**编译原理**

**第二次实验报告**

姓名： 梁芮槐

学号： 2019302789

专业： 软件工程

**西北工业大学**

**2022年6月3日**

目录

[1. 实验环境 3](#_Toc105162131)

[2. 实验内容 3](#_Toc105162132)

[3. 实验流程 3](#_Toc105162133)

[3.1概要设计 3](#_Toc105162134)

[3.2详细设计 5](#_Toc105162135)

[4. 测试说明 7](#_Toc105162136)

[4.1测试流程 7](#_Toc105162137)

[4.2测试结果 8](#_Toc105162138)

[5. 实验总结 9](#_Toc105162139)

# 实验环境

基于实验一的Antlr4环境，配置StringTemplate实验环境：

1. 在IDEA中安装StringTemplate插件
2. 修改<pom.xml>，添加StringTemplate依赖项

# 实验内容

1. 在实验一内容的基础上，完成语义分析。本次实验语义分析的内容有以下3条：
2. 命名冲突。同一命名空间内，不能出现相同名字的接口定义。
3. 未定义即使用。struct结构需要先定义才能使用。
4. 字面量类型检查。字面量的数据类型需要和变量类型相同或兼容。

实现能够对上文所提的3条语义错误进行检查和报错。报错提示要给出错误位置，以及错误类型。并给出测试说明文档。

1. 从抽象语法树生成对应的C++代码，测试用例参考“**代码生成用例集.zip**”，依据用例集中的输入输出，总结抽象语法节点的生成规则，并编写代码生成模块。（代码生成可使用ppt中第6章代码生成的方法，也可以采用附录A中介绍的StringTemplate模板方式生成目标代码，并配有学习视频“StringTemplate模板语法介绍.mp4”和“StringTemplate模板语法介绍.mp4”）。
2. 针对语义分析和代码生成设计测试用例，给出测试方法描述，提交readme.doc，如果你还有其他需要说明的问题须写在readme.doc中。

# 实验流程

## 3.1概要设计

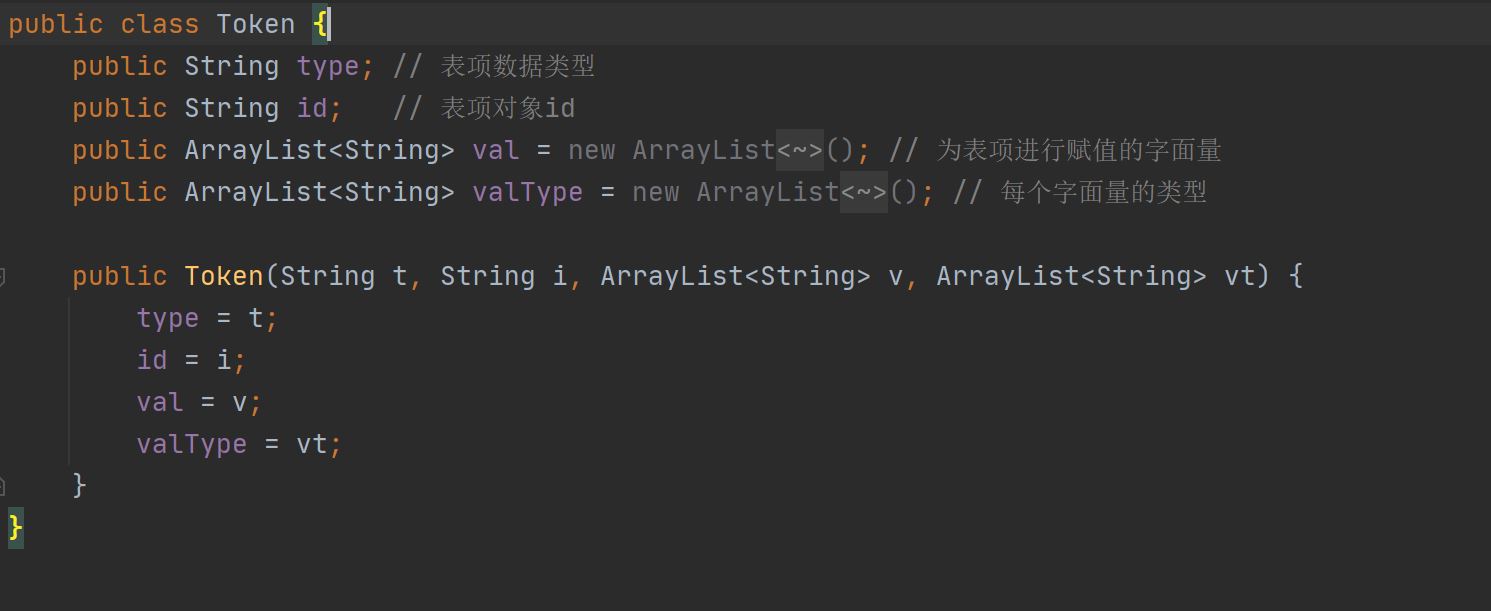
1. 根据语义分析的三条要求，在原有抽象语法树遍历生成的基础之上，设计符号表数据结构如下：



