项目测试报告

任务一

1、准备工作

tiny 的正则表达式：

```
if|then|else|end|repeat|until|read|write
\+\ | - | \*\ | / | % | < | <> | <= | >= | > | = | ; | := | \(\ |
\)\
PLUS | MINUS | MULTIPLY | DIVIDE | MOD | LT| NE | LTEQ | RTEQ | RT |
EQ | SEMI | ASSIGN | LPAN | RPAN
letter(letter|num)*
num num*
{~}
```

sample.tny 文件：

```
read xXx; {read 和标识符判断}
if ( xXx < yYy )  then {if 和<判断}
    repeat{repeat 判断}
        xXx := (xXx + 1) * 1 - 0 /1{+-*/的判断}
    until xXx = yYy{until 和=判断}
else{else 判断}
    write 001+123 % 2{%和 write、数字的判断}
end;{end 判断}
read z1;
{各种符号判断}
if z1 <= 1000 then
    z1 := 0
else
    z1 := 1
end;
if z1 <> 1000 then
    z1 := 0
else
    z1 := 1
end;
if z1 >= 1000 then
    z1 := 0
else
    z1 := 1
end;
if z1 > 1000 then
    z1 := 0
else
```

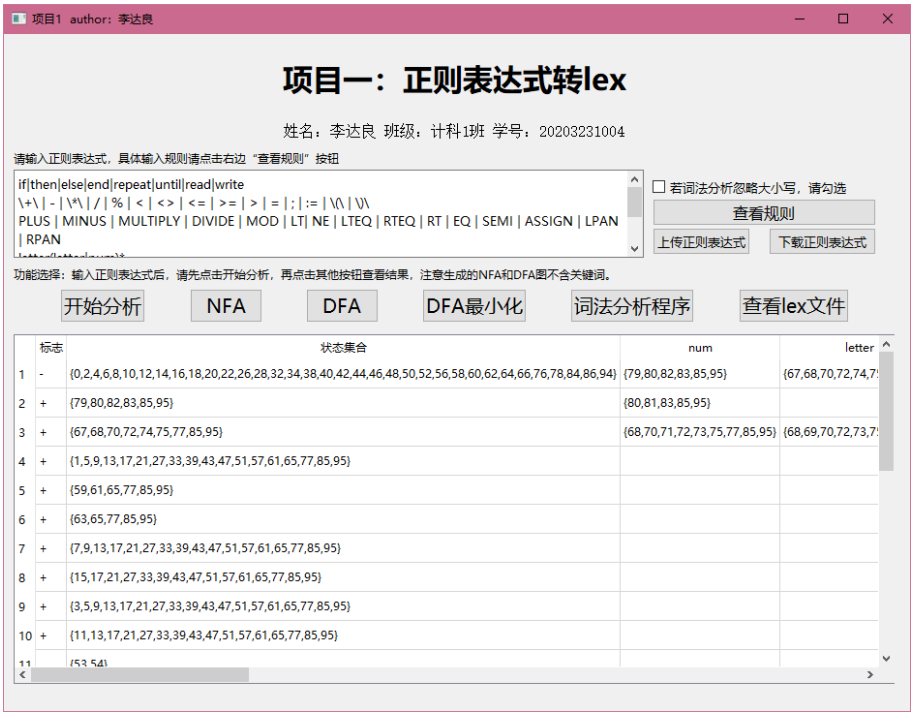
```
    z1 := 1
end
```

输入正则表达式：

2、NFA

	标志	ID	num	letter	+	(	)	*	#	%	-	/	:	;	<	=	>	{	}	非)
1	-	94							84,86											
2		84							76,78											
3		86																87		
4		87							90											
5		90							88,91											
6		88																		89
7		91							92											
8		92																	93	
9		93							95											
10		89							88,91											
11		76							64,66											

