文法分析

一：词法分析：

用枚举 Type 来定义 calculator.c 中出现的所有词法单元的类型，共 13 种，从文件中读到的所有有意义字符/字符串都会被分成如下几类

```
enum Type {
    INT,    // 整数 0
    FLOAT,      // 浮点数 1
    SIGN,   // 运算符 2
    X,      // 自定义变量 3
    DEF_I,      // int a 的 int 4
    DEF_F,      // float a 的 float 5
    CALL,       // 调用函数 6
    ASSIGN,     // = 7
    END,        // . 8
    SEM,    // ; 9
    KEY,        // 关键字 10
    CONS,       // 常量 11
    COMA        // , 12
};
```

而文件中的语句会被分割成 token 的形式,token 中仅包含名字和类型(type),其定义为

```
typedef struct _Token {
    char name[MAX_VAR_NAME];
    int type = -1;
} Token;
```

对于变量，name 中存储的为其名字，对于数字，name 存储的为其字符串形式

合法的变量名为：([a-z]|[A-Z])+(_|[a-z]|[A-Z]|[0-9])

合法数字形式为：([0-9]*)(\.[0-9]+)?

二：语法分析：

(1) <程序> -> (<语句><结束符号>(<单行注释>)*)*(<语句><全文结束符>)

支持单行注释(//)，但是全文结束符”.”后面不能有任何文本包括注释

(2) <语句> -> <声明语句>|<赋值语句>|<输出语句>

<声明语句> -> <声明类型> <变量名>(,<变量名>)*<结束符>

<赋值语句> -> <变量名> <赋值符号> <表达式><结束符>

<输出语句> -> write(<变量名>)<全文结束符>

其中声明类型只有 int 和 float 两种

<表达式> -> <表达式> + <表达式>|<表达式> - <表达式>|<表达式> * <表达式>|<表达式> / <表达式>|(<表达式>)|(-<表达式>)

支持负数，例如-1 * 2，2 * (-1)，(-1 * 2)等形式

其余形式均为非法形式

三，特性

支持 int a,b;但不支持 int a = 1;

目前只接受十进制

由于采用每行处理一次 tokens 的方法， 如果有多行函数， 将 tokens 放在一起处理的话，就会无法精确判断第几行错了

支持负数

运算部分采用逆波兰表达式，没有采用语法树，因此可扩展性不高

由于 C 语言函数需要指定返回类型且只能返回一个参数， 所以实际运算时采用 float 形式，由调用者自己强制类型转换

仅使用了 stdio.h 和 stdlib.h 两个标准函数库

重写了 strcmp， strlen， strcpy， ctrncpy， 以宏的形式重写了 assert， 如果不使用宏会无法报错出第几行

支持+ - * / ( )

浮点数可以转换成整数，整数不可以转换成浮点数

每个语句需要以;结束，整个程序以.结束，**.后面不能有任何字符**

变量需先声明，且未赋值前不能参与表达式

报错统一用 error 函数来进行，报出错误所在的行

四，大体流程

除了部分用于判断的小函数之外，函数由几个主体函数组成

每行读入一个语句，以;为分隔将一行的语句划分成不同的 token，组成 tokens

Into_Tokens

分析 tokens， 看是否符合标准形式， 同时将声明语句中声明的变量注册到符号表中

Valid_Tokens

计算赋值语句中的表达式，将结果更新到符号表中变量的值(包括直接的赋值语句和表达式形式的赋值语句)，然后实现 write 的输出

Run_Tokens

其中表达式的计算调用了 calculate，expression_to_ans，rpn_to_float，其作用分别为：处理返回值的类型，将表达式转化为逆波兰表达式(后缀表达式)，将后缀表达式计算出结果