编译原理与设计实验报告

姓名：卜梦煜 学号：1120192419 班级：07111905

# 1. 实验名称

词法分析实验

# 2. 实验目的

（1）熟悉 C 语言的词法规则，了解编译器词法分析器的主要功能和实现技术，掌握典型词法分析器构造方法，设计并实现 C 语言词法分析器；

（2）了解 JFlex 工作原理和基本思想，学习使用工具自动生成词法分析器；

（3）掌握编译器从前端到后端各个模块的工作原理，词法分析模块与其他模块之间的交互过程。

# 3. 实验内容

根据 C 语言的词法规则，设计识别 C 语言所有单词类的词法分析器的确定有限状态自动机，并使用 Java语言，采用数据中心法设计，使用JFlex自动生成实现词法分析器。词法分析器的输入为 C 语言源程序，输出为属性字流。

# 4. 实验环境

jdk-17.0.2，JFlex-1.8.2

# 5. 实验过程与步骤

实验方法为使用Java语言和JFlex自动生成C语言词法分析器，实验步骤包括安装JFlex、编写.l规则文件、JFlex编译.l文件生成Java程序、将该程序嵌入BITMiniCC框架。

## 5.1 安装、配置JFlex

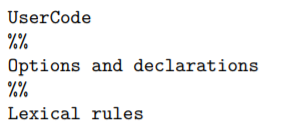
（1）github下载最新版的JFlex压缩包，解压文件夹。

（2）用记事本打开 /bin/jflex.bat 文件，修改jdk安装路径和JFlex安装路径。

（3）将JFlex添加至系统环境变量。

## 5.2 编写.l规则文件

查看JFlex用户手册，JFlex的.l规则文件结构如下：



按照图示结构编写各部分代码：

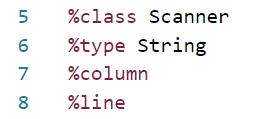
（1）用户代码

该部分包括java程序的包声明和导入语句。本实验导入java.io库进行文件读写操作。

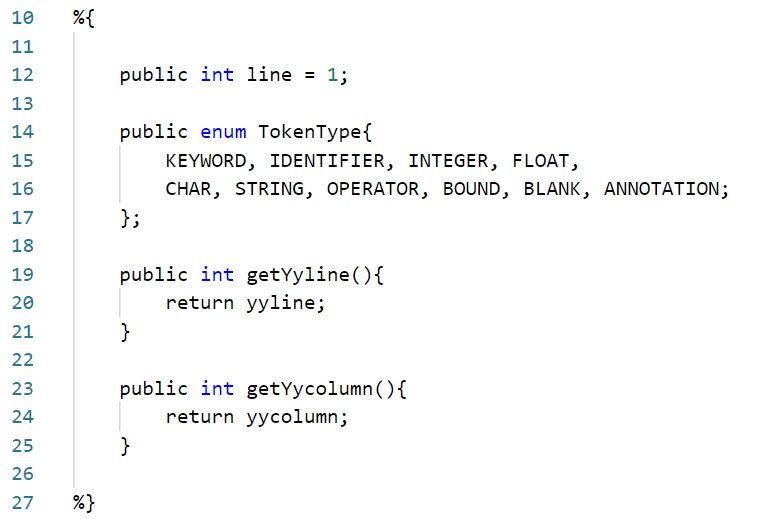
（2）选项与声明

该部分包括生成词法分析器、词法状态声明、宏定义的代码。

首先编写JFlex系统可选项，包括%class定义词法分析器类名，%type定义词法分析器返回值的数据类型，%line开启行计数，%column开启列计数。

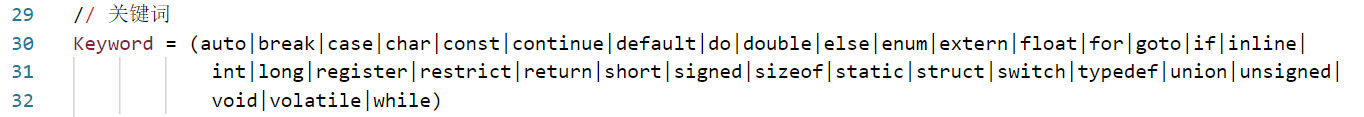


然后编写用户代码，包括变量定义、正则表达式Token的枚举类定义。这部分代码放在%{和%}之间，会原样拷贝到生成的词法分析器文件中。本文定义的Token类型包括关键词、标识符、整型常量、浮点型常量、字符常量、字符串字面量、运算符、界限符、空格、注释，基本覆盖了C语言单词类的所有内容。自定义的函数为行列数的get方法。

