北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

本科生《编译原理》课程实践报告

题　　目： 词法分析实验

学　　院： 徐特立学院

专业名称： 计算机科学与技术

姓　　名： 陈照欣-1120191086

1. 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 Flex 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。

2. 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、C\C++或者 Python 其中任何一种语言，采用程序 中心法或者数据中心法设计并实现词法分析器。词法分析器的输入为 C 语言源程 序，输出为属性字流。

   选择编码实现词法分析器，也可以选择使用 Flex 自动生成词法分析器。需要注意的是，Flex 生成的是 C 为实现语言的词法分析器，如果需要生 成 Java 为实现语言的词法分析器，可以尝试 JFlex 或者 ANTLR。

   由于框架是基于 Java 语言实现的。并且提供了相应的示例程序，建议学生使用 Java 语言在示例的基础上完成词法分析器。

3. 实验步骤

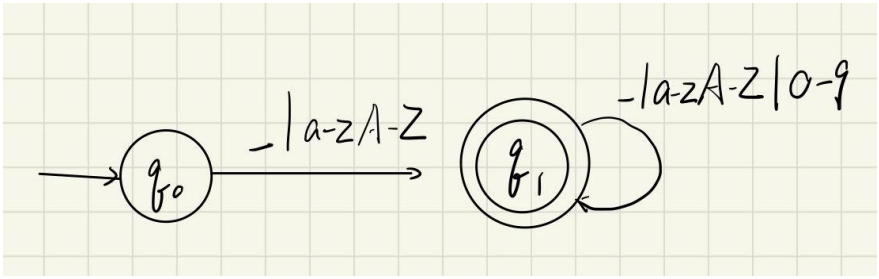
   （1） 根据 C 语言的词法规则构建 DFA

   （2） 根据 DFA 和 BITMiniCC 中提供的示例编程实现词法分析器

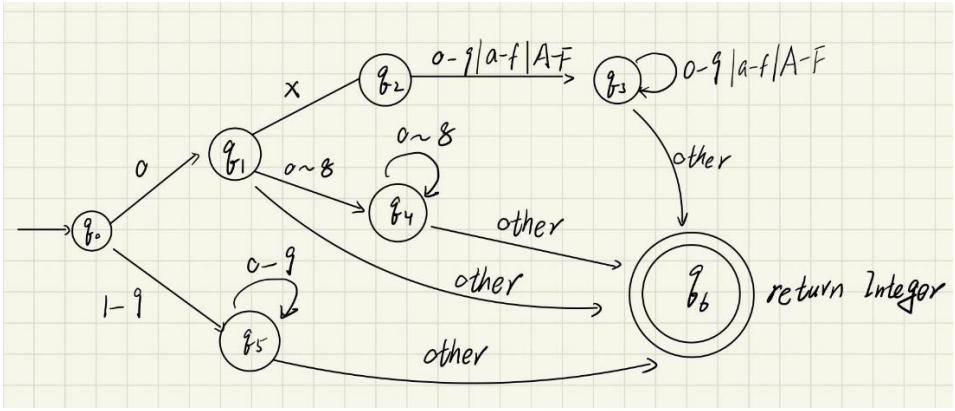
   （3） 修改 config.xml 将自己编写的词法分析器部署到 BITMiniCC 中