哈希表moduleTable中，Key为module的ID，Value为该module对应的struct哈希表在structTable中的下标；

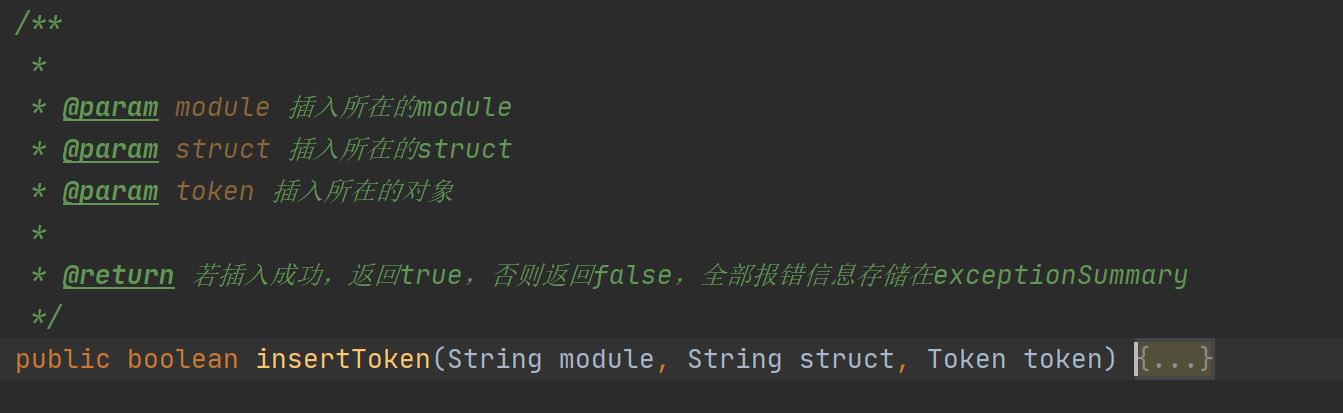
数组structTable中的哈希表，Key为struct的ID，Value为该struct对应的value哈希表在valueTable中的下标；