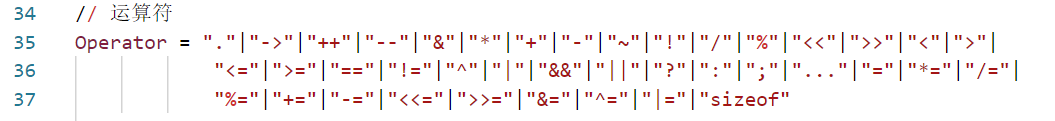


最后根据C语言单词类的定义，编写正则表达式，各Token正则表达式编写如下。

对关键字，按词直接识别C语言所有关键字，正则表达式如下：



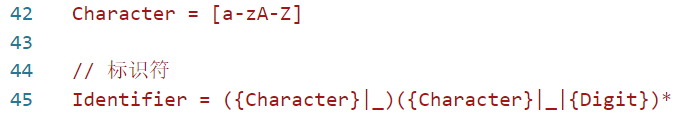
对运算符，识别包括算术运算符、逻辑运算符、关系运算符、位运算符、赋值运算符，其他杂项运算符，直接按照运算符内容编写正则表达式如下：



对界限符，直接识别C语言所有界限符，正则表达式如下：



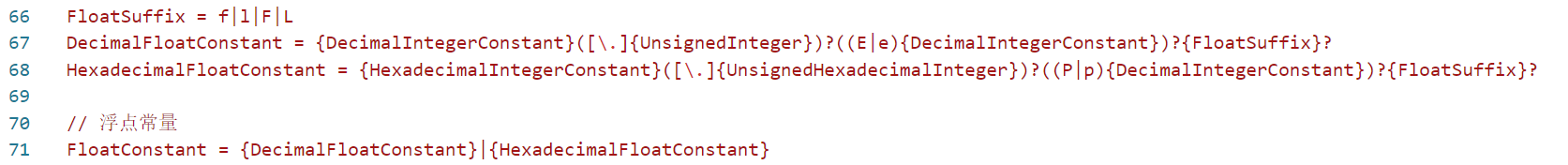
对标识符，格式为非数字开头的数字、字母、下划线，正则表达式如下：



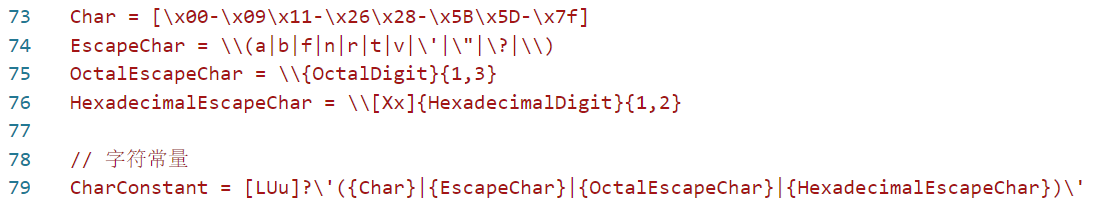
对整型常量，识别包括十进制、八进制、十六进制整型常量。对十进制数，格式为 [符号]<整数部分>[后缀] ；对八进制数，格式为 [符号]<0><整数部分>[后缀] ；对十六进制数，格式为 [符号]<0{X, x}><整数部分>[后缀] 。正则表达式如下：



对浮点型常量，识别十进制、十六进制浮点型常量。对十进制数，格式为 [符号]<整数部分>[.小数部分][{E, e}<十进制整数>][后缀] ；对十六进制数，格式为 [符号]< 0{X, x}><整数部分>[.小数部分][{P, p}<十进制整数>][后缀] 。正则表达式如下：