   （4） 进行测试

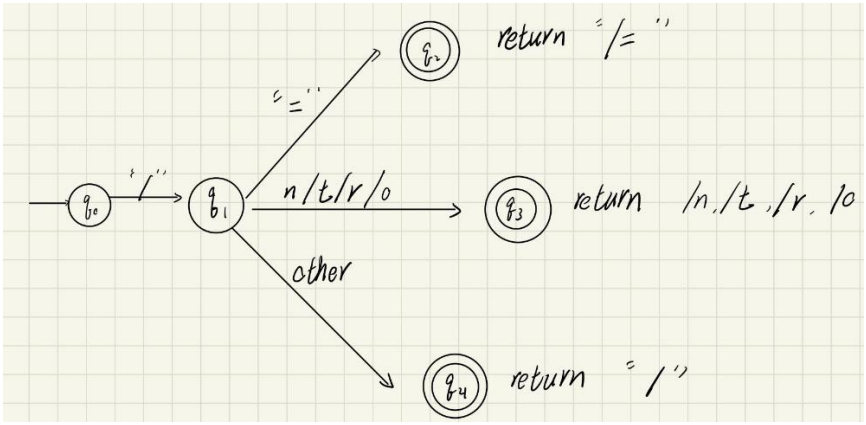
4. DFA 构造

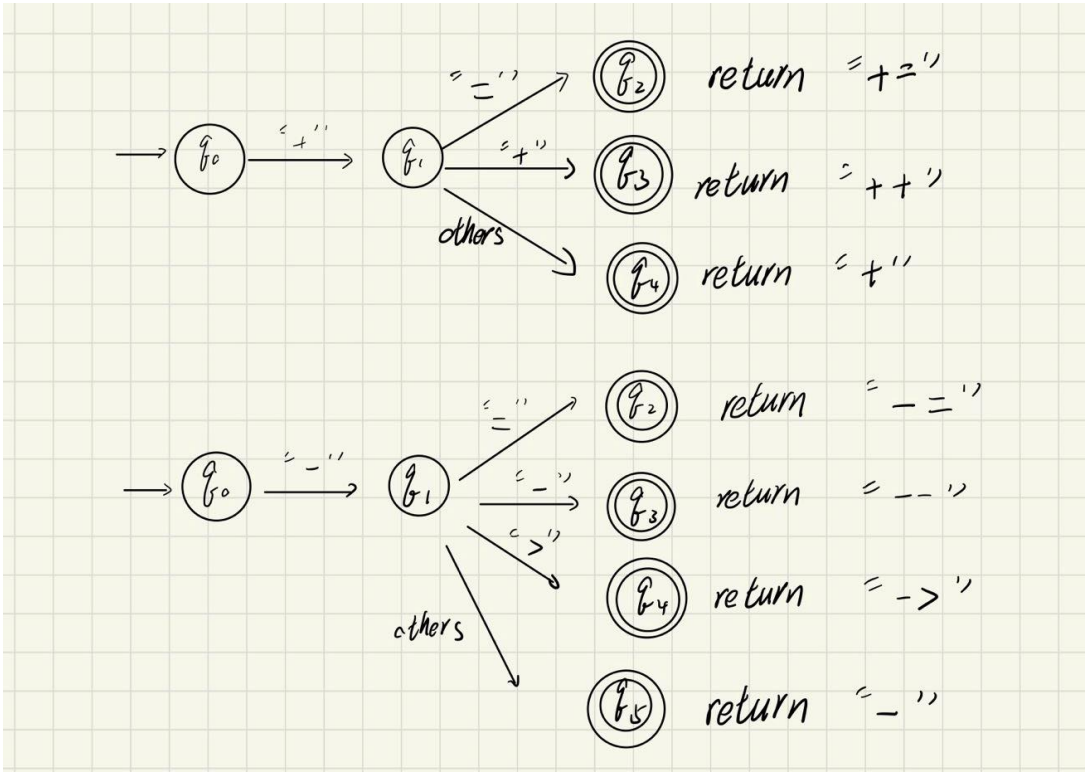


识别标识符

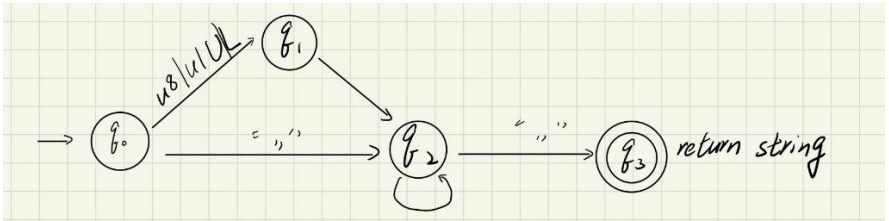


识别整型常量

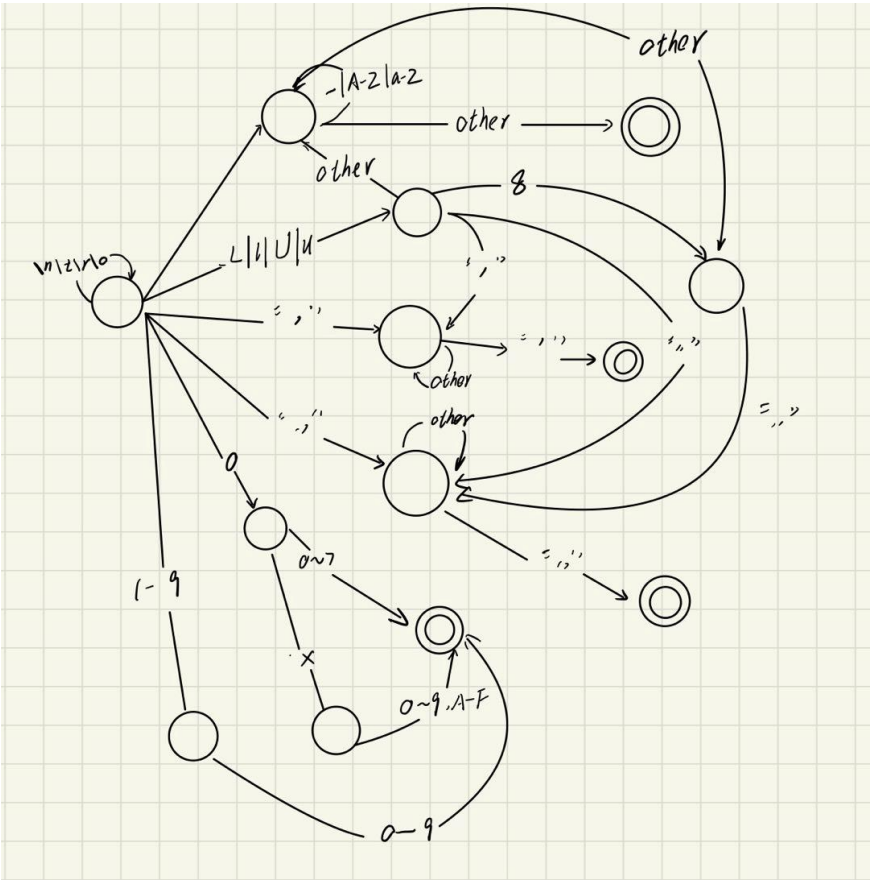




部分运算符识别



字符串识别



5. DFA 实现
   构造方式：首先根据 C 语言的词法规则对每一类单词构造一个 DFA，再采用合并初始状态的方式将 DFA 进行合并。
   代码实现：构造一个枚举，里面的元素即为 DFA 的各个状态，再使用 switch case 的结构对每一个状态进行解析。
6. 运行结果部分展示

```
[@0,1:1='+',<'+'>,1:1]
[@1,3:13='0xa.ap + 1 ',<'floating-constant'>,1:3]
[@2,14:14=';',<';'>,1:14]
[@3,1:1='-',<'-'>,2:1]
[@4,3:13='0xa.aP - 1 ',<'floating-constant'>,2:3]
[@5,14:14=';',<';'>,2:14]
[@6,1:1='+',<'+'>,3:1]
[@7,3:11='1.1e + 1 ',<'floating-constant'>,3:3]
[@8,12:12=';',<';'>,3:12]
[@9,1:1='-',<'-'>,4:1]
