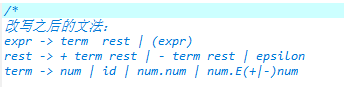
实验一 表达式的分析

1. 实验目的

通过构造一个从中缀表达式到后缀形式的表达式翻译器。

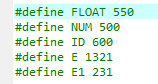
1. 实现简单的词法分析。
2. 初步了解递归下降语法分析原理。
3. 初步了解语法制导翻译的过程。
4. 实现技术

文法：



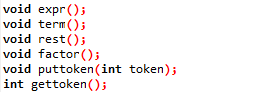
1. Token

定义了四个token。

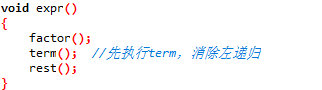


其中 FLOAT为浮点数，NUM为数字型，ID为字符型、E为带”E”的浮点数，E1为带”E-”的浮点数。

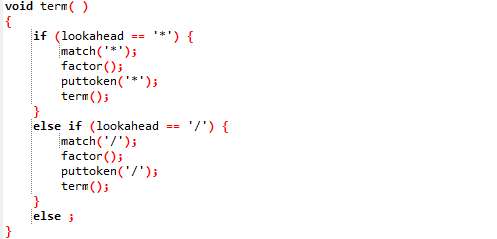
1. 函数



1. expr为求表达式函数。

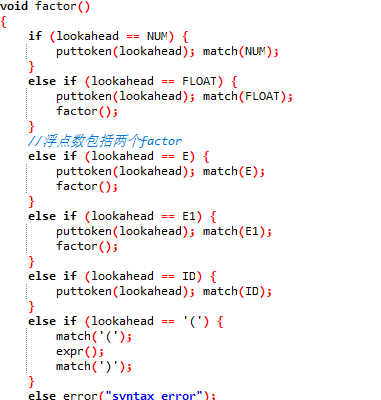


1. term为乘除语法函数



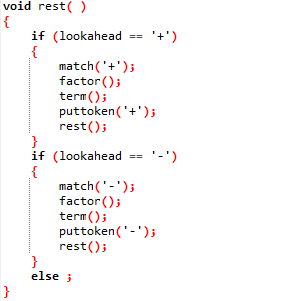
这里如果检测到“\*“，”、“，则match一下之后输出下一个factor，由于”\*“，“、”的优先级很高，所以不需要递归可以在factor之后直接puttoken。

1. factor函数



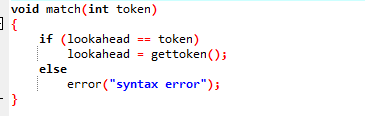
这里检测到token之后，可以直接输出。如果不是浮点数可以寻找下一个符号，否则需要递归寻找浮点数的小数部分和指数部分。

1. rest函数（处理“+”，“-”）



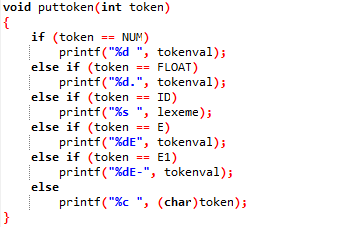
这里如果有+-,需要递归执行term。如果term函数没有执行说明后面没有乘除运算，可以直接计算。否则说明后面的是\*或/运算。并结束之后递归执行rest。

1. match



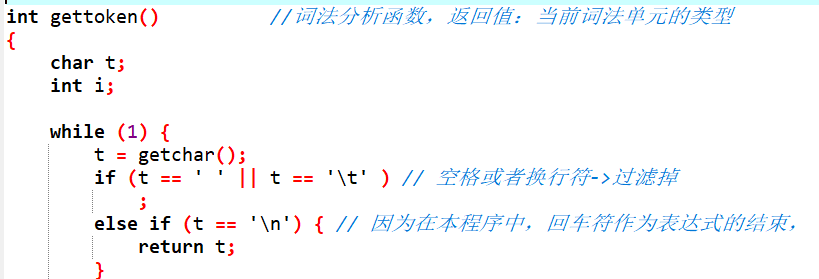
匹配token，如果正确则查找下一个token。否则报错。

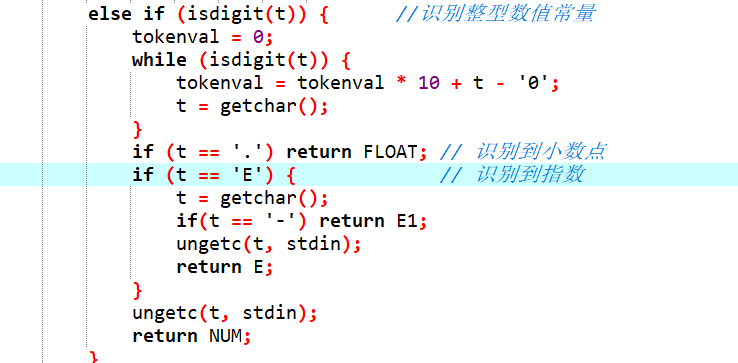
1. puttoken

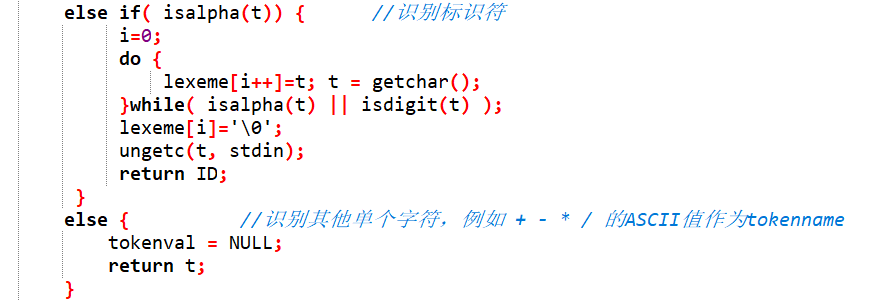


Puttoken直接输出变量factor，其中浮点数和指数形式需要输出里面的符号如小数点和“E”。

1. gettoken词法分析

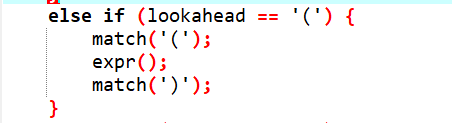






该词法分析函数可以识别数字、字符、符号以及其他的需要被过滤的符号。其中在数字中还增加了识别小数和指数的部分。

1. 括号匹配

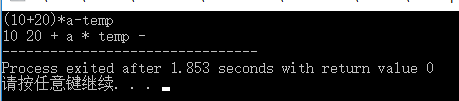


括号匹配只需要将括号后面的部分当作一个expr表达式，执行一遍expr函数并匹配到下一个“)“就行，表示括号内的运算优先执行。

1. 功能及运行情况
2. 识别整形常量。

输入样例：(10+20)\*a-temp

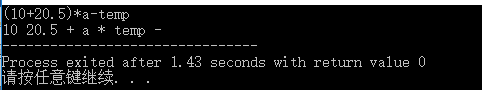
输出：



1. 识别整形常量、小数形式浮点数。

输入样例：(10+20.5)\*a-temp

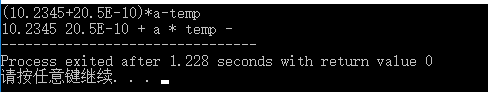
输出：



1. 识别整型常量、小数形式浮点数、指数形式浮点数。

输入样例：(10.2345+20.5E-10)\*a-temp

输出：



1. 解决的问题：
2. 识别括号
3. 识别浮点数以及指数形式
4. 消除左递归
5. 能够过滤空白、制表符、换行符。