编译原理实验：Lab1

指导老师：徐辉
助教：陈澄钧 崔漠寒

2021 年秋季学期

实验环境说明

本次实验使用环境为 Linux 系统，推荐使用版本为 Ubuntu18.04，在 Windows 或其他系统的情况下可以选择安装虚拟机或是使用 docker 容器安装对应版本的 Ubuntu。

安装完毕后，使用一下指令进行环境搭建：

```
sudo apt-get update
sudo apt-get llvm clang-9
```

这两条命令将快速便捷地安装实验所需要的 llvm 库以及 clang。安装完毕后直接用实验测试部分提供的编译指令，看是否能正常执行，正常执行则表示安装正确。

其中,clang 和 llvm 可以简单理解为一个完整编译器的前端和后端,联合起来可以编译 C/C++ 的程序生成可执行文件。不过 llvm 实际上的功能远不止于此，它可以作为一个通用的编译后端优化器，感兴趣的同学可以自行调研与学习。本课程后面的 lab 也会用到 llvm 的 api 实现后端部分来帮助生成可执行文件。

1 实验介绍

Lab1 的目标是实现一个简单的词法分析器，能够在指定的字符输入流中，识别出特定的字符串，从而来供编译器后续阶段使用。

1.1 词法分析

词法分析是将字符序列转换成单词 (Token) 的过程，实现该功能的程序或者函数就称为词法分析器 (Lexical analyzer, 简称 Lexer)，也可以叫做扫描器 (Scanner)。词法分析器一般以函数的形式存在，供语法分析器调用，其实现通常基于有限状态自动机。

在本实验中，词法分析由函数 *gettok()* 实现。首先，程序的主函数部分会打开并读取指定的待分析文件，得到字符序列输入流，之后不断调用 *gettok()* 函数进行分析识别，直至输入流结束。

本实验提供的框架代码将需要识别的特定字符串类型定义在 *Token* 枚举类中, 同时，在整个编译器的实现中，为了避免过于复杂的类型检查，我们规定了数值类型只能是 int 或者 double，同时也不考虑负数的情况 (因为负数其实是个一元运算)。类型定义在 *Types* 的枚举类型中。

单词类型	说明	示例
define	用于定义的关键字，c 语言中定义基本都以类型属性开始，因此此处也可理解为类型关键字	int, double
extern	用于声明外部函数的关键字	extern
identifier	标识符，包括函数名表示符和变量标识符	bar,foo,temp
int	int 类型常量	12, 0, 100
double	double 类型常量	8.1, 0.98, 1.0
EOF	文件结束符，表示文件已经扫描完毕	-
Others	其他字符，直接返回其 ASCII 码对应的值	)(;+-*

表 1: 识别单词类型列表

$$
\begin{aligned}
&< ident > && ::= && [A-Z\_a-z][0-9A-Z\_a-z]^* \\
&< intconst > && ::= && [0-9][0-9]^* \\
&< doubleconst > && ::= && < intconst > . < intconst > \\
&< type > && ::= && int|double
\end{aligned}
$$

表 2: 语法产生式

1.2 识别列表

本实验所需要识别出来的特定单词如表 1所示，因为本次整体的 compiler 实现偏向于 c 语言风格，因此识别目标也倾向 c 语言中的关键字。

1.3 语法产生式

由于本实验只进行语法分析，因此关注的语法产生式只有词法分析部分的语法产生式。其内容如表 2所示：

其中，整数的开头是可以为 0 的，也就是诸如 “0032” 这样的常数是合法的。而在常量中，我们默认区分浮点数和整数的关键就是在于是否有小数点。对于 identifiers，我们默认必须是以大小写字母开头，后面只能接大小写字母或者数字。

同时，Lexer 在扫描识别出指定 token 之后，对于一部分 token 要储存其代表的值以供后续语法分析使用，在本 lab 中一共有这么几个需要储存值的请况：

1. 识别情况为 identifier 时，将识别的变量名储存在变量 $IdentifierStr$ 中。
2. 识别情况为 int 时，将识别的 int 值储存在变量 $NumValI$ 中。
3. 识别情况为 double 时，将识别的 double 值储存在变量 $NumValD$ 中。
4. 识别情况为 def 时，将识别的类型储存在变量 $ValType$ 中。

框架的指定输出格式已经规定了这些值的输出，实验中只需在识别到对应的类型时进行正确的储存操作即可。

2 实验测试

lab1 实验的测试用例为压缩包中的 *input_example.data* 以及 *output_example.data*, 在配置好环境后，在当前目录的命令行下输入如下指令，即可输出测试结果。

```
clang++ lab1.cpp -o lab1
./lab1 input_file_name
```

其中，*input_file_name* 就是所要进行词法分析的目标文件。测试的输入输出框架都已提供好，不需要修改，在没有做任何代码填充的情况下，用一个空文件来进行测试，由于空文件只有一个文件结束符，因此输出的结果为-1(框架里面指定，其余返回值也可以在框架中看到)。

3 实验提交

实验完成之后，将实验代码在截止日期之前上传至 elearning，助教将根据代码完成质量以及测试用例通过情况进行打分。

提交时将代码压缩到以学号命名的压缩包中 (zip 格式)，提交压缩包文件。代码的文件名保持原样，为 lab1.c。