项目测试报告

任务三

文法分析部分

1、准备工作

mini-c 的正则表达式：

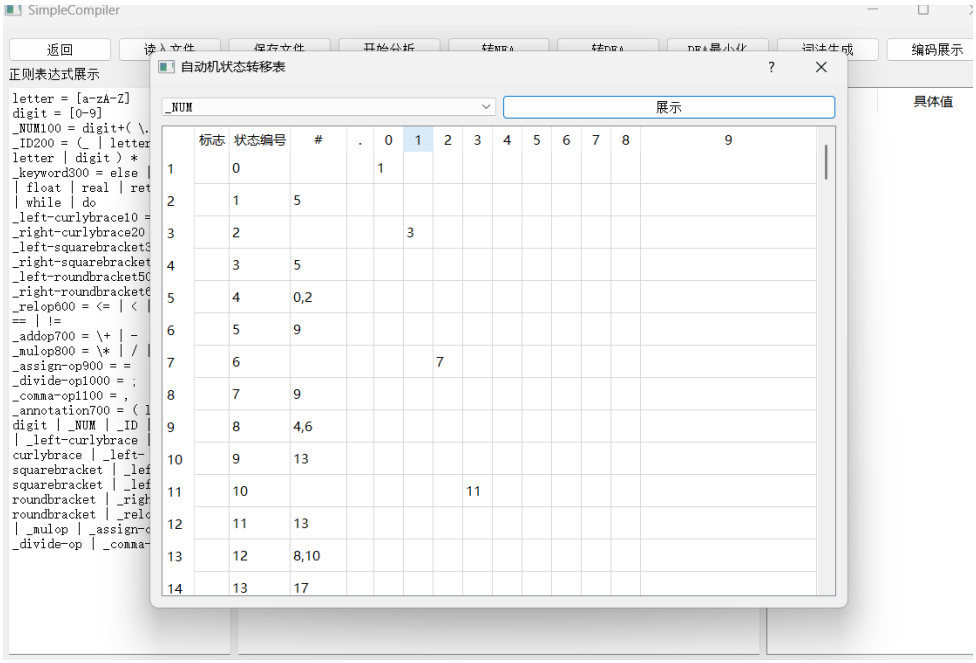
```
letter = [a-zA-Z]
digit = [0-9]
_NUM100 = digit+( \. digit+)?
_ID200 = (_ | letter) ( _ | letter | digit ) *
_keyword300 = else | if | int | float | real | return | void | while | do
_left-curlybrace10 = {
_right-curlybrace20 = }
_left-squarebracket30 = \[
_right-squarebracket40 = \]
_left-roundbracket50 = \(
_right-roundbracket60 = \)
_relop600 = <= | < | > | >= | == | !=
_addop700 = \+ | -
_mulop800 = \* | / | % | ^
_assign-op900 = =
_divide-op1000 = ;
_comma-op1100 = ,
_annotation700 = ( letter | digit | _NUM | _ID | _keyword | _left-curlybrace | _right-curlybrace | _left-squarebracket | _left-squarebracket | _left-roundbracket | _right-roundbracket | _relop | _addop | _mulop | _assign-op | _divide-op | _comma-op)
```

