本组成员：

王涛 2016302580147

何万伟 2016302580125

段小式 2016302580005

段梦梦 2016302580001

使用LLVM的工具来实现一门编程语言，从源代码开始进行词法分析来分成一个个token，再进行语法分析形成AST，进行代码生成LLVM IR，然后代码优化，之后形成平台无关代码有向无环图，最后生成机器码指令，以上即是编译的整个流程。

在LLVM源码的目录下，有一个目录叫做examples，这个目录下边有几个LLVM的例子，文档说的一些没能让人明白的东西，例子就讲的很清楚了。

Kaleidoscope展示了如何在LLVM之上构建一个支持一门自己定义的编程语言的编译器。这不但有利于理解编译器的工作流程，也非常有利于理解LLVM，因为它配有详细的文档说明，并且贯穿了编译器的整个流程。

第一部分的词法分析器相关，让我们对Kaleidoscope语言及其词法分析器有了一定的了解，直接采用C++实现功能，使我们对词法分析有了一个更清晰而直观的了解。

首先，Kaleidoscope只支持64位浮点数这一种数据类型，并且可以在使用“extern”对函数进行申明的情况下调用标准库函数，教程中给出的两个例子也比较清楚的让我们理解了这两个特点。

使用词法分析器可以讲输入分割成一个个语元（token），第一章则大概讲述了怎样使用语元来表示每个词，并用函数读取每个词对其加以判断和分类。词法分析器返回的语元会带有一个编号，并有可能带有数据，像所给例中使用枚举方法定义了5个语元种类，eof; def; extern; identifier; number，对每个词进行循环读取之后再进行分类。分类是使用c++中isdigit，isalpha之类的函数。对于词法分析器返回的语元，除了上述的几种枚举值外，还有一些字符，对于这种情况，词法分析器则会返回这些字符的ASCII值，比如“+”返回ascii码，定义ID，以后根据ID来确定是什么运算。数值部分的处理则比较复杂，例子中是先存入NumVal中，若为标识符，则名称先被存入IdentifierStr。

Kaleidoscope的词法分析器由gettok函数实现，调用这个函数可以得到标准输入中的下一个语元，通过getchar()函数从标准输入中一个个读取字符，在识别读取的字符的过程中，将最后读入的字符存到LastChar，放置到后续处理。该函数利用循环来删除了语元之间的空白符，然后再识别标识符和“def”、“extern”等关键字。

词法分析的内容并不是十分复杂，主要在于枚举出几种种类来对语元进行分类，在使用过程中，可以优先选择常用的种类先写，对于更多的少用的关键字和运算符可以再慢慢完善补充。