数组valueTable中的哈希表，Key为value的ID，value为该value相关信息的存储类Token，Token设计如下：

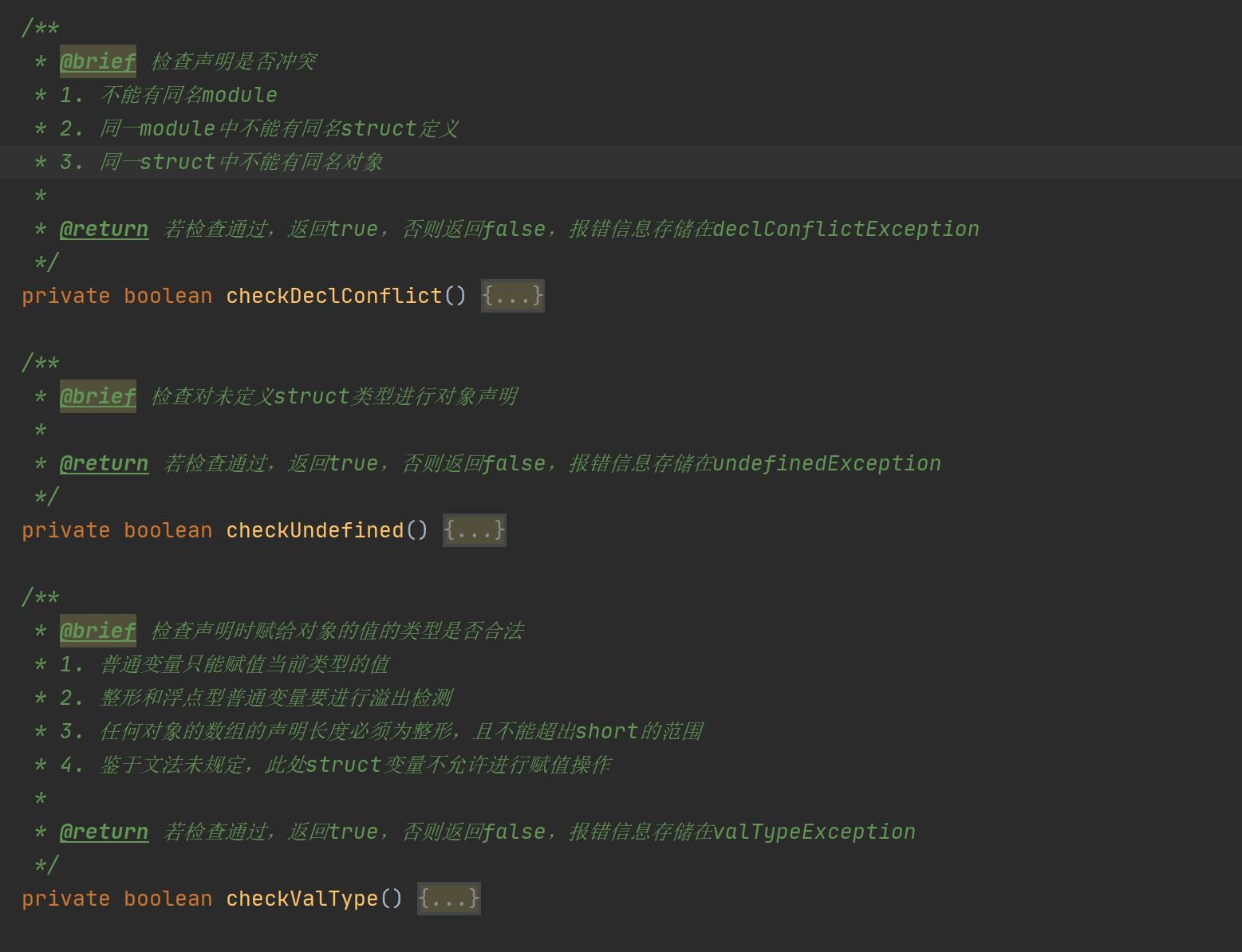


1. 根据设计好的符号表数据结构，设计语义分析实现方法。

符号表插入函数：

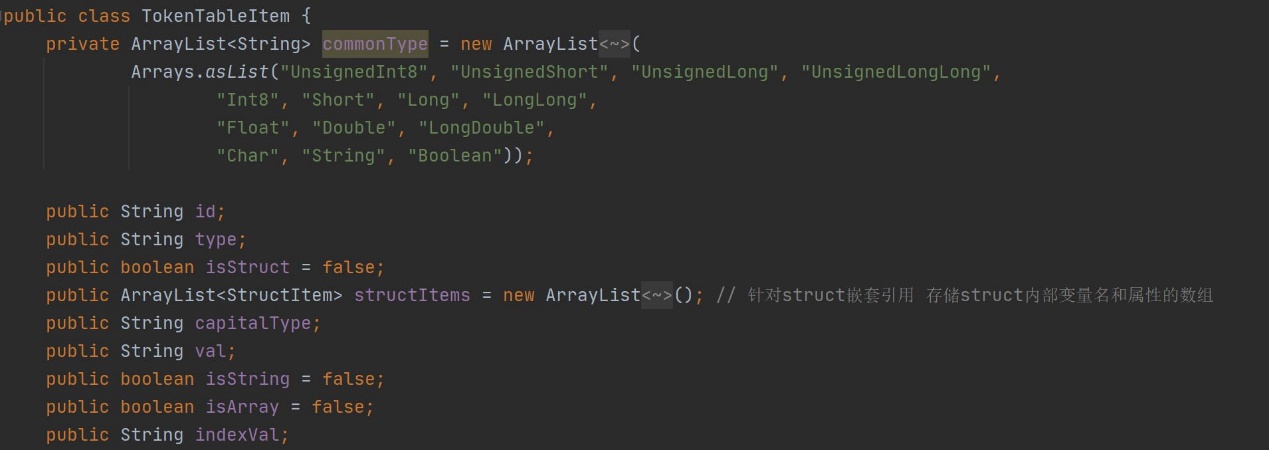


三条语义分析函数：



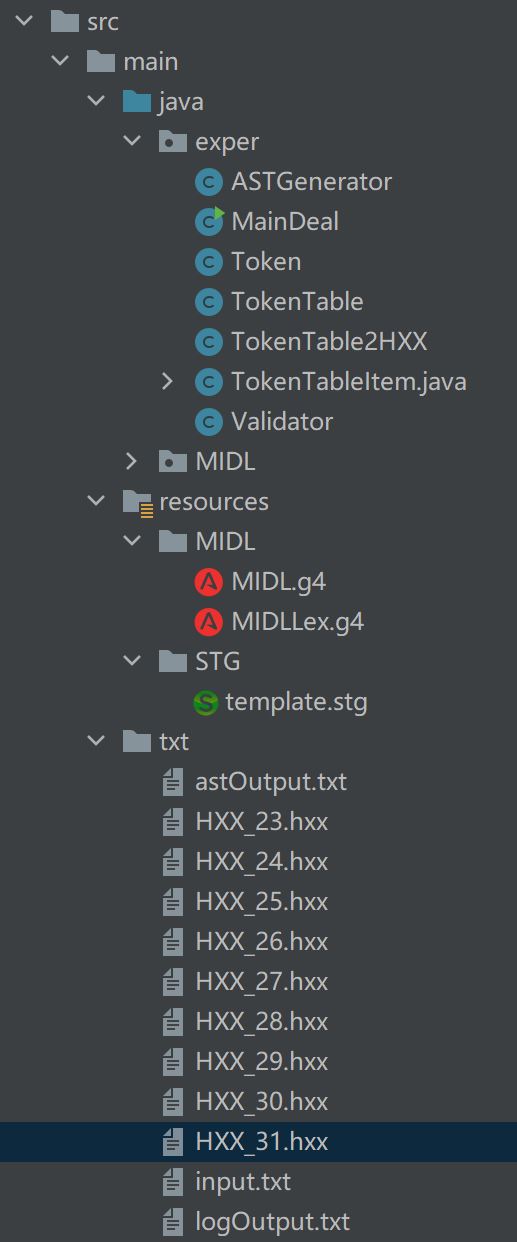
## 3.2详细设计

1. 对于命名冲突，分为三类：
2. 不能有同名module
3. 同一module中不能有同名struct定义
4. 同一struct中不能有同名对象
5. 对于未定义即使用。需要预先考虑到多重module和struct嵌套，需对名称空间编写匹配逻辑。
6. 对于字面量类型检查，分为四类：
7. 普通变量只能赋值当前类型的值
8. 整形和浮点型普通变量要进行溢出检测
9. 任何对象的数组的声明长度必须为整形，且不能超出short的范围
10. 鉴于文法未规定，此处struct变量不允许进行赋值操作
11. 对于.hxx文件的导出，考虑到有struct嵌套的情况，并且需要用到纯大写的类型字符串和首字母大写的类型字符串，设计一个从符号表转.hxx文件的TokenItemTable类型用于传递参数：



同时，根据对5个用例.hxx文件的总结，设计stg模板分为header、struct、footer三个部分，其中struct为主体部分，嵌套多个底层模板。

1. 最终代码文件结构：

java.exper.ASTGenerator：重载MIDLBaseVisitor，实现对分析树的遍历，过程中生成抽象语法树并调用TokenTable建立符号表，获取报错信息。