sample.tny 文件：

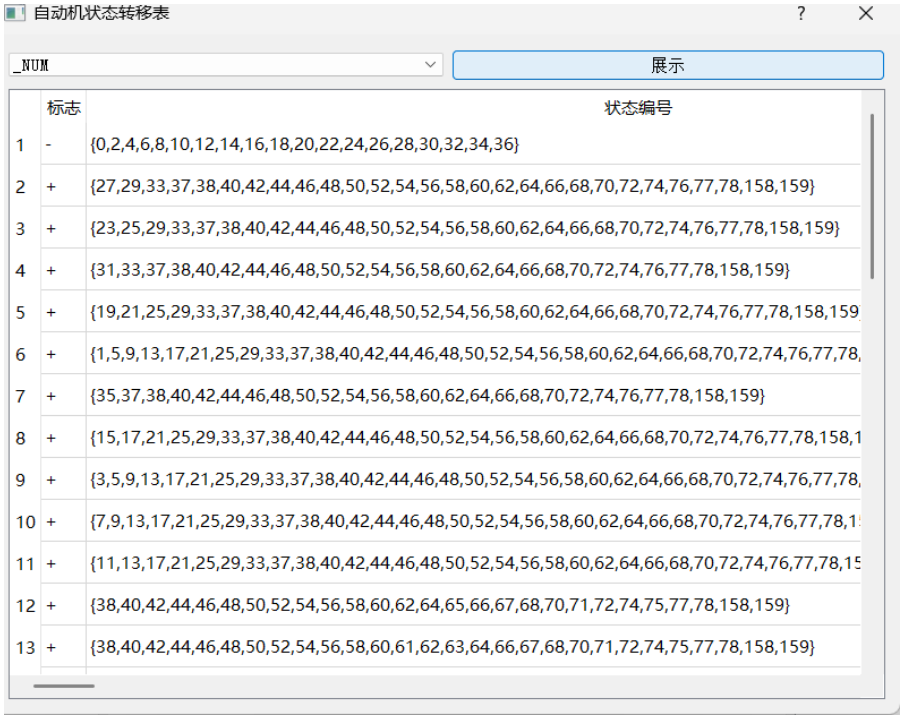
```
real ans;

//fact
int fact(int x) {
	if (x == 0) return 1;
	else return fact(x - 1) * x;;
}

//main
int main(void) {
	int n;
	int t;
	int x;
```

```
    int a[10];
    t = n = 10;
    x = (x - 1) * x % 10 ^ x / (x + 1);
    while (t >= 0) {
        a[t] = fact(a[t]);
        t = t - 1;
    };
    do {
        ans = ans * a[t] * 0.1;
        t = t + 1;
    } while(t <= n);
    if (ans < 0) {
        ans = 0;
    } else {
        if (ans > 123456) ans = 123456;;
    };
    if (ans != 123456) ans = ans + 1;;
    return 0;
}
```

