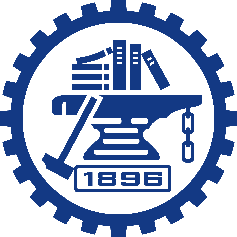
** 词法分析 C:\Users\tangbo\Desktop\学习\基础实践\SJTU PPT\gif\毛体校名（宝蓝色）2.gif**

1. **目标**

本次作业的目标为利用flex编写一个词法分析器，实现对Seal语言的词法提取。

1. **说明**

本次作业主要利用flex，定义Seal中的词法格式，从而分析Seal文件中的符号，将其转化为合适的Seal输出文件，用于进一步的语法分析。本次实验会牵涉到flex、正则表达式[[1]](#footnote-1)等内容，对于flex将有相关参考资料。

1. **flex简介**

flex是lex的一个开源实现（见课本附录A），它能够根据用户定义的正则表达式，对输入文件中的字符串进行匹配，并且对匹配的结果做出相应的处理。这也是词法分析器的基本功能。

flex能够将用户编写的规则文件编译为C源代码（C与C++语法相似，故可以以C++形式继续开发），而编译后的文件可以直接作为库引用。而库代码往往非常繁杂，但是此次作业重点不在编程，故在此次代码设计任务中，其他部分都已经给出，同学只需要编写相应的flex规则文件，也即词法构成规则即可。

flex规则文件的基本结构如下

%{

声明（Declarations）

%}

定义（Definitions）

%%

规则（Rules）

%%

处理函数（User subroutines）

声明和处理函数部分是可选的，可以在其中编写一些辅助用的函数。定义部分也是可选的，但是通常对正则表达式编写会非常有用，例如定义

LOWERCASE [a-z]

简明的定义了小写字母。flex中使用的常见的正则表达式可以查阅课本附录A，在此不再赘述。举例如[xy]，表示字符x或者y。

在flex中最重要的是规则部分，例如

[0-9]+ {// 处理函数}

将会对匹配到的符合[0-9]+的字符串（数字串）做对应的处理函数动作。注意，正则表达式以最长匹配原则，也即如果有[0-9]+和[0-9a-z]+两个正则式，则对于2a这个串，将按照第二个[0-9a-z]+匹配，而不按照一个[0-9]+一个[0-9a-z]+匹配，因为前一种匹配方式更长。

1. **文件说明**

* seal.flex

flex的规则文件，也即此次作业**要修改**的文件。此文件给出的形式为一个代码框架，注释将会指明在何处添加代码，这些注释只是为了能够确保词法分析器能够正常工作。除了这部分之外，允许对其他部分、大体框架做任意的修改，如果需要帮助，请阅读flex的手册。此外，任何需要编写的额外的辅助函数，都应该包含在这个文件中。

* 测试文件和测试答案

在test文件夹里给了10个样例文件，在test-answer文件夹中有对应的10个词法分析结果文件。可以利用自己编写的词法分析器预先分析，然后与结果对比，结果要求与样例答案完全一致。

1. **如何测试**

首先，在测试之前，请确保机器上安装了flex，具体的，ubuntu机器请运行sudo apt install flex -y来安装。

在编写完整个代码后，需要将整个代码构建为一个可执行的词法分析程序，在代码所在目录下运行make lexer命令，将构建出一个名为lexer的可执行文件，该文件即编写的词法分析器。请注意，任何代码中出现错误都将导致make失败，失败原因会提示，请根据错误提示debug。

得到lexer文件后，执行./lexer test.seal可以将test.seal中的代码的词法分析结果输出到终端上，抑或执行./lexer test.seal > test.lex将分析结果输出到test.lex文件中，查看相关的内容。

也可以运行脚本bash judge.sh，该脚本会直接调用已经在当前目录下编译好的词法分析器，对样例分析，并且与标准结果对比，输出对比结果。[[2]](#footnote-2)

此外，我们在test目录下提供了一个已经编写好的可执行的词法分析器，你可以使用这个分析器对已有的.seal测试文件进行测试，并与自己的词法分析器对比。

1. **处理结果要求**

此次作业要求提交的代码满足如下四个条件：

1. **输出结果**

要求将输入文件代码的每个识别出的单词符号，一行一个输出，格式为

#<单词符号出现行号> <类型> [值]

具体的符号类型请查阅Seal-parse.h中的相关定义，要求将每个符号都定义相关规则。例如，如果第三行匹配到了一个单词符号为BOOL\_CONST类型，且值为true，则对应行的输出为#3 BOOL\_CONST true。又例如，在第一行匹配到一个单词符号左花括号{，由于其没有值，故输出为#1 {。有值的类型包括Int、Float、Object、Bool、String和Type。这里，行号能够帮助定位错误。

可以参照样例test及对应答案test-answer。实际上，**输出函数已经给出了**，你并不用重新写，阅读lextest.cc的全局main函数和utilties.cc里的dump\_seal\_token函数会帮助到你。对于行号，需要做的就是在合适的时候将全局变量curr\_lineno加1。

