Lab.5 语法分析实验报告

一、实验目的和内容

1. 实验目的

* 熟悉C语言的语法规则，了解编译器 语法分析器的主要功能。
* 熟练掌握典型语法分析器构造的相关技术和方法，设计并实现具有一定复杂度和分析能力的C语言语法分析器。
* 了解 ANTLR 的工作原理和基本思想，学习使用工具自动生成语法分析器。
* 掌握编译器从前端到后端各个模块的工作原理，语法分析模块与其他模块 之间的交互过程。

1. 实验内容

该实验选择C语言的一个子集，基于BITMiniCC构建C语法子集的语法分析器，该语法分析器能够读入词法分析器输出的存储在文件中的属性字符流，进行语法分析并进行错误处理，如果输入正确时输出JSON格式的语法树，输入不正确时报告语法错误。

二、实验过程和步骤

1. 语法设计

对实验中的文法进行了补全和修正，支持各种运算表达式、基本变量声明、数组声明、赋值语句、循环语句、分支语句等。

部分语法规则如图1所示。

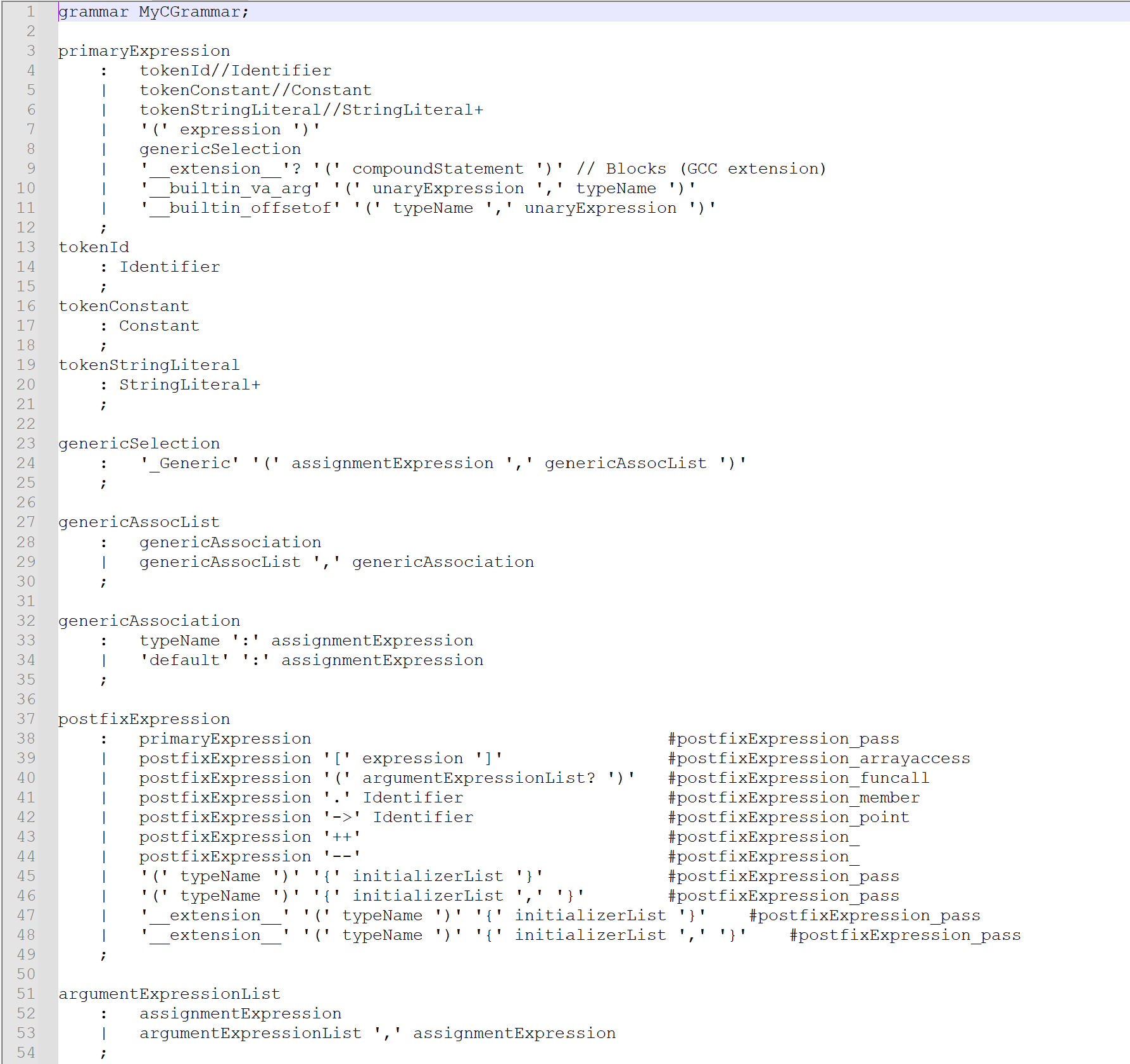


图 1：语法规则

1. 自动生成

将BITMiniCC/lib中的antlr-4.8-comple.jar复制到MyCGrammar.g4所在路径下，然后在该路径下启动控制台，执行命令

java -jar antlr-4.8-comple.jar MyCGrammar.g4

生成的文件如图2所示。

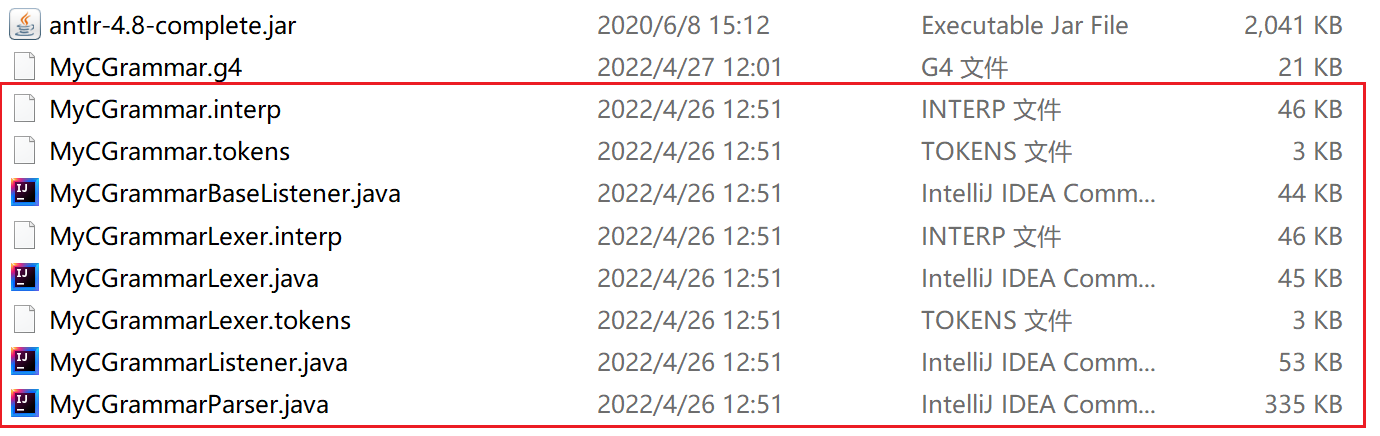


图 2：自动生成

将生成的文件和antlr-4.8-complete.jar构建为MyCGrammar.jar，并添加为BITMiniCC的库文件。

1. 词法分析

在软件包bit.minisys.minicc.scanner中新建Java类MyParser。

为了嵌入BITMiniCC框架，MyParser类实现了IMiniCCScanner中的run方法，用于调用ANTLR自动生成的词法分析器MyCGrammarLexer。