2、NFA 图

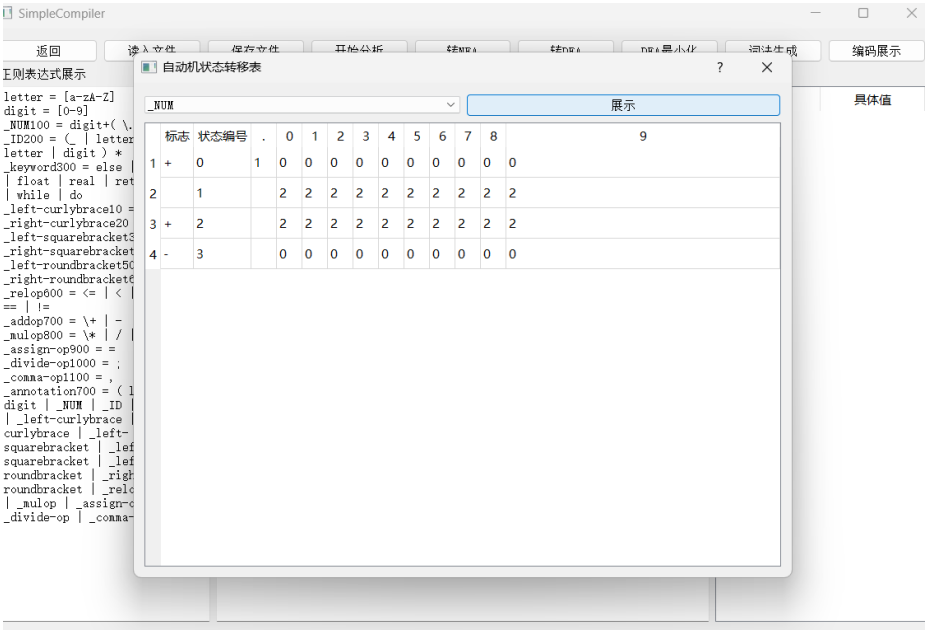


3、DFA 图



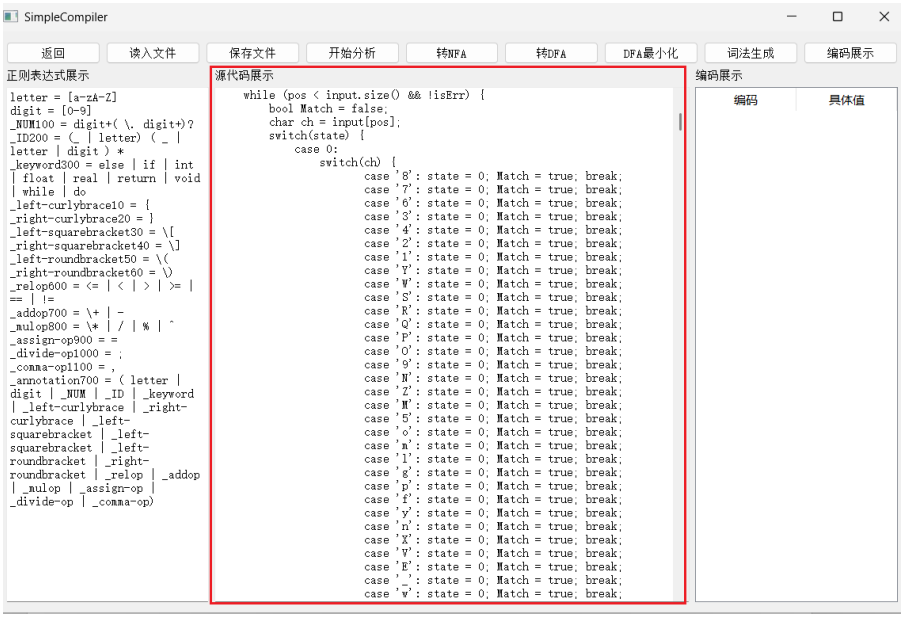
	标志	状态编号
1	-	{0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36}
2	+	{27,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,158,159}
3	+	{23,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,158,159}
4	+	{31,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,158,159}
5	+	{19,21,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,158,159
6	+	{1,5,9,13,17,21,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,
7	+	{35,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,158,159}
8	+	{15,17,21,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,158,1
9	+	{3,5,9,13,17,21,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,
10	+	{7,9,13,17,21,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,1
11	+	{11,13,17,21,25,29,33,37,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,77,78,15
12	+	{38,40,42,44,46,48,50,52,54,56,58,60,62,64,65,66,67,68,70,71,72,74,75,77,78,158,159}
13	+	{38,40,42,44,46,48,50,52,54,56,58,60,61,62,63,64,66,67,68,70,71,72,74,75,77,78,158,159}

4、DFA 图最小化



	标志	状态编号	.	0	1	2	3	4	5	6	7	8	9
1	+	0	1	0	0	0	0	0	0	0	0	0	0
2		1		2	2	2	2	2	2	2	2	2	2
3	+	2		2	2	2	2	2	2	2	2	2	2
4	-	3		0	0	0	0	0	0	0	0	0	0

5、生成词法程序



6、编译 cpp 文件运行，并查看最终 minic 编码文件

具体 minic 如下：

```
300 real 200 ans 1000 ; 300 int 200 fact 50 ( 300 int 200 x 60 ) 10 { 300 if 50 ( 200 x 600 == 100 0 60 ) 300 return 100 1 1000 ; 300 else 300 return 200 fact 50 ( 200 x 700 - 100 1 60 ) 800 * 200 x 1000 ; 1000 ; 20 } 300 int 200 main 50 ( 300 void 60 ) 10 { 300 int 200 n 1000 ; 300 int 200 t 1000 ; 300 int 200 x 1000 ; 300 int 200 a 30 [ 100 10 40 ] 1000 ; 200 t 900 = 200 n 900 = 100 10 1000 ; 200 x 900 = 50 ( 200 x 700 - 100 1 60 ) 800 * 200 x 800 % 100 10 800 ^ 200 x 800 / 50 ( 200 x 700 + 100 1 60 ) 1000 ; 300 while 50 ( 200 t 600 >= 100 0 60 ) 10 { 200 a 30 [ 200 t 40 ] 900 = 200 fact 50 ( 200 a 30 [ 200 t 40 ] 60 ) 1000 ; 200 t 900 = 200 t 700 - 100 1 1000 ; 20 } 1000 ; 300 do 10 { 200 ans 900 = 200 ans 800 * 200 a 30 [ 200 t 40 ] 800 * 100 0.1 1000 ; 200 t 900 = 200 t 700 + 100 1 1000 ; 20 } 300 while 50 ( 200 t 600 <= 200 n 60 ) 1000 ; 300 if 50 ( 200 ans 600 < 100 0 60 ) 10 { 200 ans 900 = 100 0 1000 ; 20 } 300 else 10 { 300 if 50 ( 200 ans 600 > 100 123456 60 ) 200 ans 900 = 100 123456 1000 ; 1000 ; 20 } 1000 ; 300 if 50 ( 200 ans 600 != 100 123456 60 ) 200 ans 900 = 200 ans 700 + 100 1 1000 ; 1000 ; 300 return 100 0 1000 ; 20 }
```

