手工构造词法分析器

1. Flex工具

解题思路：

首先，讲讲我们的解决这个问题的过程。课堂上所讲内容已经可以完成对简单c程序的词法分析，但是我们发现其不能完成对数字0的词法分析.在对小数的词法分析中，我们发现其只能实现整数位不为0以及小数位第一位不为0的小数decimal。因此我们加入了对0的定义ZERO。在规则中加入对0的分析，同时改进decimal的定义将0加入decimal，使得任意小数都可以在规则中匹配到。

接着我们增加了对注释行的定义以及float型数据的定义。在原有的标识符identify的基础上，定义了一个sentence句子，最终通过sentence定义了对注释的解析(/)(\*){sentence}(\*)(/);对于float型数据，其格式为:小数+e整数，因此定义FLOAT {DECIMAL}(e){INTEGER}。

在以上步骤完成后，我们完美解决了前两个测试数据的输出，但是考虑到第三个测试数据如果存在错误，则只需要输出错误的信息。因此我们抛弃了直接输出的想法，选择用字符数组存放每个单词的分析，并用一个结构体详细地将单词所在行、单词内容、单词种类储存下来。并且新添加了一个ERRO的定义：{DECIMAL}(e)，匹配到ERRO单词，则将单词分析信息存放在erro字符数组中；匹配到其他单词，将单词信息存放在结构体数组中。最后在main函数中，判断erro是否为空，若erro非空，说明有错，只输出错误信息；若erro为空，所有单词正确，将结构体数组信息输出。

1. C++程序

设计思路：

用(c=getchar())!=EOF来判断所输入的字符流是否已经结束输入，只有当输入以ctrl + z 结尾时才算结束；在词法生成之前，整个输入的内容必须先从check 函数中过一遍，最后函数返回的数保存在一个flag中，该值影响程序是否进行下一步的词法分析输出

Check函数的设计要点在于找到出错点在整个程序中的位置即第几行，还有错误点也要不多不少的按照格式输出来；以测试用例3为例说明：该函数对输入的内容一个字符一个字符的处理，同时过滤掉空格和换行符，用errstr记录当前的不带空格的字符串，当扫描到e这个字符时，我们想要让它输出带指数类型的小数，就要对errstr从头开始扫描直到找到第一个数字为止，用flagcount记录第一个数字的索引，这样就成功了，行数的判断很简单，只要扫描到换行符errorline就自增1

yylex函数功能总的来说就是判别各类单词类别，并按格式输出单词种类和单词自身的值，我们把测试用例中涉及到的单词分为7类，保存在str数组中，分别是integer,decimal,type,identify,bracket,OPT,float,这与我们之前考虑的好多关键字有很多不同，因为测试用例比较简单，没有涉及到循环等复杂语句的分析，所以我们做了大量删减。对每一类单词，基本的操作都是相似的，都是先判断这个单词的首字母符不符合他们的特征，是则进入{}内顺序执行语句，用word一步步记录下该单词，再经过一系列处理后输出类别和word，类别的区分用type可以简化记录，输出的时候是用str[type]；对于标识符和关键字，由于他们很相似，所以我们在处理标识符的过程中，就进行了是否为关键字的判断（关键字保存在关键字数组中）；此外，该函数最核心的部分就是对数字的处理，关键点在于如何输出负数，我们把对数字的处理分成3部分处理，一部分处理整数部分，一部分处理小数部分，还有一部分处理带指数的小数，type的值随着类型的改变预先改变以便处理下一条if语句；在处理过程中，用reg记录整数部分的符号，reg1记录指数部分的正负号；符号的处理放在对运算符号处理的那一步，这一步主要是改变reg和reg1的值

总结体会：

在解决问题的过程中，我们发现了定义中存在的许多不足，因此在定义上做了许多改进。针对第三问，我们想了许多解决办法，因为c语言没有字符串，因此只能用数组来存放信息，这势必会增加时间复杂度。百度到有yyless()函数，但是并不能解决问题，诸多尝试之后，还是选择了最暴力的数组存储法，一个一个字符存在数组中。

通过这次的实验，我们认识到，思考问题要全面，尽可能把可能涉及到的内容都考虑进来。二则就是要换变思维，正如解决错误问题一样，一开始我们考虑的一直是怎么检测错误，没有很快的反应过来可以直接定义一个错误的定义。

