编译原理实验1实验报告

功能及实现方式

词法分析器功能及实现

- **功能1**：识别c--语言中的关键字、标识符、整数、浮点数、运算符、界符等。**实现**如下
  - 通过flex提供的正则表达式机制，实现词法分析
  - **额外内容**：实现了指数形式的浮点数识别
  - **精妙的实现**：使用自己创建好的`ast模块`和`tree模块`（ast.h/.c，tree.h/.c），在词法分析的同时创建ast节点并标识好了节点信息，方便后续语法分析
- **功能2**：实现error type A的报错机制，即识别到非法字符时，报错并跳过，不会影响后续识别。**实现**如下：
  - 实现`log模块`，用于输出错误信息，同时具有较高的可扩展性
  - 在`. { /*log function*/ }`中实现了报错机制

辅助模块的介绍：

- 本实验中实现了三个辅助模块，提供了良好的抽象以及接口，使得词法分析器和语法分析器的实现更加简洁、清晰
  - `tree模块(tree.h/.c)`，提供了树这一数据结构的接口：初始化节点、添加子节点、遍历节点等
  - `ast模块(ast.h/.c)`，提供了ast的具体实现：ast节点类型、值、行号等信息。同时提供了ast的接口：新建ast节点、设置为ast根节点、添加ast子节点、ast节点是否为叶子节点、遍历ast输出语法树等
  - `log模块(log.h/.c)`，提供了log的接口：输出错误信息、输出调试信息等

语法分析器功能及实现

- **功能1**：根据词法分析的结果构建语法树，并标记各个终结符的行号。**实现**如下：
  - 使用`tree模块`和`ast模块`，在语法分析的同时构建语法树
  - 将产生式体第一个变量的行号传递给产生式头部的变量，方便构建语法树时标记行号
- **功能2**：实现了error type B的报错机制，能够报出程序中的语法错误，并成功进行错误恢复进行后续的语法分析，**实现**如下：
  - 在bison源文件syntax.y中添加error处理机制，不断pop直到遇到错误恢复的同步符号
  - 在我们的实现中同步符号有如下内容：RP、RC、SEMI，具体内容参见syntax.y文件error的上下文
- **功能3**：实现了语法树的存储与输出，并为之后，**实现**如下：
  - 使用`ast模块`和`tree模块`提供的接口`ast_walk(print_ast_node)`，遍历语法树，输出语法树的结构

编译相关

环境要求

- ubuntu 20.04
- flex 2.6.4
- bison 3.5.1

编译方法

```
cd Code
make parser
```

最后生成的可执行文件为`parser`，可以通过`./parser /path/to/file`运行