1.1 语法分析的任务要求

该实验选择 C 语言的一个子集，基于 BIT-MiniCC 构建 C 语法子集的语法分析器，该语法分析器能够读入 XML 文件形式的属性字符流，进行语法分析并进行错误处理，如果输入正确时输出 XML 形式的语法树，输入不正确时报告语法错误。

需要说明的是，能够分析的输入程序依赖于选用的语法子集，而输出的语法树的结构又与文法的定义密切相关。

可参考扩充的 C 语言文法

如下为 C 语言文法的一个子集：

```
CMPL_UNIT     : FUNC_LIST
FUNC_LIST     : FUNC_DEF FUNC_LIST | ε
FUNC_DEF      : TYPE_SPEC ID ( ARG_LIST ) CODE_BLOCK
TYPE_SPEC     : int | void
PARA_LIST     : ARGUMENT | ARGUMENT , PARA_LIST | ε
ARGUMENT      : TYPE_SPEC ID
CODE_BLOCK    : { STMT_LIST }
STMT_LIST     : STMT STMT_LIST | ε
STMT          : RTN_STMT | ASSIGN_STMT
RTN_STMT      : return EXPR
ASSIGN_STMT   : ID = EXPR
EXPR          : TERM EXPR2
EXPR2         : + TERM EXPR2 | - TERM EXPR2 | ε
TERM          : FACTOR TERM2
TERM2         : * FACTOR TERM2 | / FACTOR TERM2 | ε
FACTOR        : ID | CONST | ( EXPR )
```

读者可以在此基础之上进行文法扩充，包括全局变量声明，循环语句、分支语句、函数调用语句以及 switch 语句等。要求至少包括局部变量声明语句、赋值语句、返回语句、一种分支语句（if, if-else, switch 等）和一种循环语句（for, while, do-while 等）。

1.2 实验过程与方法

在 BIT-MiniCC 框架下，可以按照如下步骤完成语法分析实验：

（1）参照 3.3.2 节给出的文法，扩充定义自己希望实现的 C 语言语法子集。

参考文法只给出了函数定义以及简单的表达式相关的文法。局部变量声明、分支语句以及循环语句等需要自己进行扩充。主要采用自顶向下的分析方法时，不能有左递归，避免文法产生式的多个候选式存在公共因子。如果出现左递归或者公共因子，则可以通过文法等价变换进行消除。

（2）从递归下降分析方法、LL(1)分析方法、算符优先分析方法、LR 分析方法中选择一种，作为 BIT-MiniCC 框架中设计并实现语法分析器的指导方法。

（3）构建语法分析器，BIT-MiniCC 中已经定义了一个类 CMyMiniCCParser，并定义了 run 方法，实验以该类为主进行。语法分析的输入为词法分析的输出，因此语法分析器首先要读入 xxx.token.xml 文件；在分析的过程中构建语法树；

（4）将语法树输出为 xml 文件；

目前已有的分析方法包括递归下降、LL(1)和 LR 等多种分析方法，可以选择其中的一种实现，递归下降更为直观。

例如，基于框架自带的文法及其对应的实现，当输入为如下的程序时：

```
int main(){
    int a=1;
    a = a * 2;
    return a;
}
```

得到的语法树如下所示：

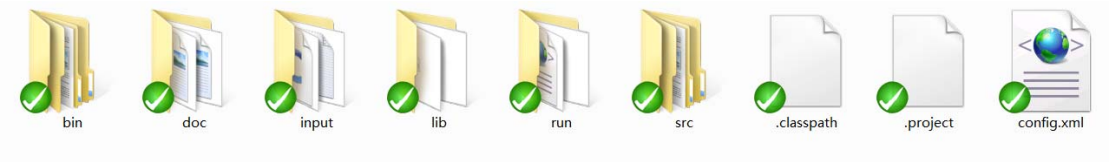
对应的输出的 xml 的部分内容为：

```
<?xml version="1.0" encoding="UTF-8"?>
<ParserTree name="test.tree.xml">
  <compilationUnit>
    <translationUnit>
      <externalDeclaration>
        <functionDefinition>
          <declarationSpecifiers>
            <declarationSpecifier>
              <typeSpecifier>int</typeSpecifier>
            </declarationSpecifier>
          </declarationSpecifiers>
          <declarator>
            <directDeclarator>
              <directDeclarator>main</directDeclarator>
            <punctuation>(</punctuation>
            <punctuation>)</punctuation>
            </directDeclarator>
          </declarator>
          <compoundStatement>
          <punctuation>{</punctuation>
            <blockItemList>
              <blockItemList>
                <blockItemList>
                  <blockItem>
                    <declaration>
                      <declarationSpecifiers>
                        <declarationSpecifier>
                          <typeSpecifier>int</typeSpecifier>
                        </declarationSpecifier>
                      </declarationSpecifiers>
                      <initDeclaratorList>
                        <initDeclarator>
                          <declarator>
                            <directDeclarator>a</directDeclarator>
                          </declarator>
                        <separator>=</separator>
                          <initializer>
                            <assignmentExpression>
```

1.3 实验提交内容

本实验要求提交语法分析器实现源码，C/C++需提供对应的可执行程序（不需要编译的中间文件），Java 提供编译后的 class 文件或者 jar 包，每个人提交一份实验报告。

提交目录如下所示：



实验报告放置在 doc 目录下，应包括如下内容：

- 实验目的和内容
- 实现的具体过程和步骤
- 运行效果截图
- 实验心得体会