[@10,3:11='1.1E - 1 ',<'floating-constant'>,4:3]
[@11,12:12=';',<';'>,4:12]
[@12,1:6='1.1e1f',<'floating-constant'>,5:1]
[@12,1:7='1.1e1f ',<'floating-constant'>,5:1]
[@13,8:8=';',<';'>,5:8]
[@14,1:6='1.1E1L',<'floating-constant'>,6:1]
[@14,1:7='1.1E1L ',<'floating-constant'>,6:1]
[@15,8:8=';',<';'>,6:8]
[@16,1:3='0.0',<'floating-constant'>,7:1]
[@17,5:5=';',<';'>,7:5]
```

7. 实验心得体会

本次实验使我对 C 语言的单词类型及其特点有了更深入的了解。比如关键字、标识符、各种常量的特点和规则。

通过在实验过程中采取 DFA 的分析方法，我对于编译器模块中词法分析的实现有了更深刻的认识。实现词法分析的过程类似于枚举的过程，关键在于画出的 DFA 是当前状态的，而读入的字符其实是“下一个”状态的，也就是说，除了分隔符等某些符号之外，对于一个字符串的结束要根据最后一个字符的下一个字符来判断。这与 DFA 是不同的，也体现出了理论与实际实现的区别。

本实验中我遇到的困难主要在于，预处理文件除了会去除注释和一些空行之外，还会给运算符增加一些空格，例如将“>>”处理为”> >”，将浮点数“1.1e+1”处理为“1.1 e + 1”，这样会给自动机的判定带来一些困难，需要增加一些额外的状态作为过渡状态。