java.exper.MainDeal：主运行类，读取输入文件，调用ASTGenerator和TokenTable2HXX，在终端输出结果并写入到指定的文件中。

java.exper.Token：符号表底层对象数据结构类，存储对象ID，类型，字面量值与类型。

java.exper.TokenTable：符号表主类，对设计好的符号表数据结构进行管理，对试图插入元素的操作进行语义分析，返回插入结果，并得出报错信息。

java.exper.TokenTable2HXX：导出hxx工具类，使用传入的TokenTable符号表，导出hxx类型的输出。

java.exper.TokenTableItem：TokenTable2HXX的专用数据结构，用于存储导出用得到的数据信息，包括id、type、是否是结构体的标志位、是否是字符串的标志位、是否是数组的标志位等。

java.exper.Validator：用于进行字面量范围合法性检查，根据输入字面量及其类型，判断其是否合法。

java.MIDL：Antlr导出的词法分析代码。

resources\MIDL：MIDL的g4文法文件。

resources\STG\template.stg：.hxx文件的stg模板代码，分为header、struct、footer三个部分。

txt：输入文件和输出文件所在文件夹

# 测试说明

## 4.1测试流程

* 主运行类[java\exper\MainDeal.java](java/exper/MainDeal.java)
* 输入文件[txt\input.txt](txt/input.txt)，可以多行输入，并进行按行语义分析与报错。其中#符号分隔的，下一轮输入不导出.hxx；$符号分隔的，下一轮输入导出.hxx。结束行必须是#或$。如下（部分）：

