# 编译原理实验报告

## 实验一

设计方案  
功能模块结构图  
主程序流程图  
主要子程序的流程图（若有必要） 主要数据结构：符号表、TOKEN串表等。  
开发平台（操作系统、设计语言）  
具体实现（包括主控程序、各个功能模块的具体实现，给出主要函数部分 即可，不用粘贴全部代码）。  
经验总结

### 设计方案

#### 功能模块结构图

#### 主体结构流程图

对于词法分析器，主要功能是读入源代码字符串流，输出 Token 流和符号表。其中 lexer 函数是主要程序，其结构为

对于其调用的函数，依赖图为：

#### 主要数据结构

* 符号表，在扫描的同时生成并维护，示例代码的符号表如下

{'addr': '0', 'value': 'a', 'type': 'id', 'line': 1, 'cur': 7}  
{'addr': '1', 'value': 'b', 'type': 'id', 'line': 2, 'cur': 7}  
{'addr': '2', 'value': 'c', 'type': 'id', 'line': 3, 'cur': 7}  
{'addr': '3', 'value': '2', 'type': 'digits', 'line': 4, 'cur': 7}  
{'addr': '4', 'value': '1', 'type': 'digits', 'line': 5, 'cur': 7}

* Token串，边识别边维护，示例代码部分 Token 串如下，其中保留行和光标位置用于报错和警告提示

["Keyword", "int", {"line": 1, "cur": 5}]  
["id", "0", {"line": 1, "cur": 7}]  
["OP", ";", {"line": 2, "cur": 1}]  
["Keyword", "int", {"line": 2, "cur": 5}]  
["id", "1", {"line": 2, "cur": 7}]  
["OP", ";", {"line": 3, "cur": 1}]  
["Keyword", "int", {"line": 3, "cur": 5}]  
["id", "2", {"line": 3, "cur": 7}]  
["OP", ";", {"line": 4, "cur": 1}]  
["id", "0", {"line": 4, "cur": 3}]  
["OP", "=", {"line": 4, "cur": 5}]  
["digits", "3", {"line": 4, "cur": 7}]  
["OP", ";", {"line": 5, "cur": 1}]  
["id", "1", {"line": 5, "cur": 3}]

#### 开发平台

* Python 3.9.10 on Darwin arm64

Python 3.9.8 (main, Nov 10 2021, 03:48:35)  
[Clang 13.0.0 (clang-1300.0.29.3)] on darwin for arm

* 额外依赖包：无

### 具体实现

对于词法分析器首先需要扫描源文件到缓冲区，可以使用

def read(\*\*kwargs):  
 content = ""  
 if "mode" in kwargs and kwargs["mode"] == 'file':  
 if "filepath" in kwargs and kwargs["filepath"] is None:  
 print("Error:", "Empty File Path.")  
 return False  
 try:  
 f = open(kwargs["filepath"])  
 except:  
 print("Error:", "Open file failed, please confirm your filepath is right.")  
 return False  
  
 content = f.read()  
 else:  
 print("Please input your code HERE ,press 'EOF' and ENTER to end your input!")  
 print(">>> ", end="")  
 tmp = ""  
 while tmp != "EOF":  
 tmp = input()  
 print(">>> ", end="")  
 if tmp != "EOF":  
 content += tmp+"\n"  
 print("")  
 return content