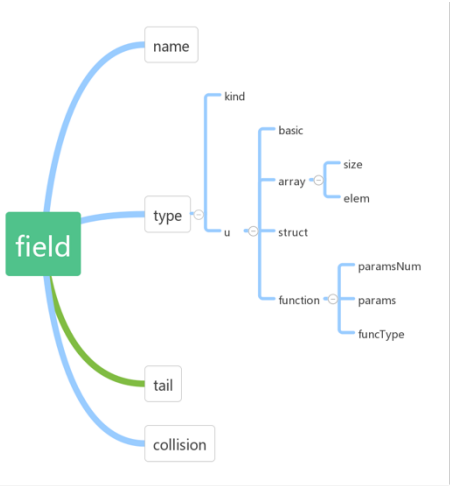
编译原理第二次实验报告

141270022 刘少聪 141270037 汪值

第二次实验任务是在词法分析和语法分析程序的基础上编写一个程序，对 C—源代码进行语义分析和类型检查，并打印分析结果。在实际的实验实现过程中，我们选择将语法分析相关代码写入单独文件中，而不是像 SDT 一样在 Bison 代码中插入语义分析的代码，这样的优点是可以使用 L 属性，而不仅仅限于 S 属性，缺点是要在语法分析树中对节点进行判断分析之后执行，代码实际运行与 Bison 中重复，而且代码显得有些复杂。

首先对语法树进行分析，在实验要求的文法中，设计语义分析的总体上有定义和调用两类，定义操作在 ExtDef 或者 Def 语法单元中出现，主要由 Specifier 和 ExtDeclist、Declist 组成，而调用操作在 Exp 语法单元中出现，所以对一个没有词法、语法错误的程序而言，语义错误只需要分析这三个语法单元即可。

语义分析的过程中最重要的部分就是符号表，实验手册上约定所有变量都是全局的，基础变量、结构体、数组之间不可同名，函数与它们可以同名。在线性链表、平衡二叉树和散列表中我们选了了散列表，规定 HASH_SIZE 为 65536，使用了手册上推荐的 hash 函数，在处理冲突上，使用了闭 hash 的办法（方便、迅速），对这个哈希表建立了初始化、插入、查找和遍历函数，其中查找函数的两个参数为变量名和是否为函数，插入函数的参数为一个不判别冲突的表节点，遍历函数的目的是方便 Debug。



类型检测是语义分析的另一个重要组成，参考实验手册上的数据结构，我们将函数也统一到类型里面进行判断，没有为函数建立单独的哈希表，而是将函数名稍加变化再哈希函数，这样结合起来有利于操作，缺点是在实际操作过程中需要对函数和其他变量进行区分，尤其是查找函数中专门加入了一个参数来判断函数。类型中的 kind 用于初步判断域的所属类型其中数组中的 elem 和函数中的 funcType 为类型，结构中的 struct 和函数中的 params 为不存在冲突的哈希表节点的结构，利用尾指针来对节点进行将几个节点进行连接，形成了一片独立的空间，实际上就是形成了一个链表的结构。

下面具体分析语法单元，在语义分析中最常使用的语法单元为定义结构 Specifier，这个结构将会对节点进行分析并生成一个类型，如果是基础类型就直接赋值生成，如果是结构类型，操作较为复杂。对于第一次定义的结构类型（STRUCT OptTag LC DefList RC），对于其中的每个子式生成全局的变量，并将这个全局变量的节点接到 struct 的节点链表上，直到所有子式都接上去，最后返回类型，如果 OptTag 不为空，那么就生成一个结构类型的变量插入表中；对于利用定义好的结构体构造类型生成结构体（STRUCT Tag），直接查表获得结构体类型，生成新的变量插入。

与类型生成相关的还有一个语义单元 VarDec，由于数组结构类型的标志在变量后，所以不能只使用综合属性，要结合 Specifier 生成的结构进行操作，将获得的类型作为基础类型，在每一层结构中，将方括号中间的词法单元的内容取出，获得 size，上一层的类型作为数组结构中的 elem，获得的结构类型返回，最后一层的类型范围即为数组的类型。

为了方便后面的语义分析，构造了类型比较函数，对于两个类型，首先判断 kind 是否相等，基础类型比较直接比较 basic 大小，数组类型中递归比较 elem 类型，函数类型中先比较参数数量是否相等，之后对两个函数的参数链表中的每个参数只比较类型，不比较名称，由于我们是 11 组，选作要求 2.3，所以对于结构比较，需要比较结构体域中的每个变量，实

现方法和函数参数的比较方法相似，但是每个变量的相对位置必须一致，只比较类型，不比较名称，实现结构等价。

语法树的结构部分，即语句之间的连接，核心思想是把下层的结构传递到上层，就是使用了递归的方法对语法树进行分析。

对于最复杂的语法单元 Exp，针对不同错误情况进行分析，主要进行的是类型比较，将下层的类型属性传递到上层来，同时针对操作符的不同进行判断，只有同一属性的变量才能进行操作，对于数组或者函数出现在 Exp 里的，判断方法是根据节点的特征构建一个新的类型，与在表中查到的类型进行比较，如果不同则报错，对于出现的结构类型，先查找到确定的结构，之后再结构的节点链表中寻找是否存在这个域内定义，如果不存在就报错。有关于左值的错误也出现在这里，从语法的角度来判断左值，判断语法单元是否满足结合条件。

本次实验思想并不复杂，要求也只是规定的 17 个错误类型，但是实现过程繁琐，很容易出错，在使用了实验要求上的测试用例进行检测后，我们又自己创造了成吨的测试用例，发现了各种奇奇怪怪的错误类型，往往是由于对语法规则的意义理解不够深刻导致的细节性的错误，最终修改的版本基本能满足实验要求。