3、DFA 图

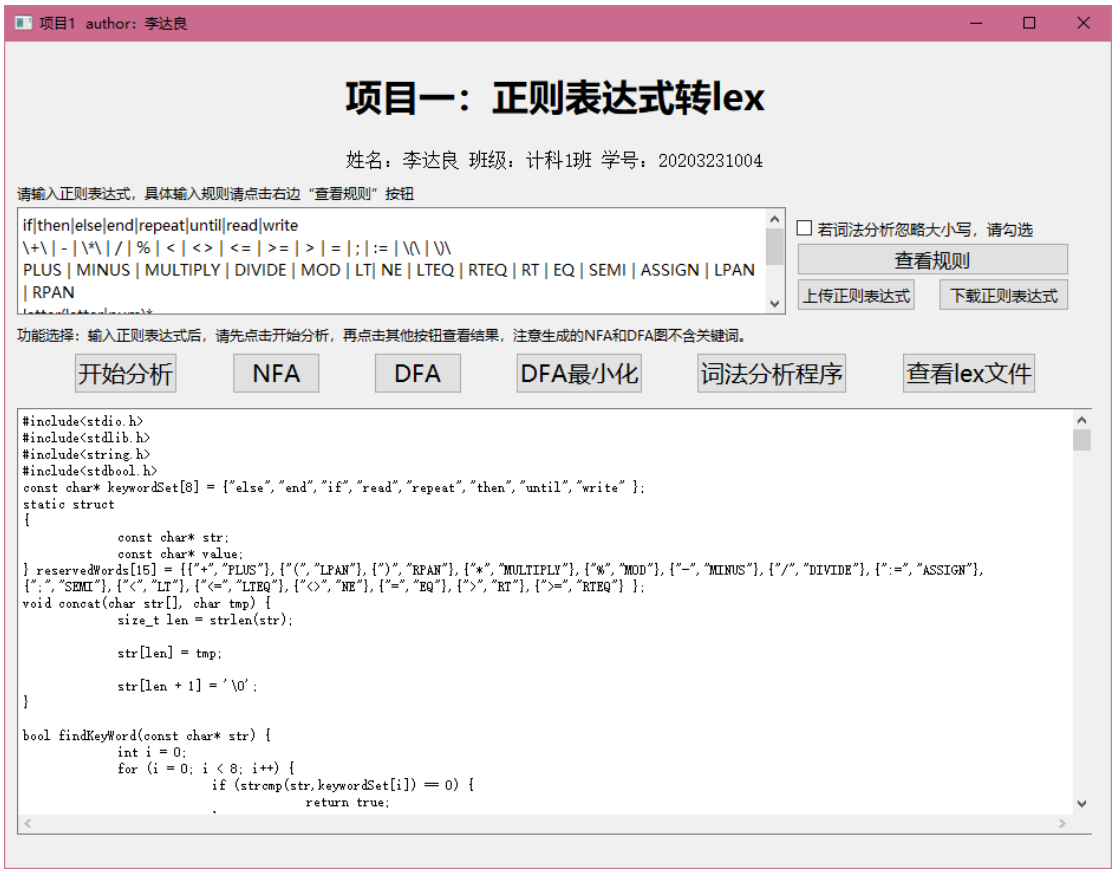


	标志	状态集合	num	letter
1	-	{0,2,4,6,8,10,12,14,16,18,20,22,26,28,32,34,38,40,42,44,46,48,50,52,56,58,60,62,64,66,76,78,84,86,94}	{79,80,82,83,85,95}	{67,68,70,72,74,7
2	+	{79,80,82,83,85,95}	{80,81,83,85,95}	
3	+	{67,68,70,72,74,75,77,85,95}	{68,70,71,72,73,75,77,85,95}	{68,69,70,72,73,7
4	+	{1,5,9,13,17,21,27,33,39,43,47,51,57,61,65,77,85,95}		
5	+	{59,61,65,77,85,95}		
6	+	{63,65,77,85,95}		
7	+	{7,9,13,17,21,27,33,39,43,47,51,57,61,65,77,85,95}		
8	+	{15,17,21,27,33,39,43,47,51,57,61,65,77,85,95}		
9	+	{3,5,9,13,17,21,27,33,39,43,47,51,57,61,65,77,85,95}		
10	+	{11,13,17,21,27,33,39,43,47,51,57,61,65,77,85,95}		
11		{53,54}		

4、DFA 图最小化

	标志	ID	num	letter	+	(	)	*	%	-	/	:	;	<	=	>	{	}	非)
1	-	0	1	4	3	3	3	3	3	3	3	2	3	6	3	7	5		
2	+	1	1																
3		2													3				
4	+	3																	
5	+	4	4	4															
6		5																3	5
7	+	6													3	3			
8	+	7													3				