语法分析部分

1、准备工作

tiny 的文法

```
program -> definition-list
definition-list -> definition-list definition | definition
definition -> variable-definition | function-definition
variable-definition -> type-indicator ID ; | type-indicator ID [ NUM ] ;
type-indicator -> int | float | real | void
```

```
function-definition -> type-indicator ID ( parameters ) compound-stmt
parameters -> parameter-list | void
parameter-list -> parameter-list , parameter | parameter
parameter -> type-indicator ID | type-indicator ID [ ]
compound-stmt-> { local-definitions statement-list }
local-definitions-> local-definitions variable-definition | #
statement-list-> statement-list statement | #
statement -> expression-stmt | compound-stmt | condition-stmt | while-stmt | dowhile-stmt | return-stmt
expression-stmt -> expression ; | ;
condition-stmt-> if ( expression ) statement ; | if ( expression ) statement else statement ;
while-stmt -> while ( expression ) statement ;
dowhile-stmt -> do statement while ( expression ) ;
return-stmt -> return ; | return expression ;
expression -> variable = expression | simple-expression
variable -> ID | ID [ expression ]
simple-expression -> additive-expression relop additive-expression | additive-expression
relop -> <= | < | > | >= | == | !=
additive-expression -> additive-expression addop term | term
addop -> + | -
term -> term mulop factor | factor
mulop -> * | / | % | ^
factor -> ( expression ) | variable | call | NUM
call -> ID ( arguments )
arguments -> argument-list | #
argument-list -> argument-list , expression | expression
```

minic 的 sample.minic 编码文本（见上词法分析最终结果的编码文件）

tiny 的语义动作表

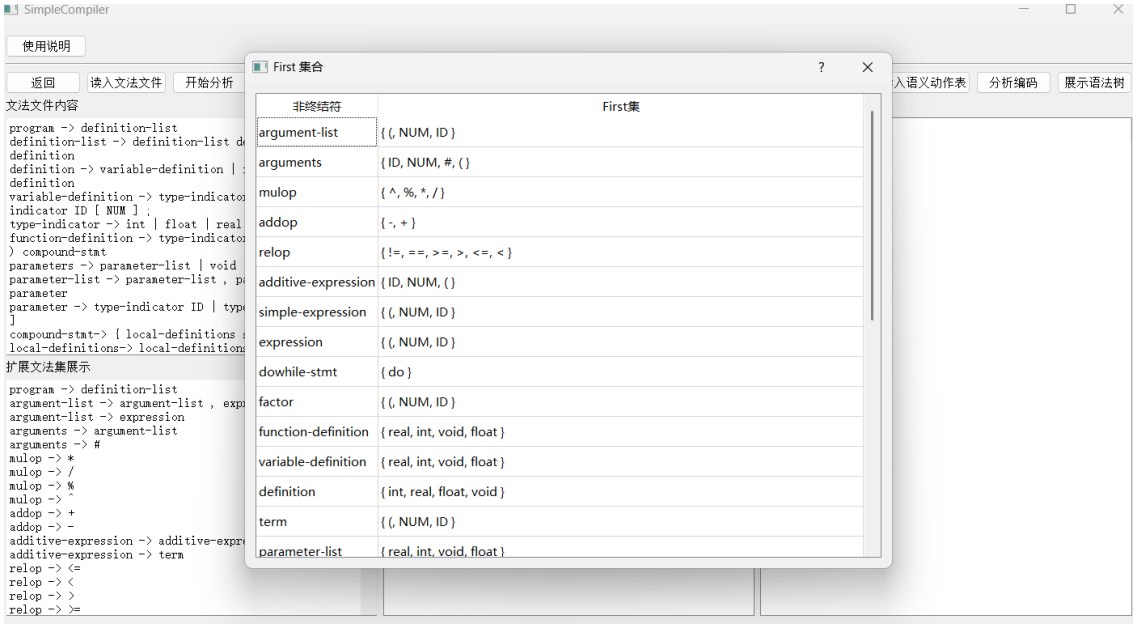
```
program -> stmt-sequence
1
stmt-sequence -> stmt-sequence ; statement
1 0 3
stmt-sequence -> statement
1
statement -> if-stmt
```

```
1
statement -> repeat-stmt
1
statement -> assign-stmt
1
statement -> read-stmt
1
statement -> write-stmt
1
if-stmt -> if exp then stmt-sequence end
1 2 0 2 0
if-stmt -> if exp then stmt-sequence end else stmt-sequence end
1 2 0 2 0 0 2 0
repeat-stmt -> repeat stmt-sequence until exp
1 2 0 2
assign-stmt -> identifier := exp
2 1 2
read-stmt -> read identifier
1 2
write-stmt -> write exp
1 2
exp -> simple-exp comparison-op simple-exp
2 1 2
exp -> simple-exp
1
comparison-op -> <
1
comparison-op -> >
1
comparison-op -> =
1
comparison-op -> <=
1
comparison-op -> <>
1
comparison-op -> >=
1
simple-exp -> simple-exp addop term
```

