一、实验目的

在这一阶段的实验中，同学们要完成Decaf编译器的词法分析工作，即用flex工具生成一个扫描程序，对Decaf语言编写的源程序从左至右逐个字符进行扫描，产生一个单词序列，用以进行下一阶段的语法分析。

二、Decaf语言的单词符号

与C、C＋＋等语言类似，Decaf语言的单词符号主要有以下5类：

1. 关键字： void int double bool string class extends this true false null while if else return

public private New NewArray Print ReadInteger ReadLine

1. 标识符：以字母开头，后面可以跟字母、数字和下划线。请注意，Decaf语言区分大小写，也就是说，if是关键字，而IF则是标识符。同样的，IF和If是两个不同的标识符。
2. 常量：包括三类：

·整数：整数常量可以是十进制或十六进制。十进制整数就是0～9的数字组成的序列，十六进制整数以0x（或0X）开头，后面跟由0～9和a～f（或A～F）组成的十六进制数。8，012，0x0，0X2aE都是合法的整数。

·实数：实数由十进制的整数部分、小数点和小数部分组成。实数可以表示成科学计数法，即用E（或e）后跟整数来表示10的幂。请注意，实数的整数部分和小数点不能省略，小数部分若为0，则可以省略，指数部分的符号不能省略。因此，.12e+2，1.2e2都是不合法的实数，而12.e＋2是合法的实数。

·字符串：字符串是嵌在双引号内的字符序列。字符串中可以包含任何字符，直至遇到换行或双引号结束。因此，Decaf的字符串不能换行：

“this is not a

valid string constant”

1. 运算符和界符：

·单字符：＋ － \* / ％ \ < > = ; , . ! [ ] ( ) { }

·双字符：&& || <= >= == !=

1. 注释：

·单行：以“//”开始，直行尾结束；

·多行：嵌在“/\*”和“\*/”直接的字符序列。请注意，Decaf运行多行注释的嵌套。例如，/\* everything inside here is a comment

and it can span multiple lines \*/

/\* unlike C, Decaf comments can

/\* nest /\* inside \*/ \*/

one another \*/

三、实验结果

1、Decaf语言的单词符号

关键字keyword直接用符号|分隔所有关键字即可。

标识符以字母开头，后跟字母或数字或下划线，且区分大小写，因此用[A-Za-z][A-Za-z0-9\\_]\*的正则表达式。

常量中整数有三种进制，十进制多位数首位不能为0，因此为[1-9][0-9]\*，十六进制以0x或0X开头，后面跟0-9和a-f或A-F，因此为0[xX]([0-9A-Fa-f])+，八进制则为0([0-7])\*；实数由整数部分和小数点和小数部分组成，整数部分0不能省略为([0-9])+，小数点不能省略为”.”，小数部分0可以省略为([0-9])\*，后接指数部分的符号（不可省略）和数字（可省略）为([eE][-+]?[0-9]+)?，因此一起为([0-9])+"."([0-9])\*([eE][-+]?[0-9]+)?；字符串由双引号括起来且不能换行，所以不能有\n，因此为\"[^"\n]\*\"。

运算符由单字符和双字符，直接用|分隔即可，注意使用转义符，因此为\+|-|\\*|\/|%|<|<=|>|>=|=|==|!=|&&|\|\||!|\.|\[|\]|\(|\)和[,;\{\}]。

注释有两种，第一种单行注释以//开始，为(\/\/[^\n]\*)；第二种为多行，嵌在“/\*”和“\*/”中，且可多层嵌套，正则表达式要注意开头和结尾不仅顺序要匹配且数量也要匹配，因此中间部分为((([^\\*])|(\\*[^\/]))\*)，加上两边的起始则为(\/\\*((([^\\*])|(\\*[^\/]))\*)\\*\/)。但此正则表达式有问题，在后面的输出中会提到。

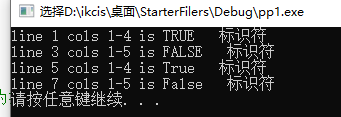
2、程序与输出

程序中用到了yytext和yylineno两个内建的量，yytext返回的是字符串形式的内容，yylineno返回的是所在的行数。计算字符个数位置的内建量在文档中没有找到，不知道有没有，因此程序中用一个全局变量num代替，初始值为1，遇字符加1，遇\n重置为1，来达到记录字符位置的功能。

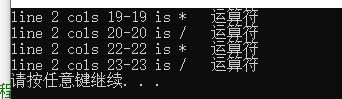
未实现的功能：记录首次出现的位置和次数。暂时想不到解决方法，待后续解决。

输出：（2）（8）（9）与out文件不一致

（1）badpool与out一致

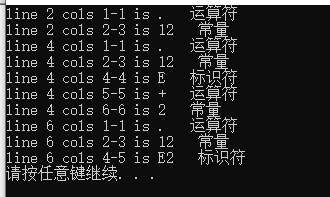


（2）badcomment.

