语法分析阶段设计文档

输入输出要求摘录

- 输入：`testfile.txt`
- 输出：`output.txt`
- 把读入的词法内容，分析出程序中包含的语法部分（参照给定语法），并进行输出（不包括错误处理）

程序环境

- 语言：C++
- IDE：VS2019
- 测试环境：VS自带编译器、C++ 14
- 目标环境：Clang 8.0.1、C++ 11

程序设计

编码前设计

1. 设计基类（`Base`），包含分析语法的方法(`read_in()`)、输出成字符串的方法(`to_string()`)以及存储该语法包含所有基类的属性（`vector<Base>`）。
2. 每个语法成分公有继承基类，重写上述两个方法，存储自己包含的内容到vector。

存在问题

我们并不需要输出所有语法成分，意味着部分语法成分不需要封装而可以沿用`WordInfo`的封装。因而统一设计基类会比较麻烦。

编码后设计

- 对原有`Lexer`进行增量改造，便于进行语法分析：
  - `int word_pos`

    初始化为0，用于充当读取被存储的`vector<WordInfo>`的指针，便于语法分析时依次取用读取到的词法成分。
  - `WordInfo get_next()`

    返回位于`word_pos`处的词法成分，并把`word_pos`更新为`word_pos+1`
  - `WordInfo peek_next()`

    返回位于`word_pos`处的词法成分
  - `void set_pos(int pos)`

    设置`word_pos`为传入值
  - `int get_pos()`

    获取当前`word_pos`值
- 设置用于记录程序中包含的函数的表格`FunctionTable`，暂时只记录函数名称对应的返回值类型
  - `map<string, int> func_to_type`

记录函数名称-返回值类型的键值对。

0：`void`

1：`int`

2：`char`

- `void add(WordInfo w, int type)`

  功能：用于登记新读入到的函数

  输入：`WordInfo`，要求其类型为`IDENTF`，即函数名称；`type`，函数类型，如上。

- `bool have_return(WordInfo w)`

  功能：返回指定函数是否有返回值

  输入：`WordInfo`，要求其类型为`IDENTF`，即函数名称。

- 对于每个需要输出的语法成分，单独设计一个`class`来封装，其属性因其需要存储的词法成分而不同，但都有读入（分析）和输出成字符串两个方法。
  - `int read_in(Lexer& lexer)`

    功能：传入已读取到的词法成分开始分析

    输入：读取完成的词法成分

    返回：正确得到语法成分返回-1；错误时返回出现错误处的词法成分的位置

  - `string to_string()`

    功能、返回值：把语法成分变成字符串格式以供输出

  - `int word_pos`

    记录该成分保存的（或读取到的）第一个词法成分在整个读入的程序的词法程序中的位置（全局排序）。这个数值可以用于预读入之后的回溯，也可以用于之后的错误分析。

- 采用自顶向下的递归语法分析方法