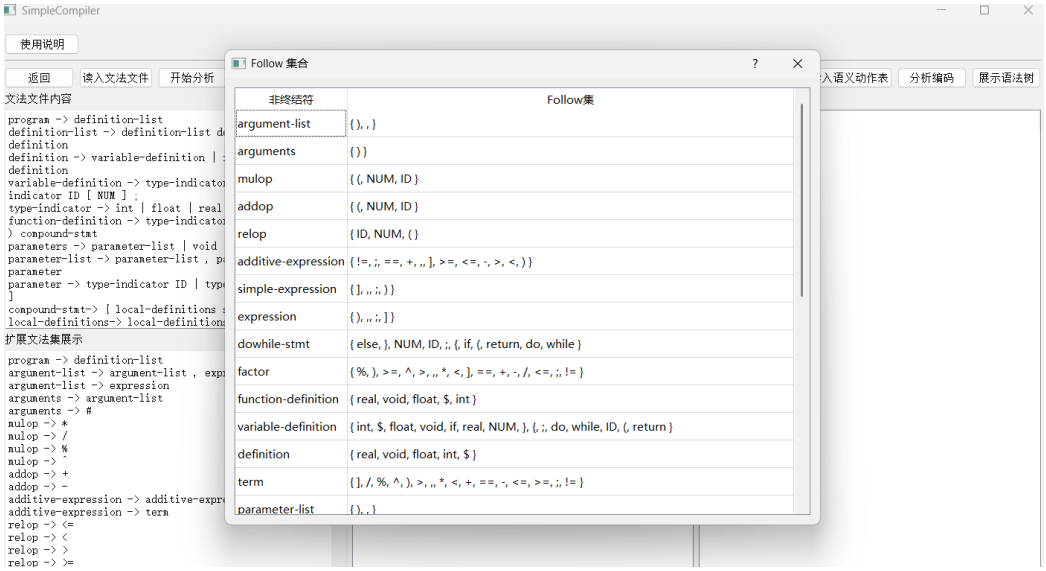
```
2 1 2
simple-exp -> term
1
addop -> +
1
addop -> -
1
term -> term mulop factor
2 1 2
term -> factor
1
mulop -> *
1
mulop -> /
1
mulop -> %
1
mulop -> ^
1
factor -> ( exp )
0 1 0
factor -> number
1
factor -> identifier
1
```