5、生成词法程序

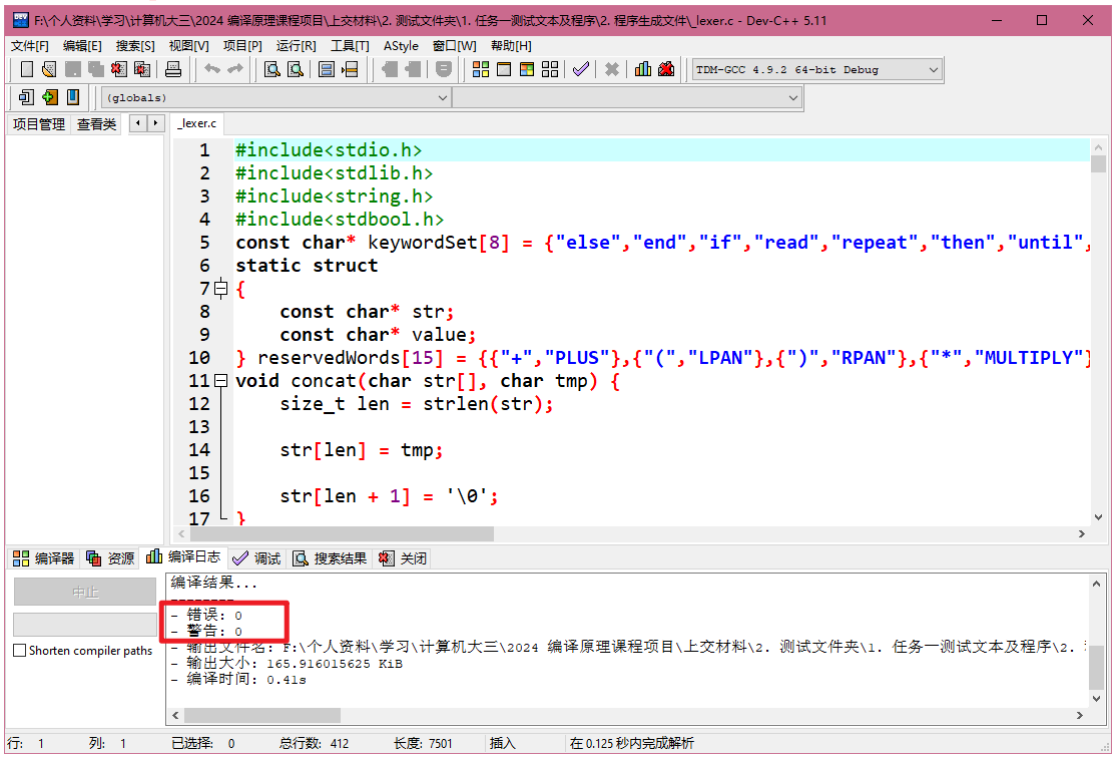



```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
const char* keywordSet[8] = {"else","end","if","read","repeat","then","until","write" };
static struct
{
            const char* str;
            const char* value;
} reservedWords[15] = {{"+","PLUS"},{"(","LPAN"},{")","RPAN"},{"*","MULTIPLY"},{"%","MOD"},{"-","MINUS"},{"/","DIVIDE"},{":=","ASSIGN"},
{";","SEMI"},{"<","LT"},{"<=","LTEQ"},{"<>","NE"},{"=","EQ"},{">","RT"},{">=","RTEQ"} };
void concat(char str[], char tmp) {
            size_t len = strlen(str);

            str[len] = tmp;

            str[len + 1] = '\0';
}

bool findKeyWord(const char* str) {
            int i = 0;
            for (i = 0; i < 8; i++) {
                        if (strcmp(str,keywordSet[i]) == 0) {
                                    return true;
```

代码太长，具体可查看：
2. 测试文件夹\1. 任务一测试文本及程序\2. 程序生成文件中的_lexer.c 文件

6、编译_lexer.c 文件并运行

注意：sample.tny 必须和该程序放在同一个文件夹下。



上图可知：编译成功
并成功生成 lex 文件：

7、查看 lex 文件

具体 lex 如下：

```
read:read
ID:xXx
SEMI:;
if:if
LPAN:(
ID:xXx
LT:<
ID:yYy
RPAN:)
then:then
repeat:repeat
ID:xXx
ASSIGN:=
LPAN:(
ID:xXx
PLUS:+
NUMBER:1
RPAN:)
MULTIPLY:*
NUMBER:1
MINUS:-
NUMBER:0
```

```
DIVIDE:/
NUMBER:1
until:until
ID:xXx
EQ:=
ID:yYy
else:else
write:write
NUMBER:001
PLUS:+
NUMBER:123
MOD:%
NUMBER:2
end:end
SEMI:;
read:read
ID:z1
SEMI:;
if:if
ID:z1
LTEQ:<=
NUMBER:1000
then:then
ID:z1
ASSIGN::=
NUMBER:0
else:else
ID:z1
ASSIGN::=
NUMBER:1
end:end
SEMI:;
if:if
ID:z1
NE:<>
NUMBER:1000
then:then
ID:z1
ASSIGN::=
NUMBER:0
else:else
ID:z1
ASSIGN::=
NUMBER:1
```

```
end:end
SEMI:;
if:if
ID:z1
RTEQ:>=
NUMBER:1000
then:then
ID:z1
ASSIGN::=
NUMBER:0
else:else
ID:z1
ASSIGN::=
NUMBER:1
end:end
SEMI:;
if:if
ID:z1
RT:>
NUMBER:1000
then:then
ID:z1
ASSIGN::=
NUMBER:0
else:else
ID:z1
ASSIGN::=
NUMBER:1
end:end
EOF:EOF
```

对照 tiny 源程序，可知解析完全正确。

测试结果

任务一测试完全通过