行`sh github_judge.sh`或者`sh student_judge.sh`快速判断目标文件与生成文件之间的差别

符号表

数据结构

使用C++中`unordered_map`作为储存符号表的数据结构。`key`为符号的名字，`value`为符号的数据类型。即为`unordered_map<std::string, GlobalType*>`。`GlobalType`为自定义的数据结构，具体声明如下：

```
class GlobalType{ // 类型
public:
    NodeType nodetype;
    Array* array_ptr = nullptr;
    ParamsList* params_ptr = nullptr;
    Category category;

};
```

```
enum Category{
    PRIMITIVE, ARRAY, STRUCTURE, FUNCTION
};
```

```
class Array { // 数组类型（复合类型）
public:
    GlobalType *base;
    int size;
};
```

```
class ParamsList{ // struct 或者 class
public:
    string name;
    GlobalType *global_type = nullptr;
    ParamsList* next = nullptr;
};
```

`Category`声明包含初始类型（int, float, char），数组类型，结构类型以及函数类型。在实际使用中`GlobalType`中的`category`成员为四者之一。

`Array`是对数组类型的声明，声明数组的宽度和数组的基础类型。基础类型（`base`）为`GlobalType *`，通过层层嵌套来达到多维数组的效果。

`ParamsList`是对函数和结构体类型的阐释，表示所传入的参数或者包含的变量。通过指针`next`指向下一个参数（变量）构成链表。`name`表示该参数的名字，`global_type`表示该参数的类型。

设计思路

递归向下的解析（Recursive-Descent Parsing）

当可以没有错误地构建出一棵解析树（parsing tree）时，程序开始从根节点检查，这是 L-attributed SDD的具体实现方式。在解析树构建完成后，根据生产式逐一做判断，进行相应进一步的检查。

检查过程类似深度优先搜索，对于每一个非终结符`A`，这个解析器都会有一个函数`A`对其进行解析，这个过程使用函数`A`的参数来传递非终结符`A`的继承属性，这样做可以使得解析树的子节点可以使用非终结符`A`的属性。

当函数 `A` 完成这个过程之后，将会返回非终结符`A`的合成属性，这样位于非终结符 `A` 父节点的节点也可以使用这个属性了。

报错信息

对于报错信息，我们分为三个部分进行归纳实现，分别是定义错误（Definition Error），运算错误（Operation Error）以及参数错误（Argument Error）。

定义错误（Definition Error）

对于定义错误，我们通过在遇到一个新的标识符的时候，判断标识符是否在哈希表当中，即使用 C++ 当中 `unordered_map`的 `count()`方法进行判断，判断这个新的标识符是否已经存在于符号表当中，进而判断其是否是未定义标识符还是重复定义的标识符，这样的标识符可以是函数，也可以是变量。对应于 `Required Rules` 中的

- Type1
- Type2
- Type3
- Type4
- Type14
- Type15

类型错误（Operation Error）

对于类型错误，我们是通过比较参与运算的变量类型来判断的，变量类型是通过查看`unordered_map`的`key`对应的值来确定的，如果参与运算或者赋值的变量类型不一致，则会提示类型错误。对应于`Required Rules`中的

- Type5
- Type6
- Type7
- Type8

- Type10
- Type11
- Type12

在我们的实现当中，我们约定，如果在运算过程中右值报出了类型错误，那么将会产生一个`null`类型的结果，如果此时左值不是`null`类型，进行赋值，将会附带产生`unmatching type on both sides of assignment`的错误。

参数错误（Argument Error）

对于参数个数错误的判断，我们是通过遍历代表函数的`ParamList`的每一个参数确定的，如果在程序中传入的参数与声明时的参数个数不同，则会在遍历的过程中不同时出现`nullptr`，此时就会报出参数错误。并且在遍历的过程当中，会同步地进行参数类型的对应检查，如果发现声明的参数类型和实际传入函数的参数类型不一致，也会报出参数错误。参数错误对应于`Required Rules`中的

- Type9

以上就是我们小组在 project phase2 的报告，感谢阅读！