编译原理实验 - 词法分析 & 语法分析

161220084 刘笑今

161220084@smail.nju.edu.cn

1 实验环境

- GNU Linux Release: Ubuntu 18.04.1
- G++ version 7.3.0
- GNU Flex version 2.6.4
- GNU Bison version 3.0.4

2 编译与运行方法 (make)

- make parser: 生成分析器，并拷贝到上一目录；
- make test: 对 Test 目录下的所有 *.cmm 文件进行分析；
- make clean: 删除 make parser 过程中生成的所有中间文件。

3 分析器功能

可以对 C——语言的程序进行词法分析。如果识别到一个串满足词法单元格式，则返回该词法单元供语法分析使用，并创建语法分析树节点。否则报错。

可以对 C——语言的程序进行语法分析。如果识别到一个串满足语法单元格式，则创建语法分析树节点并加入到树中。若遇到错误，则报错并进行错误恢复。

在词法分析中，可以识别单行注释 (//) 和多行注释 (/*...*/)，并且可以识别多行注释右端未闭合的情况，后者可以报出一个词法分析错误。

4 分析器实现简介

4.1 语法分析树的维护

本分析器将所有的终结符号和非终极符号都作为一个语法分析树中的节点。定义了一个 AstNode 数据结构，其中维护了该语法单元的类型，行号，值类型与值，是否是错误类型等信息，以及并列节点指针 AstNode* sibling 和子节点 AstNode* child。

利用以上的数据结构，可以维护一棵语法分析树。对于终结符，在其被识别为词法单元时创建该节点 (在 lexical.l) 中；对于非终结符，在其被识别为语法单元时创建该节点 (在 syntax.y) 中。

4.2 选做部分：注释及其错误类型识别

4.2.1 单行注释

```
\/\/[^\n]*[\n]
```

4.2.2 正确的多行注释

```
\/\*[^*]*\*+([^*/][^*]*\*+)*\/
```

4.2.3 右端未闭合的多行注释

```
\/\*[^*]* | \/\*[^*]*\*+([^*/][^*]*\*+)*([^*/][^*]*)
```