2、开始测试

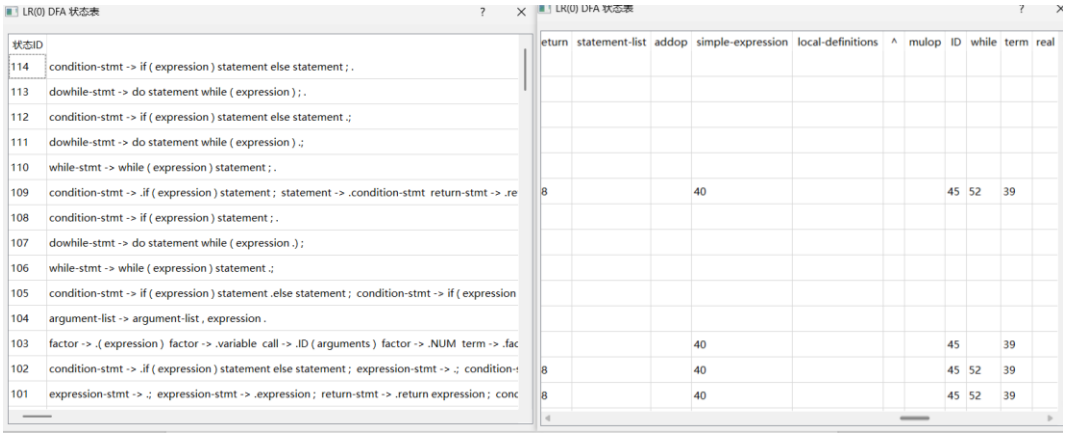
① First 集合：



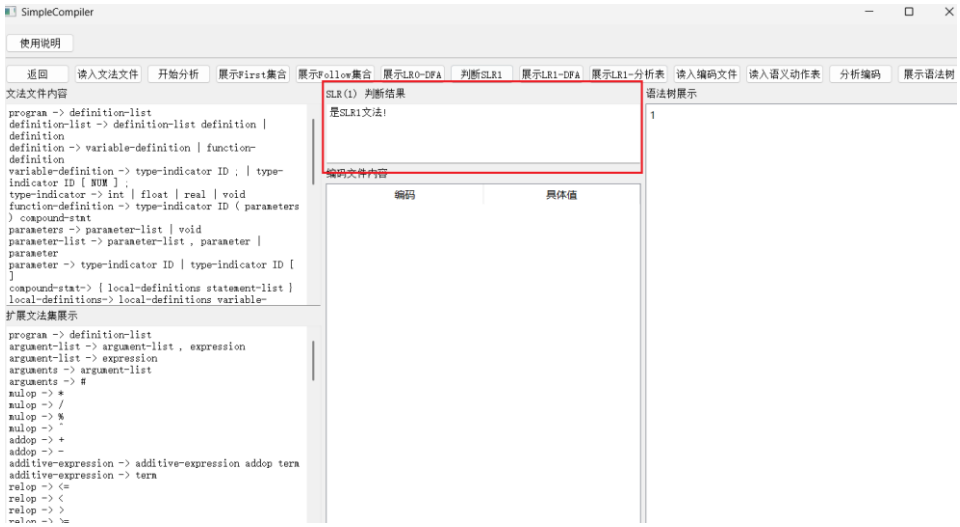
② Follow 集合：



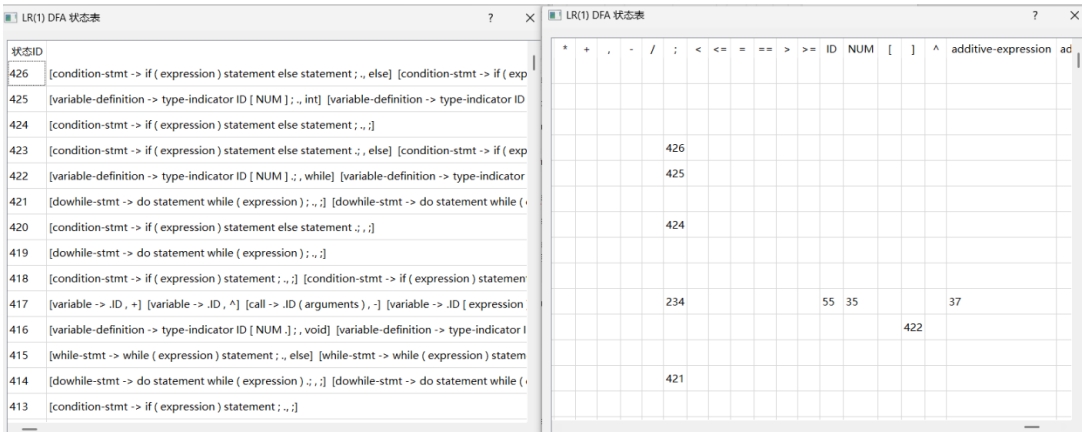
③ LR0-DFA：



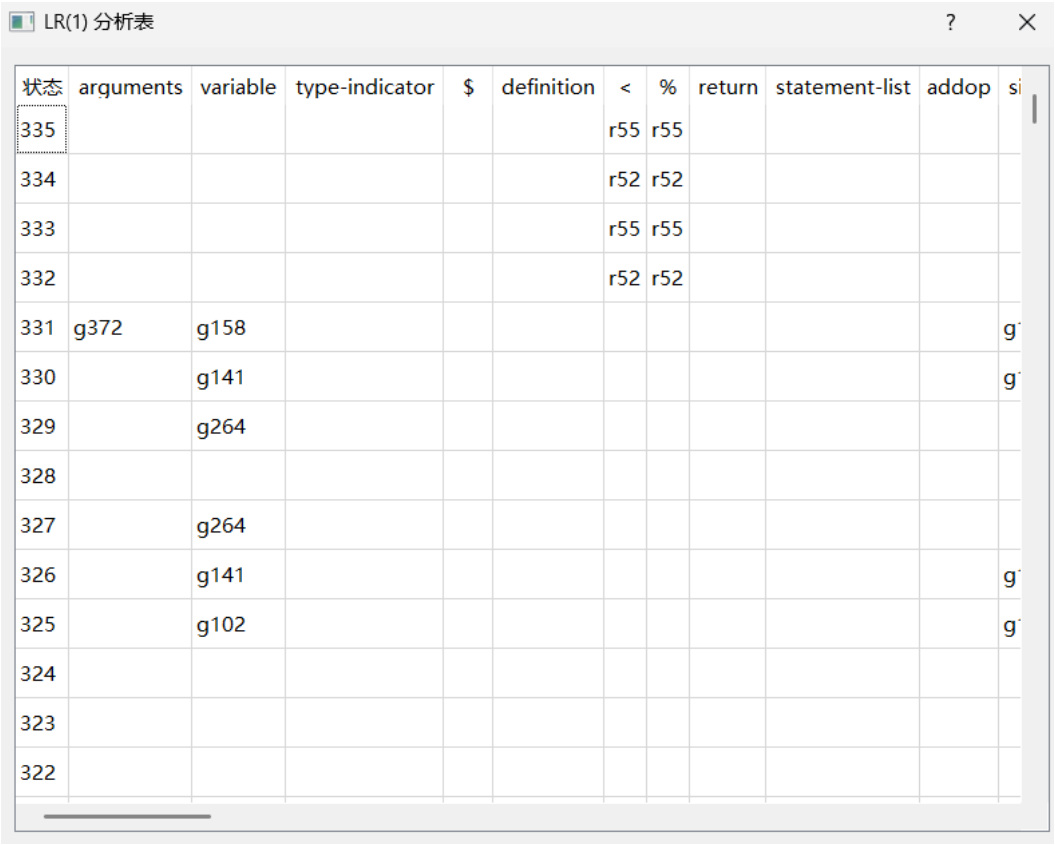
④ 判断是否为 SLR1：



⑤ LR1-DFA：