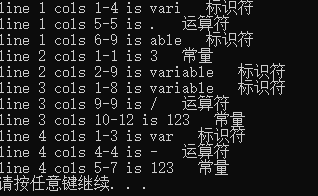


与out不一致。我程序所用的表达式只匹配第一次\*/的出现，从而会导致在单行嵌套注释的情况下，无法识别整个注释。想了很久没想到怎么改正则表达式，待解决。

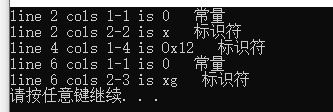
（3）baddouble与out一致



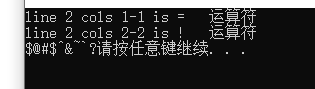
（4）badident与out一致



（5）badint与out一致

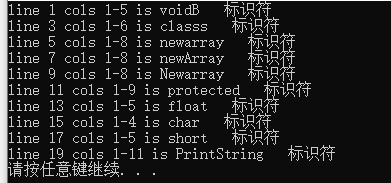


（6）badop与out一致

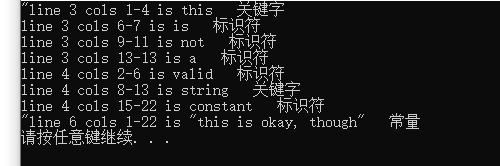


第三行为无法识别的字符

（7）badreserve与out一致

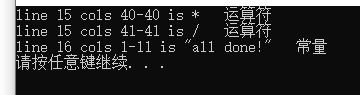


（8）badstring



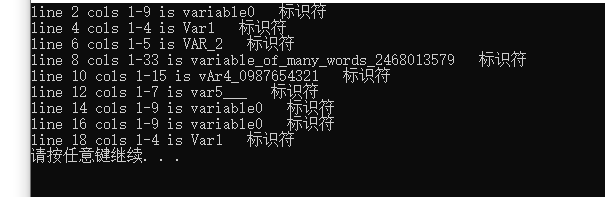
错误原因：文本中的“和printf中的”发生了匹配导致输出错误。没有考虑到关键字和引号在一起出现的情况。