源码如图3所示。

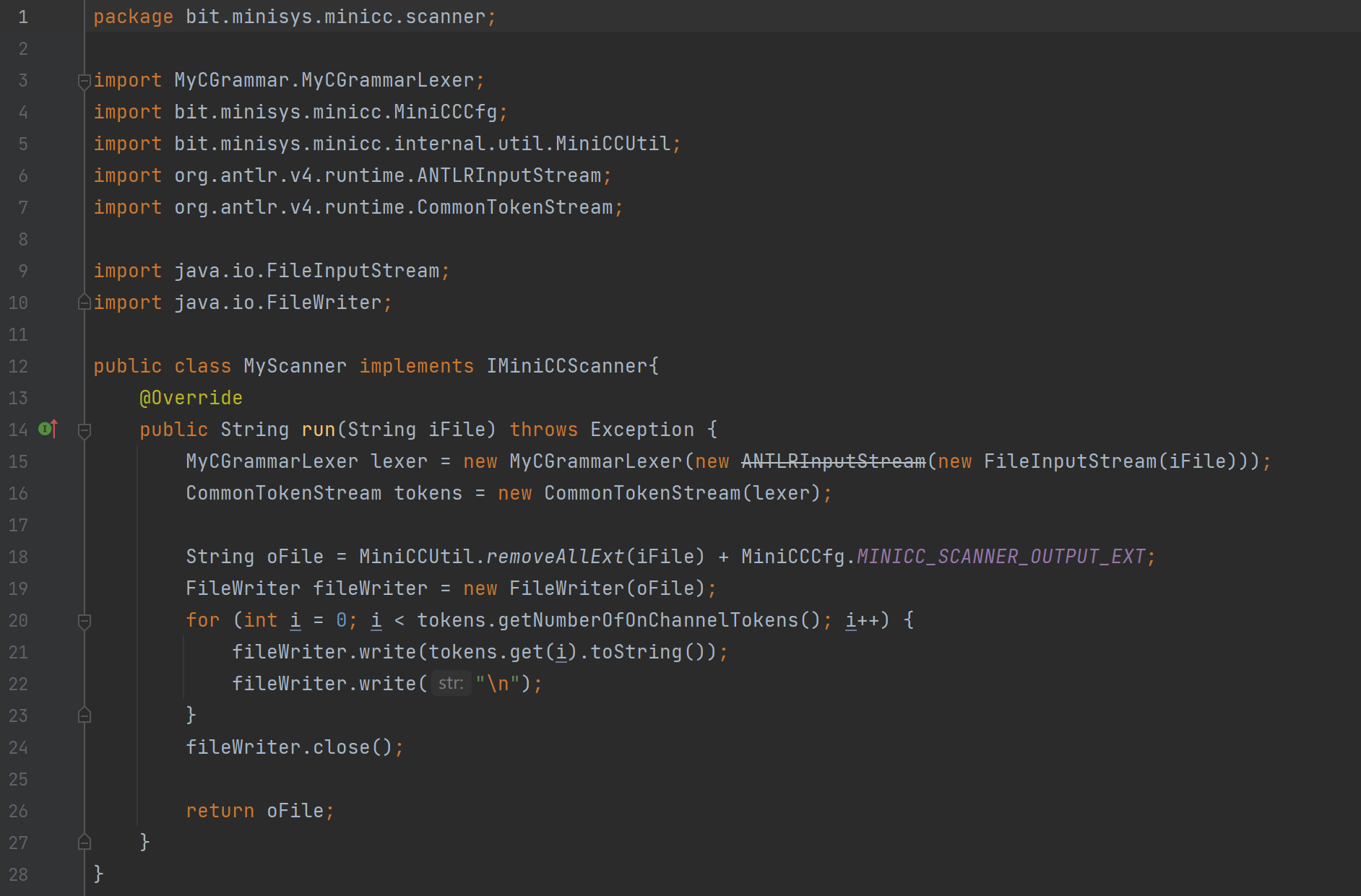


图 3：MyScanner源码

1. 语法分析

在软件包bit.minisys.minicc.scanner中新建Java类MyListener，继承了由ANTLR自动生成的虚类MyCGrammarBaseListener，并实现了其中的方法。

