**说明文档**

**一、概述**

**Requirements**：ply

**源语言：**C语言

**目标语言**：python

**测试实现：**整数排序程序，回文字符串检测程序翻译

**二、文件和运行说明**

1. 文件说明

pre\_post\_process--预处理

lex--词法分析

yacc--语法分析

AST--语法分析树

translate--语义分析、代码生成

C\_utils--C函数预定义

C2P--入口文件

2. 进入主目录，执行

python C2P.py sourcefile.c

或python C2P.py sourcefile.c outputfile.py

**三、具体实现**

**1. lex实现思路**

使用ply中的lex工具进行分析，分析工具需要手动添加tokens和rules；其中每个token和一条rule对应，表示一类字符串。

tokens除了默认的newline, ignore, comment,error等token之外，分为：常量、变量、运算符号、关键字等几类，在lex.py中分别实现，实现方法使用正则表达式。

**难点**：主要是如何将所有word进行合理的分类，方便之后进一步处理。

词法分析的实现思路参考了：

<https://www.jianshu.com/p/0eaeba15ee68>

**2. yacc分析思路**

yacc实现时使用了ply中的yacc工具，实现思路主要是根据代码的结构构造一颗语法分析树，具体通过自动机实现；语法大致可分为代码、语句、表达式三个层面，结构图如下：



**难点**：主要是理清代码实现思路，保证语法能够包含到测试程序中所有可能的部分，同时不出现漏洞，如无法推导或死循环等情况(自动机中unreachable等情况)。

语法分析参考了:

<https://lotabout.me/2016/write-a-C-interpreter-4/>, <https://github.com/wish142857/iTranslator-C2P>

**3.语义分析和中间代码生成**

translate.py文件中定义了Translator类，其中定义一个成员变量字典，该字典中的键为语法分析数中各节点的key的token类型，每个键对应的值为处理该键的类型的token的处理函数，translate(self,input\_filename,output\_filename)为最外层的入口函数，逐级调用到code\_generrator()函数内部逐个判断当前树节点的key值的token类型，并调用Tranlator类中处理函数字典的对应函数。

**创新点：**

处理了C语言中的字符串结束符’\0’,将其成功转成了python中的None类型。

**四、小组分工**

阮崇智和关孟潇负责开发部分以及部分测试代码，张思旭负责部分测试代码，文雨负责编写代码文档。