编译原理Lab2实验报告

个人信息

191250012 陈益超

功能实现

程序实现了对c--语言的简单语法分析，包括发生语法错误时的提示和没有发生错误时的语法树打印。

首先需要编写CmmParser.g4语法规则文件。主要参考实验提供的附录C--语法来进行编写。附录中给出了基本的语法规则，在这些规则的基础上进行一定程度的改造，主要是将一些左递归或右递归的文法改成不递归，和考虑到非终结符exp的优先级和结合性，对exp的产生式进行改造。

然后是进行语法树的构建和遍历。遍历的时候，使用了listener机制来获取树的信息，调用Walker的方法来遍历。

打印语法树

语法树的输出要求每条词法或语法单元的信息独占一行，子节点的信息相对于父节点的信息要缩进两个空格，这就要涉及空格数量的计算。在ParseTreeWalker 的 walk 方法中，有enterRule(listener, r)和exitRule(listener, r)方法调用，这两个方法又会调用listener的enterEveryRule和exitEveryRule。于是，我在自己实现的CmmWalkListener中添加一个全局变量tabs记录缩进的次数。每次调用enterEveryRule，对tabs加一，并在输出时用tabs*2得到空格的数目并进行打印；在exitEveryRule时对tabs减一。

语法树的输出还要求判断节点的类型，根据不同节点类型输出不同的信息。这些信息都使用parser提供的静态方法来获取。

错误的打印

模仿了ConsoleErrorListener 的写法，自定义一个错误监听器CmmErrorListener，并且重写了syntaxError() 方法，按要求编写输出格式。在Main中，通过调用parser的removeErrorListeners()方法将原来的默认的错误监听器ConsoleErrorListener移除，再通过

```
parser.addErrorListener(CmmErrorListener.INSTANCE);
```

将自己实现的监听器加入，达到自定义错误打印的目的。

勘误备选分支

对数组下标类型的错误进行了手动报错。首先改写数组相关的语法规则，将数组下标的类型扩展为INT、ID、FLOAT，即忽略了下标是ID、FLOAT的错误，并在语法规则的合适位置加入java代码，这样在进行匹配时遇到这类型的错误就会陷入勘误备选分支。

```
varDec : ID (LB (OCT_INT|DEC_INT|HEX_INT|{notifyErrorListeners("Missing
closing ')'");}FLOAT|{notifyErrorListeners("Missing closing ')'");}ID) RB)*
```

实验过程遇到的一些状况

- 空串匹配

实验文档中说明语法单元产生了ε则无需打印该语法单元的信息。一开始以为ε会是一个可识别出的字符，于是采用正则表达式来匹配。后来发觉语法规则中根本就没有关于ε的情况，这时候产生的就是一个空字符串，于是开始根据字符串长度来判断。

- 勘误备选分支java语句的插入位置

java语句可以插入语法规则的任意位置，并且会在生成的Parser类中的对应位置，生成一模一样的java语句。所以插入位置很关键。一开始是在上面所写的规则中，紧接着ID或FLOAT的后面插入语句，期望在识别出ID或FLOAT后就陷入勘误备选分支。而实际生成代码时，已经对节点加一了，获取节点信息得到的就是跟在ID或FLOAT的下一节点的信息，发生错误。于是将插入位置提前。