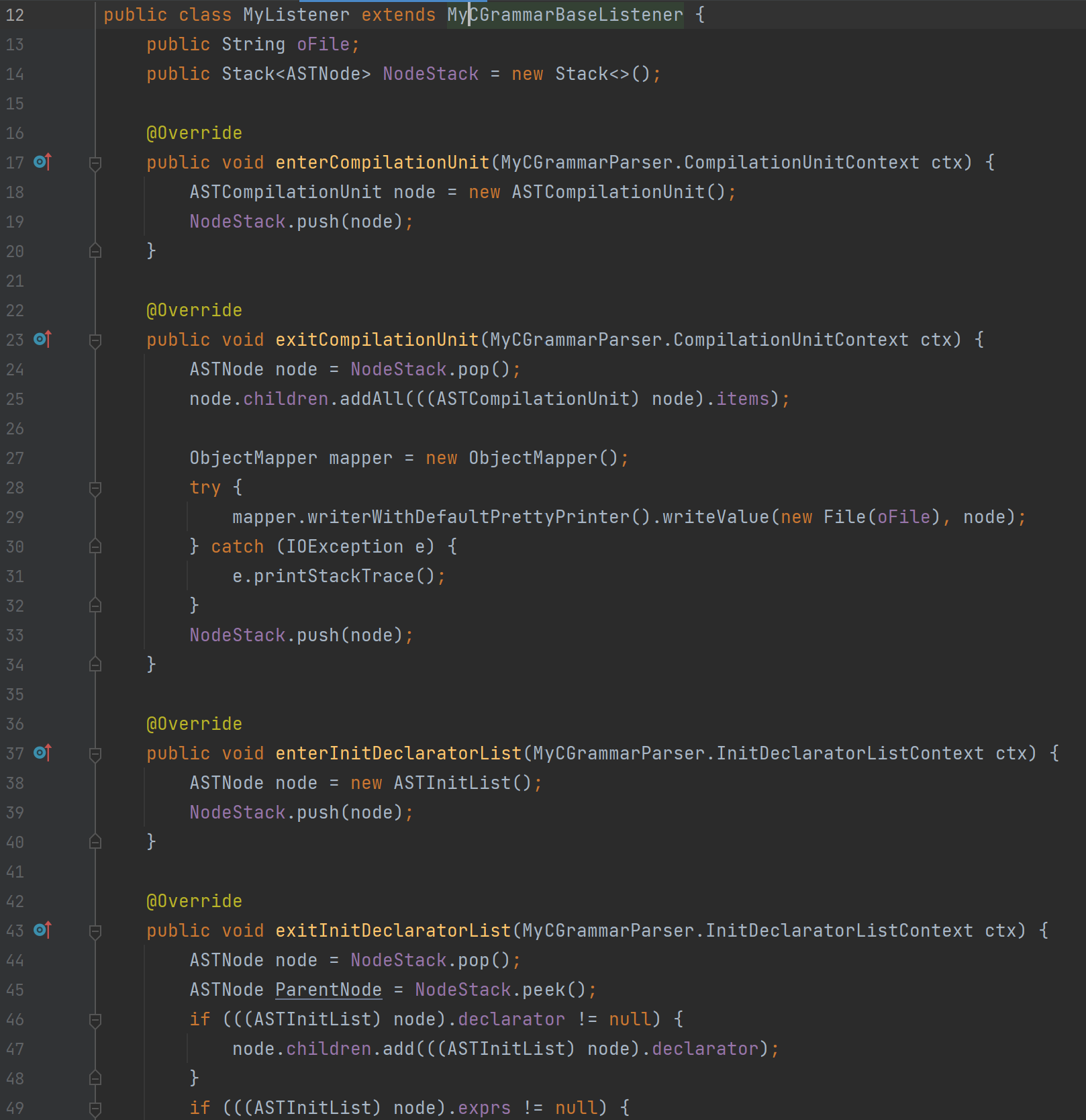
部分源码如图4所示。

图 4：MyListener源码

然后在同一软件包中新建Java类MyParser，并实现接口IMiniCCParser中的run方法。

在run方法的实现中调用了由ANLTR自动生成的语法分析器MyCGrammarParser，得到测试用例对应的具体语法树。然后调用ANTLR运行时API中的ParseTreeWalker中的walk方法。walk方法会根据参数中传入的MyListener实例，按照MyListener类中各方法定义的规则遍历具体语法树，生成相应的AST树。