################################################ 22 功能3 样例4 报错 普通类型的对象赋值字面量须在对应范围内

struct B{

float f = 100000000000000000000000;

};

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ 23 case1

module space{

struct A{

short i1=10;

};

};

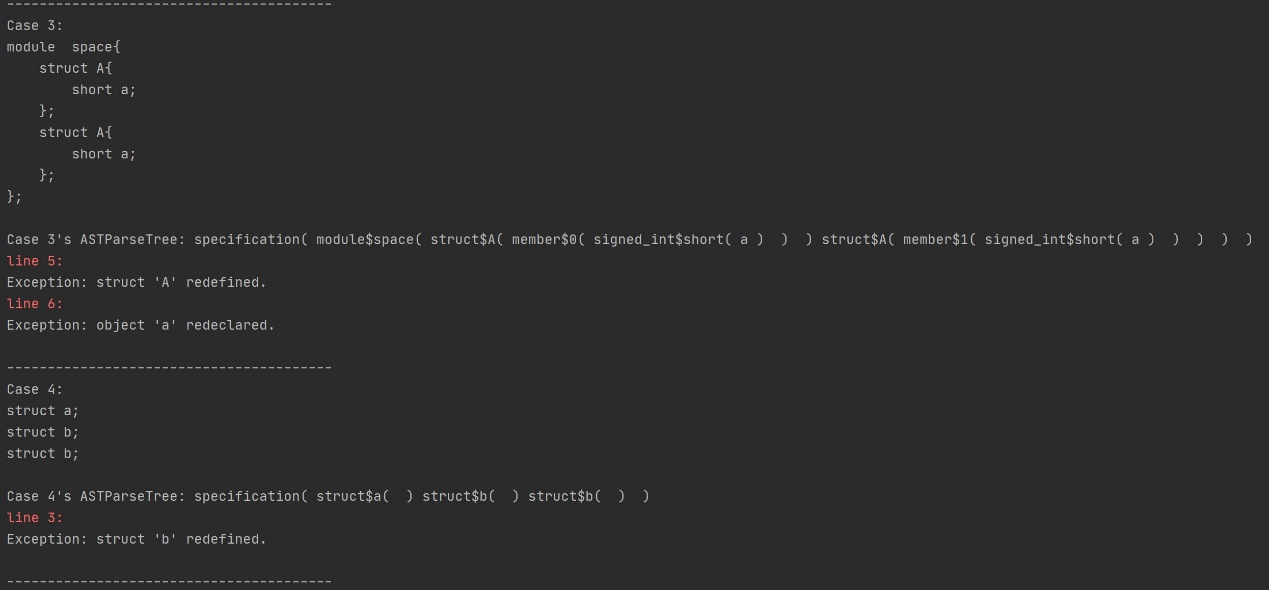
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

## 4.2测试结果

* 语义分析测试说明文档：<语义分析测试说明文档.docx>
* 语义分析与.hxx文件导出综合测试：<README.docx>
* ast格式化输出文件：[txt\astOutput.txt](txt/astOutput.txt)
* 语义分析报错输出文件：[txt\logOutput.txt](txt/logOutput.txt)
* .hxx导出文件所在目录（文件名与[txt\input.txt](txt/input.txt)中测试用例序号一致）：txt

终端输出部分截图：





# 实验总结

1. 在这两次实验中，基本按照要求实现了一个源到源的微型编译器，达到了实验要求。学到了词法分析、语法分析、语义分析和代码生成的代码设计概念，并在编程实现中积累了不少编译原理数据结构和逻辑设计的经验。
2. 实验过程中最大的困难在于需要一个人完成整个端到端的流程，由于能力有限，在设计时常常不能考虑到所有细节，后续的测试和debug大概会花了总时间的2/3。归其原因还是因为设计得不够完善和精细，导致一些问题只能在编码基本完成后修改。