1. **字符串**

需要强调的是，编写的词法分析器要对于字符串，要保持其原有的值，例如匹配到的结果"ab\ncd"，这是8个字符，输出的单词符号STR\_CONST的值应该为ab\ncd，5个字符。其中\n为ASCII中的换行符。具体的，参阅Seal语法手册Strings节，有符合规定的strings词法。注意到Seal中的字符串不允许空字符\0（ASCII码为0x00），但是字符串词法分析结果允许有\0（可能来源于原字符串为"\\0"）。事实上，这次作业的一个难点就是对Seal中各种字符串词法提取规则设置。（参考test3和test4及其答案，可能更能说明问题）

1. **字符串表**

通常，程序中会有很多重复的符号元素（例如同一个计算程序中可能多次调用函数符号add，亦或是多次调用函数符号printf，或者多次利用常数值符号PI），为了节约时间和空间，编译器往往会把这些符号元素存在一个*符号表*中。默认给出了idtable（变量及函数名）、inttable（整数常量）、floattable（浮点常量）、stringtable（字符串常量）四个符号表。

**我们要求**，分析到相关的词时，使用<表名>.add\_string(匹配到的字符串)，将该常量的字面值添加到符号表中，特别需要强调的是，Int类型，Seal支持十六进制和十进制表示，请将各种十六进制等统一转为十进制的形式再添加进字符串表中。

例如对于Int类型，对于匹配到的字符串，你**应该**有这样的语句

seal\_yylval.symbol = inttable.add\_string(yytext);

return (CONST\_INT);

其中，yytext为lex匹配到的字符串，inttable.add\_string函数将该字符串存储在符号表，并且返回符号表项的地址指针，然后将该指针赋给seal\_yylval.symbol。return的值为CONST\_INT，如前所述，你可以在seal-parse.h中找到要定义匹配正则式的语言符号（除了运算符，运算符在seal手册中有给出，可以自行查阅并定义）。

1. **错误处理**

词法分析器在分析过程中产生的任何错误，需要显式的打印出来，主要包括：

* 非法字符
* 字符串过长，设最大允许长度为256个字符
* 字符串包括一个未转义的新行，也即在引号内空行而没有带\
* 字符串未闭合遇到文件结束（EOF）
* 未匹配的多行注释符号\*/

此外，**请勿**检查其他任何非词法分析的错误，例如代码变量先使用后定义等。另外，当字符串是多行时，请在指示行的位置标明字符串最后一行的行号。

此外，请注意如果有某些单词符号没有定义正则表达式或者其处理函数，整个词法分析输出的结果将没有意义，所以，请确保定义的完备性。

1. **操作提示**

* 分析器对每个符号的分析结果都和输入有关。如前所述，需要每次将分析到的单词符号以#<单词符号出现行号> <类型> [值]的形式，这里的值存储在一个union联合体seal\_yylval中，这个定义在seal-parse.h中可以查到。对于object、int、string、float，值需要以symbol形式存储在Seal\_yylval.symbol中。对于布尔常数类型，值要存储在Seal\_yylval.boolean中
* 关于字符串表的操作，在作业中已经给出。如果需要，请查阅Seal语法手册，现在仅仅需要知道字符串表的项是符号（symbol）即可。
* 当词法分析器出现任何错误时，Seal\_yylex应该返回一个符号ERROR，匹配到的词的值是存储在Seal\_yylval.error\_msg中的，是具体的错误信息。请注意，Seal\_yylval.error\_msg是一个普通的string，而不是symbol符号。

1. **如何评分**

作业提交之后，我们将根据同学提交的词法分析器对若干个已有的样本进行分析（与给出的实例样本不完全相同），其中样本可能包含词法错误，包含错误的样本要求输出错误及其行，给出错误原因和出错行为通过，无错误样本要求将输出结果与标准结果比对，完全一致的通过，否则不通过。按照所有样本分析通过率给分（即例如满分10分，通过率0%，给0分，通过率70%，给7分）。

1. **文件提交要求**

要求将原作业目录下的所有源代码文件（不包括测试文件\*.seal、临时文件\*.o以及自行编译的文件lexer，只需要源代码，我们将在自己的机器上make生成并检查），放置在一个名为<学号>的目录下，并且将整个目录打包为<学号>\_<姓名>.tar格式。

1. **提交截止时间**

请同学们在2020.11.2的0:00（2020.11.1的24:00）之前，将结果提交到canvas（迟交结果按0分处理）。

1. 可参考https://www.runoob.com/regexp/regexp-syntax.html [↑](#footnote-ref-1)
2. 因为lex本身的问题，词法分析运行结果可能会出现段错误，多试几次又可能正常。请保证能够正常执行，方便检测。 [↑](#footnote-ref-2)