源码如图5所示。

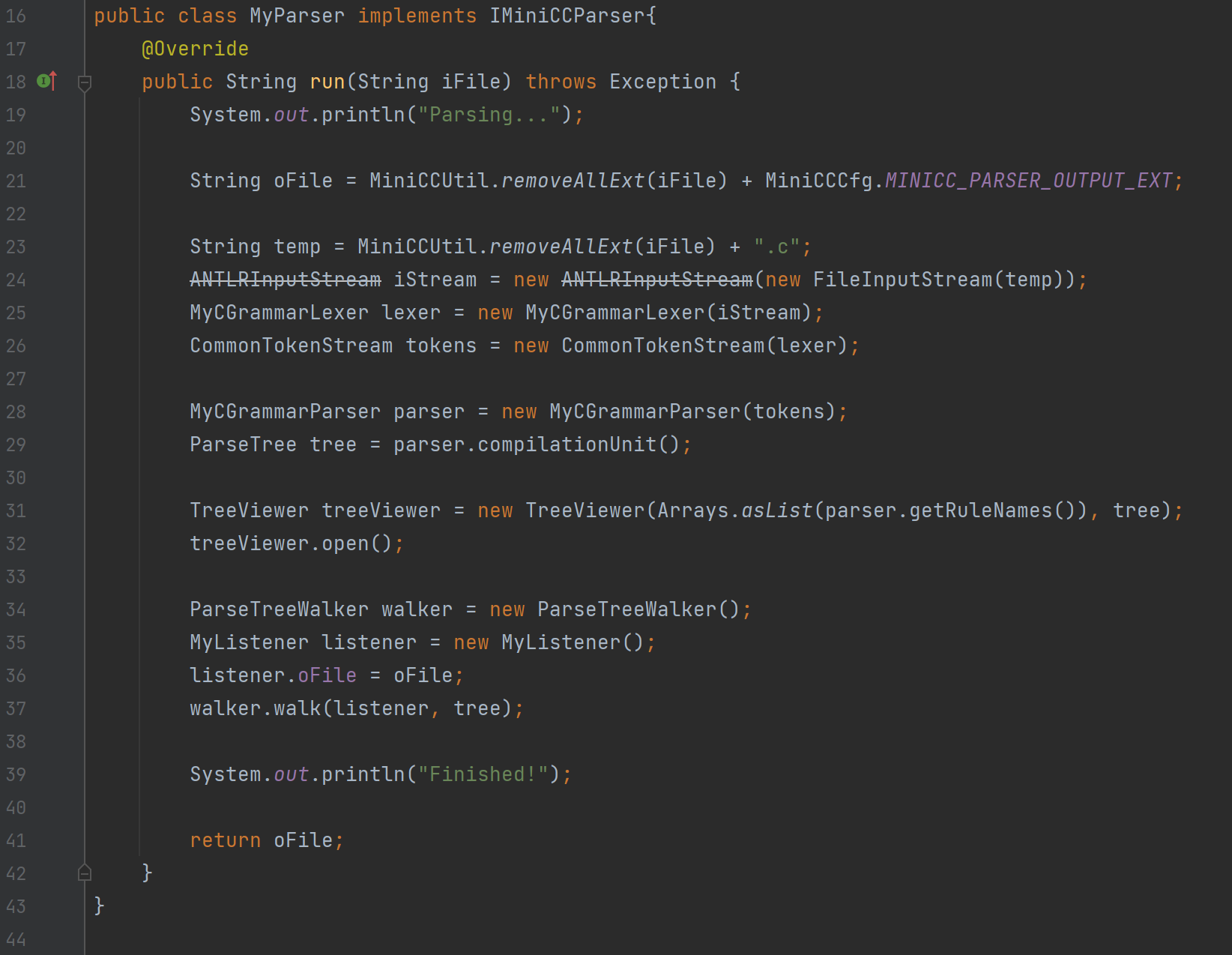
三、运行效果截图

图 5：MyParser源码

测试用例

int main() {

    int x[10];

    x[0] = 0;