对字符常量，识别单引号内的前缀加普通ASCII码字符常量、转义符常量、八进制字符常量、十六进制字符常量。普通ASCII码字符常量为ASCII码0~127去除’、\、Tab；转义符常量包括\v等11个；八进制字符常量为转义符的八进制表示，格式为 <\>[000-177]；十六进制字符常量为转义符的十六进制表示，格式为 <\>{X, x}[00-7f]。正则表达式如下：

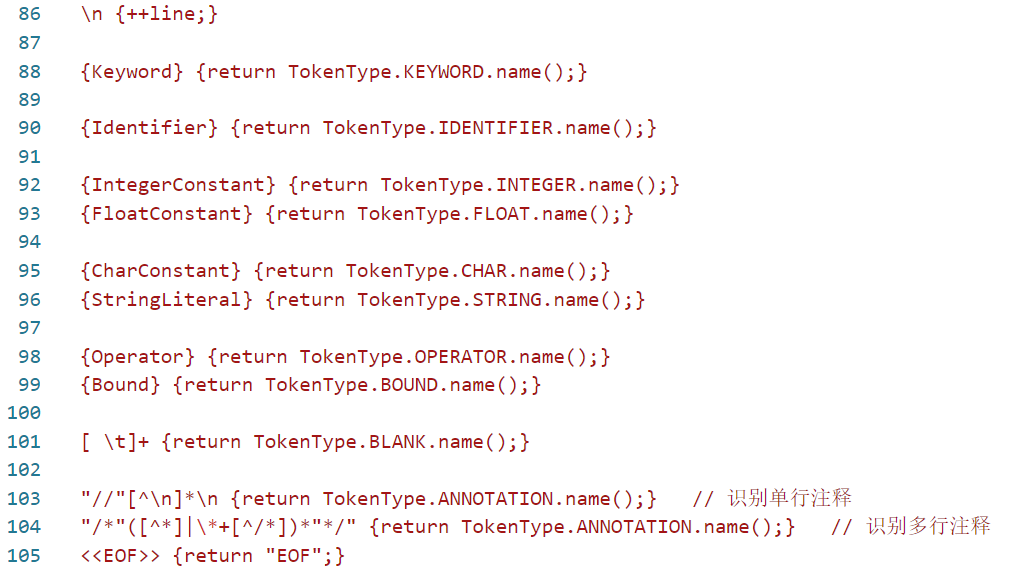


对字符串字面量，识别为双引号内的前缀加普通ASCII码字符常量、转义符常量、八进制字符常量、十六进制字符常量。正则表达式如下：



（3）词法规则

该部分包括正则表达式和扫描程序匹配到该正则表达式后执行的操作，主要为Token标签的返回，以及换行、空格与Tab、文件尾的识别，用于后续文件识别标签。该部分代码如下：



## 5.3 JFlex编译.l文件生成Java程序

用JFlex编译.l规则文件生成Java文件，命令为 JFlex myScanner.l，生成Scanner.java词法分析器。

## 5.4 嵌入BITMiniCC框架

（1）编写MyCScanner.java文件作为词法分析器入口。将上一步生成的Scanner.java文件移动到 BITMiniCC的/src/bit/minisys/minicc/scanner/ 中，编写MyCScanner.java，将词法分析器嵌入BITMiniCC框架。

MyCScanner包括三部分：重写run()方法，嵌入BITMiniCC框架；Token识别方法，识别单词，成功识别则调用Token生成函数，否则调用错误处理代码；Token生成方法，生成识别的单词的Token标签。