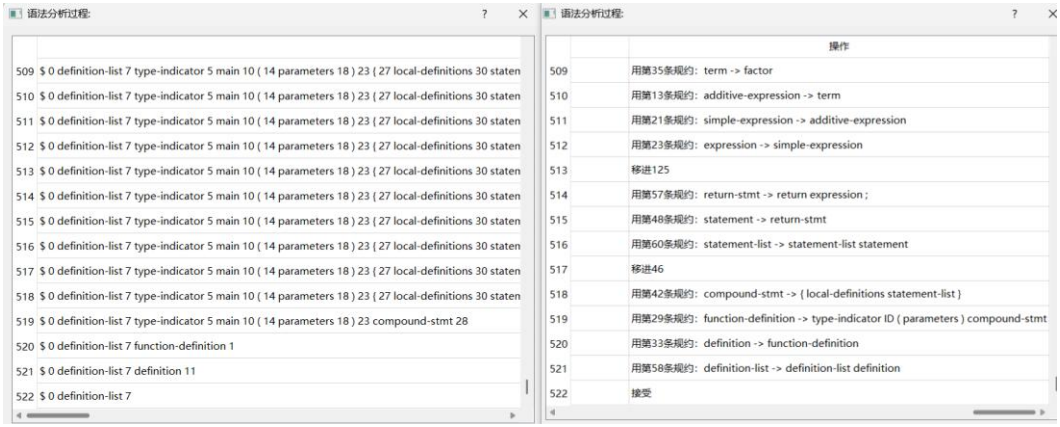


⑥ LR1-分析表：

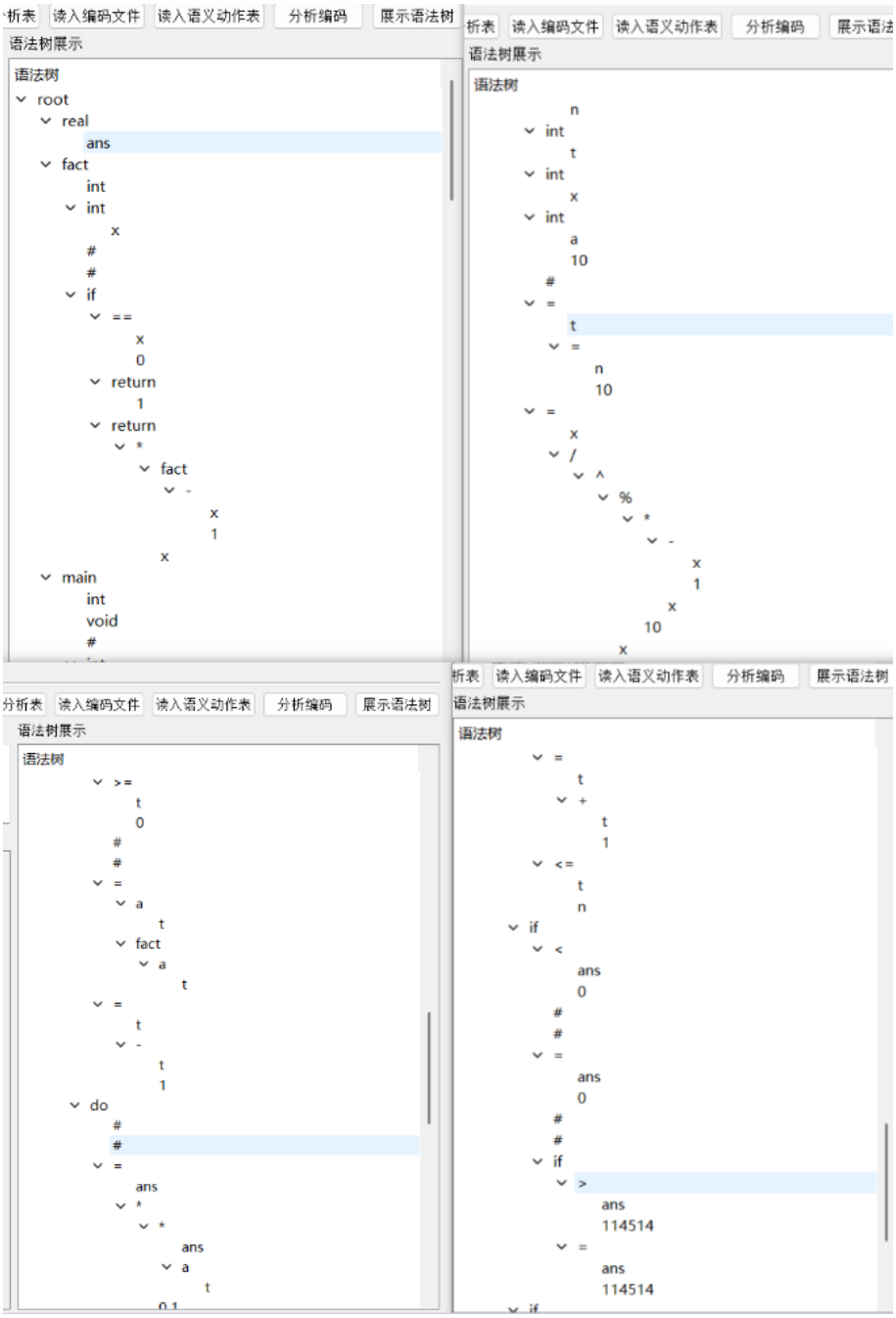


状态	arguments	variable	type-indicator	$	definition	<	%	return	statement-list	addop	si
335						r55	r55				
334						r52	r52				
333						r55	r55				
332						r52	r52				
331	g372	g158									g
330		g141									g
329		g264									
328											
327		g264									
326		g141									g
325		g102									g
324											
323											
322											

⑦ 分析编码过程展示



⑧ 语法树展示



总结：根据我们最后展示的语法分析过程的结果，以及语法树的展示结果来看，我们本次任务三的测试完全通过。