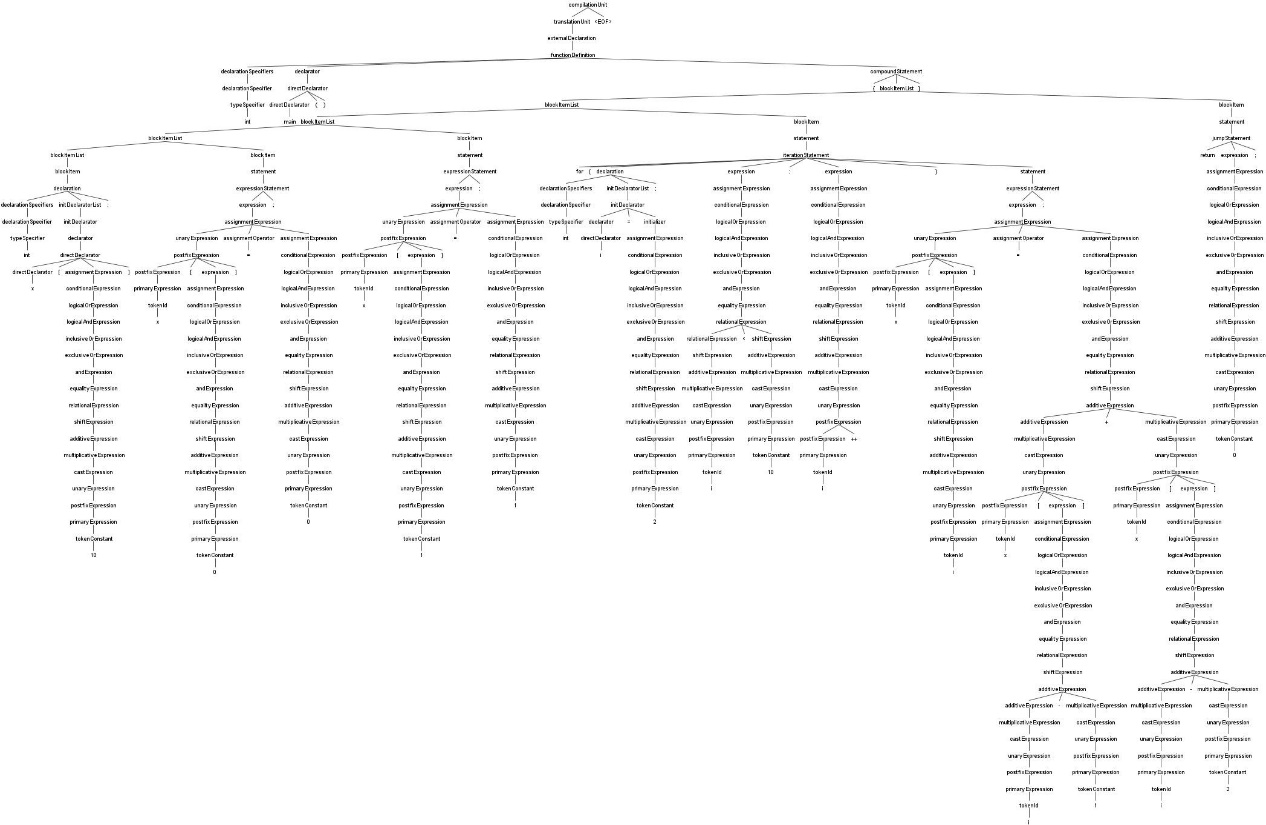
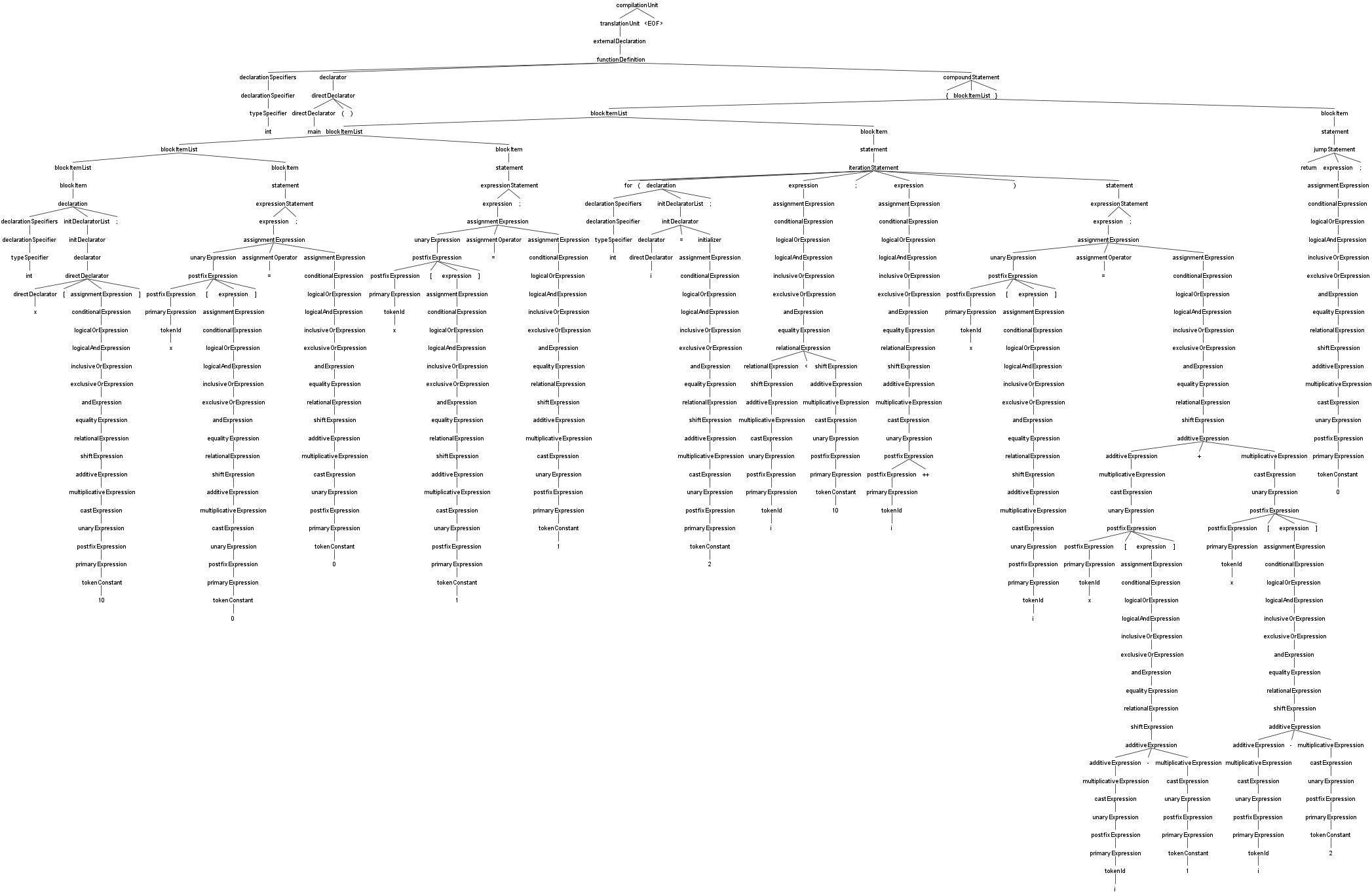
    x[1] = 1;

    for (int i=2; i<10; i++)

        x[i] = x[i-1] + x[i-2];

    return 0;

}

具体语法树

 部分.json文件

四、实验心得体会

在本次实验中，尽管使用了自动生成工具ANTLR，但是由于语法规则扩充得过于复杂，导致MyListener的代码量变得极其庞大，耗费了大量时间。

在嵌入框架的过程中，由于未能找到将词法分析器生成的.tokens文件还原为CommonTokenStream的方法，导致在MyParser中再次调用了词法分析器对源文件进行分析，产生了代码冗余。