编译原理实验二实验报告

161220092 孟华 dreamanne15@gmail.com

161220091 马可欣 3476769672@qq.com

一、实现功能

1. 语义分析，选做2.1，支持函数声明。
2. 类型检查，除了规定的17种语义错误+函数声明错误，还实现了if while的判断只能是int型判断

二、如何编译

1. 编译命令：make　清除可执行文件：make clean
2. 测试命令：./parser Test/test1

三、数据结构介绍

使用的pdf中提供的Type_和FieldList_的定义，同时增加了一些自己的域方便实现。符号表、函数表利用散列存储，类型表用数组存储（因为觉得类型并不会特别多）。

```
struct FieldList_
{
    char name[maxIdLength]; // FieldList name
    Type type;
    FieldList next;
    char isLeftValue[10]; // 这个field是否是左值表达式，用于ASSIGOP判断
    char isFollowEqual[10]; // 这个field是否跟着=，用于检查struct中是否有赋值
    int line; // 记录这个域所在的行号
    enum { DEFVAR, DECVAR } status; // 变量是在函数声明中出现，还是在其他地方出现
};
```

```
struct FuncList_ // 函数结构体
{
    enum { DEC, DEF } status; // 函数的状态是声明还是定义
    Type return_type; // 返回值
    char name[maxIdLength]; //函数名
    FieldList parameters; // 参数链
    FuncList next;
    int line; // 函数第一次出现的行号
};
```

四、实验设计的亮点与困难解决

1. 首先，我们进行了完整的框架设计，封装了不同层次的函数接口，使得整个程序的易读性提高。在散列表层面，有getHashIndex(char* string)；符号表层面有insertSymbol(FieldList f),向符号表中插入

新符号， getSymbol(char* name), 查询并取到符号表中名字为name的符号地址，generateField(char* name, Type type) 形成一个新的FieldList域，简化代码操作。类型表和函数表也类似。

2. 同时因为增加了函数声明，所以需要在语法分析结束之后，判断是否有函数未定义，我们再Main函数里增加了对于函数表的遍历，如果遇到status为DEC的function，则输出这一错误。
3. 困难一：因为语法树是自底向上归约的，所以先识别到VarList，再向上规约到函数声明或定义，而且函数形参的作用域也是全局的，所以我一开始的实现就是识别到VarList之后insertSymbol插入到符号表中，如果已经有定义了，则报重复定义错误。但是引入函数声明之后，如果函数声明是正确的，那么这个形参不应该报错。解决办法是，首先为insertSymbol引入ifPrint参数，指示是否需要在insertSymbol内部输出错误信息，而在VarList当中只插入符号不输出错误，并且对于FieldList引入stauts enum类型，标识是声明中的形参还是别的地方的定义。在规约到可以区分出函数声明的时候【自己添加的一句文法：ExtDef : Specifier FunDec SEMI】，将所有的FunDec中的参数的status置为DECVAR，因为都是先声明后定义，所以这样处理使得声明中不会报错，定义中不会和声明中的形参报错。
4. 困难二：Error type3和Error type15的区分，相信很多同学都遇到了这个问题orz。我们的解决办法是，对于DefList涉及到的重定义错误，同样是将错误符号加入到errorSymbol错误表中，到上层之后再决定是哪种错误分别相应输出。
5. 为报错加了颜色以及报错说明，方便更好的读取。