（9）comment

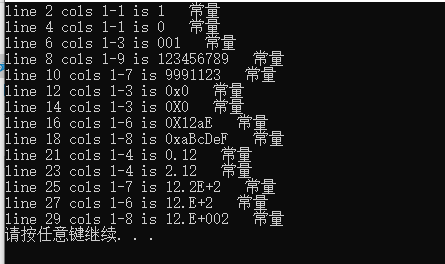


错误原因同（2）

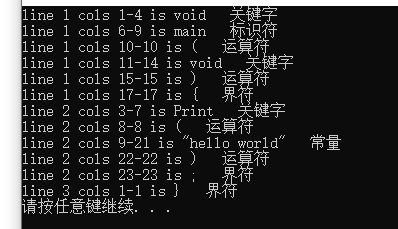
（10）ident与out一致



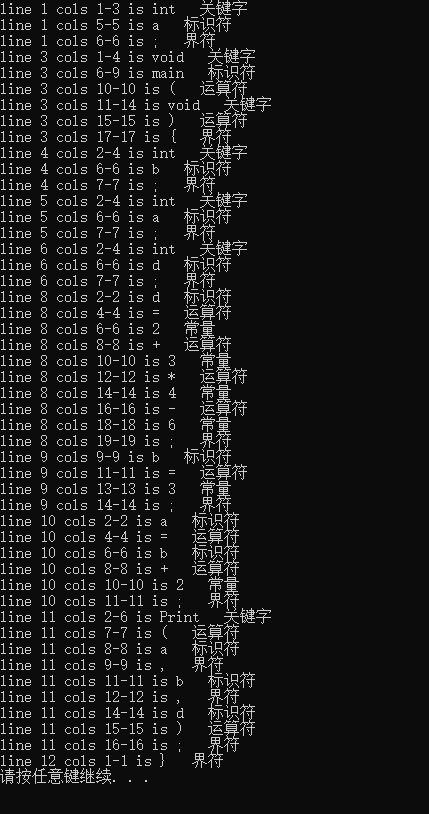
（11）number与out一致



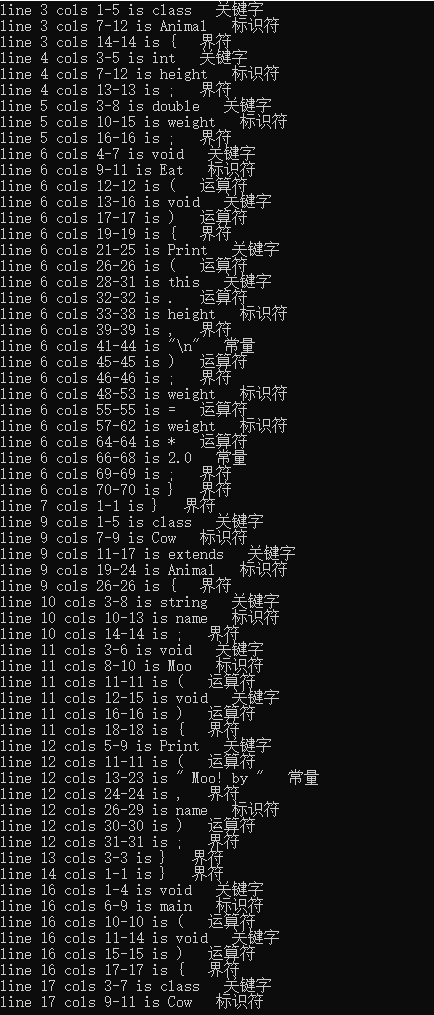
（12）program1与out一致



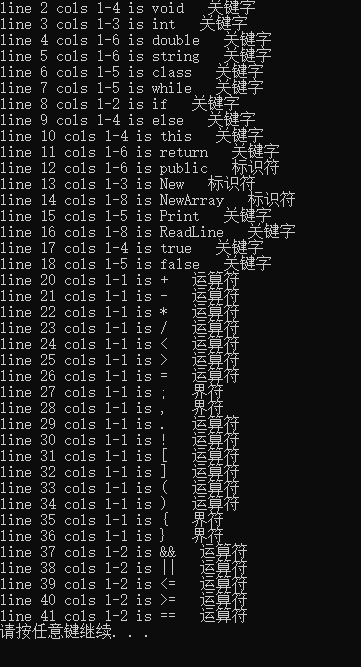
（13）program2与out一致



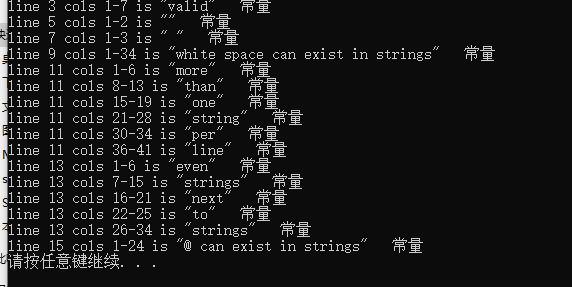
（14）program3与out一致



（15）reserve\_op与out一致



（16）string与out一致



3、实验体会

（1）之前一直没有搞清楚正则表达式中三种括号的区别，导致在写常量和运算符的正则表达式时一直报错。后来查资料了解[]是单个匹配，控制其取值范围，()本身不匹配或者限制什么，只是把某些项合在一起，与数学表达式中的用法类似，{}一般用于匹配长度。

（2）实验一开始拖着是不知道怎么记录当前行数，就写了几个全局变量在每一项里修改，但是调试的时候总是有问题，之后没办法只能去看文档，总算找到个yylineno。但是还是有些功能是用自定的全局变量记录数值的，有点麻烦。时间关系有一个功能也还没有实现，只能之后补起来了。

（3）程序运行感觉是把一部分代码替换到C代码之中，才会出现字符串的项莫名其妙被打印出来，问题还是出在写正则的时候没有特判特殊情况，比如关键词和“连在一起的时候。

（4）scanner.l里看了一万年没看出来的缺了一个空格导致run不出来的问题，之后再差文档才知道空格的必须性。

附录：

int num=1;

/\*

\*在这里定义你的辅助定义和开始条件

\*/

keyword bool|break|class|else|extends|for|if|int|double|new|null|return|string|this|void|while|static|Print|ReadInteger|ReadLine|instanceof|true|false

identifier [A-Za-z][A-Za-z0-9\\_]\*

dec\_num [1-9][0-9]\*

hex\_num 0[xX]([0-9A-Fa-f])+

oct\_num 0([0-7])\*

double\_num ([0-9])+"."([0-9])\*([eE][-+]?[0-9]+)?

str \"[^"\n]\*\"

numbers {dec\_num}|{hex\_num}|{oct\_num}|{double\_num}|{str}

operator \+|-|\\*|\/|%|<|<=|>|>=|=|==|!=|&&|\|\||!|\.|\[|\]|\(|\)

delimiter [,;\{\}]

comment1 (\/\/[^\n]\*)

comment2 (\/\\*((([^\\*])|(\\*[^\/]))\*)\\*\/)

comment {comment1}|{comment2}

whitespace [ \t]

enter [\n]

%option yylineno

%%

{keyword} {

printf("line %d cols %d-%d is %s 关键字\n",yylineno,num,num+strlen(yytext)-1,yytext);

num+=strlen(yytext);

}

{identifier} {

printf("line %d cols %d-%d is %s 标识符\n",yylineno,num,num+strlen(yytext)-1,yytext);

num+=strlen(yytext);

}

{numbers} {

printf("line %d cols %d-%d is %s 常量\n",yylineno,num,num+strlen(yytext)-1,yytext);

num+=strlen(yytext);

}

{operator} {

printf("line %d cols %d-%d is %s 运算符\n",yylineno,num,num+strlen(yytext)-1,yytext);

num+=strlen(yytext);

}

{delimiter} {

printf("line %d cols %d-%d is %s 界符\n",yylineno,num,num+strlen(yytext)-1,yytext);

num+=strlen(yytext);

}

{comment} {

num+=strlen(yytext);

}

{whitespace} {num+=1;}

{enter} {num=1;}

%%