《编译原理》实验二 语义分析

李泽昆，杜映潇

一、完成情况

完成了必做内容和选做要求 2.1，并对讲义上的样例和自己构造的样例进行了测试。

程序能够进行语义分析并检查 17+2 种错误类型。

二、实验思路

实验二是在实验一构造的语法树的基础上进行的。我们的思路是从根节点<Program>开始，对语法树进行深度优先遍历，在此过程中，进行符号表的构造以及类型的检查。具体而言，我们在 Semantic.c 文件中为每一类语法单元都编写了函数，用以指示每当遇到这一类的语法单元的节点时，应当对该节点及其子节点进行怎样的操作。与构造语法树的 Bison 代码相类似，用以进行语义分析的函数彼此之间也是需要递归调用的。

三、数据结构的设计

在本次实验中，我们需要自定义数据结构用于存放符号表内各个表项的相关信息，参考手册中的实现，以及综合考虑程序中可能出现的情况，我们设计的符号表结构如下图所示：

```
struct FieldList_ {
    char* name;
    Type type;
    FieldList tail;
};
```

具体域对应的含义与实验手册中的一致，此处不再赘述。

接下来问题的关键是设计上述结构体中 Type 这一类型。考虑符号表能存储的符号 ID 其可能是基本类型（如 int、float）的 ID、数组的 ID、结构体类型的 ID 和函数的 ID，因此我们改进了手册中提供的关于 Type 的定义，如下图所示：

```
struct Type_ {
    enum { BASIC, ARRAY, STRUCTURE, FUNCTION } kind;
    union {
        int basic;
        struct { Type elem; int size; } array;
        Structure structure;
        Function function;
    } u;
    char* funcName; /* This field is only valid when Type_ is pointing to a paramter */
};
```

上图中的域 kind 代表了该 ID 的类型，联合体类型的域 u 表示的是不同类型的 ID 所需的额外信息（例如对于基本类型来说就是该 ID 是 int 还是 float），域 funcName 比较特殊，从注释中可以看出，我们用其指向表示函数参数的 ID 对应的函数名。这样做的原因在于，由于我们需要完成选做 2.1，因此需要支持函数声明，而函数声明和定义的形参名可以相同，此处便起区分之用，使得此种情况下不会报错。

接下来，我们需要设计 Structure 和 Function 这两个类型，如下图所示：

```
struct Structure_ {
    char* name;
    FieldList member;
    Structure tail;
};
```

```
struct Function_ {
    int isDeclared;
    int isDefined;
    int linenum;
    Type retVal;
    FieldList parameters;
};
```

对于 Structure 我们认为需要记录其名称和域，其中 tail 为处理 Hash 表冲突时所用到的域，对于 Function 我们认为需要记录是定义还是声明，所在行号（报错时用到）、返回值以及参数列表。

最终，有了上述的数据结构，我们定义了类型为 FieldList 的数组 varTable 表示符号表、类型为 FieldList 的数组 fieldTable 用于临时存放结构体定义时的域信息（用于报错）和类型为 Structure 的数组 structTable 用于记录定义过的结构体信息（用于检测变量名称和结构体名称是否相同）。

四、类型检查的方法

（1） 必做部分

必做部分的错误的检查方法可以分为以下三种：

- 在符号表中查找，若不存在则报错，如错误类型 1,2，14，17
- 向符号表中插入，若已有同名同类型的对象则报错，如错误类型 3,4，15,16
- 处理表达式时对变量类型进行检查，如错误类型 5~13

尽管检查各个错误类型时需要注意的细节不尽相同，但是核心就是通过对符号表的操作，确定当前产生式所需要的上下文信息。

（2） 选做 2.1

为了完成选做 2.1 的任务，我们在设计数据结构的时候进行了相应的处理。对于每一个 struct Function_变量，它都有两个域 isDeclared 和 isDefined 来记录当前函数是否已被声明和定义。

在向符号表中插入一个新的函数声明时，若符号表中已有同名的函数，则检查参数类型和返回值类型是否一致，不一致则报错误类型 19；在向符号表中插入一个新的函数定义时，若符号表中已有已被定义的同名函数，则也报错误类型 19。

在 Program 函数的最后，遍历符号表中的所有函数，若还有未被定义的函数（isDefined==0），则报错误类型 18。

五、匿名结构体的实现

对于匿名结构体，我们同样也会把它插入符号表中，这就需要我们为它自动地指定一个名字。为了避免与源代码中的非匿名结构体的名称冲突，我们依次将匿名函数体的名字指定为“00”，“01”……由于 C--语言不允许以数字开头的标识符，因此这样不会产生冲突。