# 说明文档

## 编译信息

- 测试平台：Linux(ArchLinux)、Windows
- 编程语言：C++
- 编译器：gcc
- 编译器版本：6.1.12
- 注：

  1、src文件中包括Linux和Windows两个版本的两个可执行文件。

  2、在Linux与Windows都是用CLion开发。

## 程序设计结构

- 设计结构模块化，一个模块实现一个功能。充分运用了面向对象的特性，所有功能均封装为对象。其中，具体每个模块的功能见下方。

## 各模块的功能说明

- *calculator.cpp*、*calculator.hpp*:

  计算器类：用以计算各个表达式的值。实现时使用了调度场算法，能够自动处理各类运算符的优先级，并且自动判断。

- *grammarAnalyst.cpp*、*grammarAnalyst.hpp*:

  语法分析器：进行顺序结构、分支结构、循环结构的相应处理。

- *iteratorManager.cpp*、*iteratorManager.hpp*:

  迭代器托管器：负责托管 token 流，可以在需要 token 的地方获得下一个 token 、跳过给定数量的 tokens 或者取得任意位置的 token 。同时也在程序运行时，将结果进行输出。

- *main.cpp*:

  主函数：声明了所需要的对象、依次调用了所有的方法。

- *memoryStack.cpp*、*memoryStack.hpp*:

  内存栈：模拟编译器在进行编译的时候，对变量进行的管理，以变量树的形式来管理所有的变量。

- *token.hpp*:

Token 类：即一般编译器在运行时，会将代码处理成的形式。

- *util.cpp*、*uril.hpp*:

  工具类：输出管理工具

- *wordAnalyst.cpp*、*wordAnalyst.hpp*:

  词法分析器：读取 input.txt 文件中的c语言代码，并进行词法分析，形成 token 流。每读入一个字符进行相应判断，按照关键字、标识符、数字、运算符、注释（单行注释、多行注释、文档注释）等等并生成相应的 token 流

# 实现的功能

## 顺序结构

1. 声明语句：
   - int a;
2. 初始化语句
   - int a = 3;
3. 赋值语句
   - a = 4;
   - a = a + b;
4. 输出语句
   - printf("xxx is %d\n",变量名或表达式);
5. 空语句 ;
6. a++, a--, --a, ++a
7. a = b = c = d;

## 分支结构

1. //举例 1

   if(表达式 1) {

   语句 1;

   语句 2;

   }

   else if(表达式 2){

   语句 3;

   语句 4;

   }

   else {

语句 5;

}

2. //举例 2

   if(表达式 1)

   语句 1;

   else

   语句 2;

### 循环结构

1. for

   for(表达式 1; 表达式 2; 表达式 3) { 语句 1; 语句 2; ... }

   其中的三个语句均可以为空。 可以break跳出循环。

2. while

   while (表达式 1){ 语句 1; 语句 2;

   ... }

   可以break跳出循环。

3. do-while

   do{ 语句 1; 语句 2; ... } while (表达式 1)

   可以break跳出循环。

## 特色

- iterator中使用了**托管模式**，管理所有iterator实例，保证了在获得token的时候
- 将各个模块进行了封装，减小了耦合，提高了程序的健壮性和安全性。
- **词法分析器可以识别除比赛要求之外的类型，比如浮点型，位运算、宏定义等等。**
- 在表达式的计算方面使用了调度场算法，将中缀表达式转换成后缀表达式。

## 补充说明

- 源文件中对于某个类或函数说明的注释大都写在头文件(.hpp)中, cpp文件中也有注释。
- 感谢裁判的耐心解答和种子杯宣讲会。宣讲会中的内容让我们受益匪浅，给了我们很大启发。对我们的思路梳理、代码编写、以及逻辑架构有很大的影响。