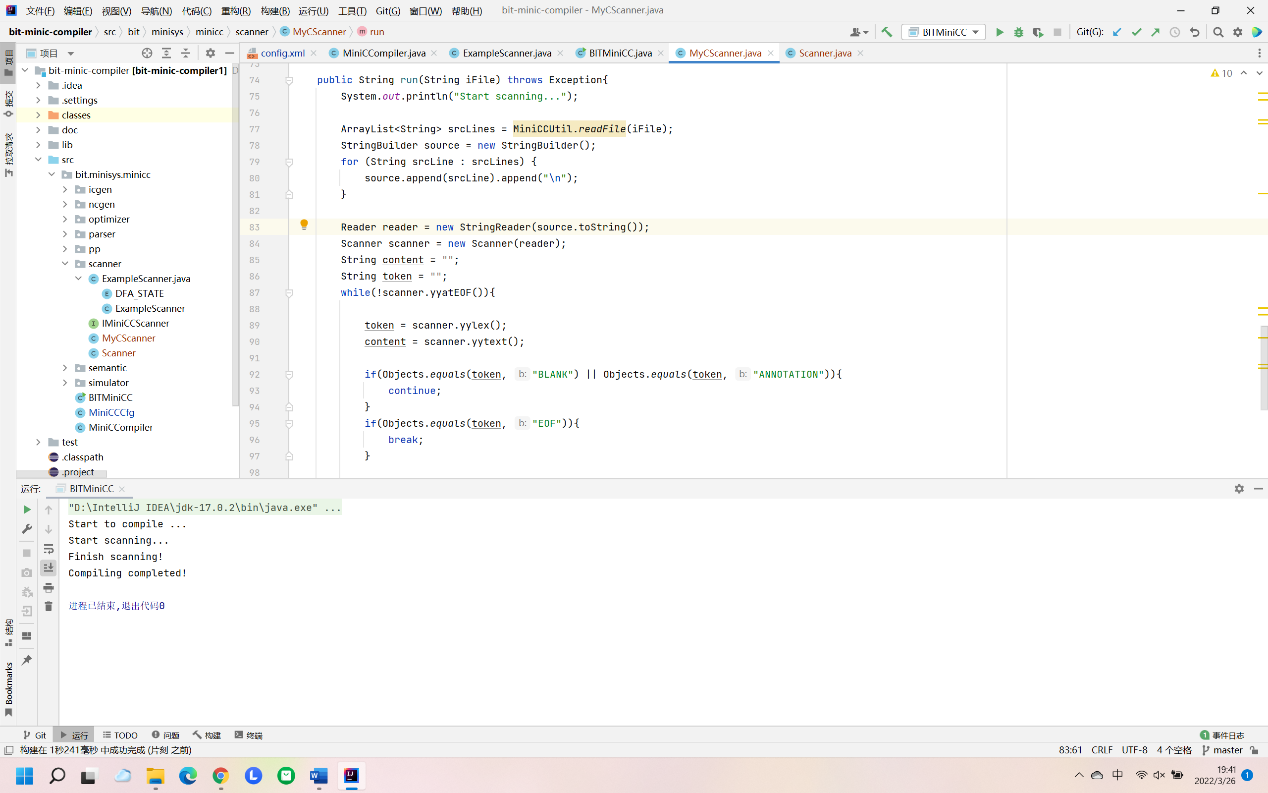
（2）修改BITMiniCC配置文件。将除词法分析外的步骤skip设为”true”，将MyCScanner。Java路径加入词法分析的path中。

（3）添加测试的.c文件路径至BITMiniCC.java文件的实参。

（4）运行BITMiniCC。

# 6. 运行效果截图

测试文件为提供的1\_scanner\_test.c文件。idea运行截图如下，可以看到词法分析器能够正常工作：



生成的文件包括1\_scanner\_test.c、1\_scanner\_test.pp.c、1\_scanner\_test.tokens文件，1\_scanner\_test.tokens文件部分内容如下图：



# 7. 实验心得体会

通过本实验，我有如下收获：

（1）对词法分析器的工作原理和具体步骤有了深入理解。词法分析器输入源程序，输出属性字流，通过确定有限自动机实现。

（2）通过阅读JFlex用户手册，对JFlex自动生成词法分析器的方法有了一定了解。JFlex能够根据用户编写的正则表达式与操作方法自动生成确定有限自动机。优点是节省时间、减少人力编写自动机的工作量、减少出错的可能，让用户能够专注于正则表达式的撰写。缺点是自动生成的代码可阅读性和可维护性较差。