SLR(1)文法生成器,分析器报告

0. 组员

组长: t吴文正
组员: t程远 t韩霈霆

1. 有关文法的假设

非ASCII字符转换

由于原文法中符号∧及∨,以及形如$\hat{S}$的变元都不是字符,因而作如下处理:

- ∧改为AND
- ∨改为OR
- $\hat{S}$改为S'
- 空生成式用下划线_表示
  修改后的文法如同:

```
P  -> D' S'
D' -> _
D' -> D' D ;
D  -> T  d
D  -> T  d [ i ]
D  -> T  d ( A' ) { D' S' }
T  -> int
T  -> void
A' -> _
A' -> A' A ;
A  -> T  d
A  -> d  [ ]
A  -> T  d ( )
S' -> S
S' -> S' ; S
S  -> d = E
S  -> if ( B ) S
S  -> if ( B ) S else S
S  -> while ( B ) S
S  -> return E
S  -> { S' }
```

```
S  -> d ( R' )
B  -> B AND B
B  -> B OR B
B  -> E r E
B  -> E
E  -> d = E
E  -> i
E  -> d
E  -> d ( R' )
E  -> E + E
E  -> E * E
E  -> ( E )
R' -> _
R' -> R' R ,
R  -> E
R  -> d [ ]
R  -> d ( )
```

以变元P为初始符号.

2. 文法规则的生成/文法生成程序的运行结果

文法规则由我们自己编写的程序生成,该程序见附件中的`Generator.cpp`及`Generator.exe`,该程序或需支持C++23标准之编译器才能重新编译.(所有文件的副本也在报告的最后,下同.)

该程序接受两个文件,一个是语法文件,另一个是人工计算得到的FOLLOW集文件.其中,语法文件接受的是原始文法,由n+1行构成,第1行是指定的初始符号,随后的n行是原文法的n个产生式,由文法生成程序在运行时产生增广文法.我们所用的输入文件为附件中的`gram.txt`和`follow.txt`.

由于给定文法在采取SLR(1)消解冲突后仍有13个shift-reduce冲突,因而该程序在产生结果时允许手动为每个冲突采取消解策略.

由该程序生成的结果包括两部分,一部分是ItemDFA的状态表(见附件`states.txt`),另一部分是转移表(包括了ACTION table及GOTO table,见邮件附件`trans.xlsx`).

在标准输入输出中,允许手动消解冲突,其中带>的行是输入.

```
DFA states count: 84
Conflict No.1detected: for state 1, Symbol #, acc/r0
Please select action for this conflict:
0: acc
1: r0
2: ignore
```

```
> 0
Conflict No.2detected: for state 10, Symbol +, s9/r31
Please select action for this conflict:
0: s9
1: r31
2: ignore
> 1
Conflict No.3detected: for state 10, Symbol *, s11/r31
Please select action for this conflict:
0: s11
1: r31
2: ignore
> 0
Conflict No.4detected: for state 12, Symbol +, s9/r32
Please select action for this conflict:
0: s9
1: r32
2: ignore
> 1
Conflict No.5detected: for state 12, Symbol *, s11/r32
Please select action for this conflict:
0: s11
1: r32
2: ignore
> 1
Conflict No.6detected: for state 15, Symbol +, s9/r27
Please select action for this conflict:
0: s9
1: r27
2: ignore
> 0
Conflict No.7detected: for state 15, Symbol *, s11/r27
Please select action for this conflict:
0: s11
1: r27
2: ignore
> 0
Conflict No.8detected: for state 29, Symbol ), s30/r34
Please select action for this conflict:
0: s30
1: r34
2: ignore
> 0
```

```
Conflict No.9detected: for state 38, Symbol else, s39/r17
Please select action for this conflict:
0: s39
1: r17
2: ignore
0
Conflict No.10detected: for state 53, Symbol AND, s52/r23
Please select action for this conflict:
0: s52
1: r23
2: ignore
> 1
Conflict No.11detected: for state 53, Symbol OR, s54/r23
Please select action for this conflict:
0: s54
1: r23
2: ignore
> 1
Conflict No.12detected: for state 55, Symbol AND, s52/r24
Please select action for this conflict:
0: s52
1: r24
2: ignore
> 0
Conflict No.13detected: for state 55, Symbol OR, s54/r24
Please select action for this conflict:
0: s54
1: r24
2: ignore
> 1
```

3. 生成文法概况

增广文法共13个变元,22个终结符(包括结束符号#),生成的ItemDFA共84个状态,SLR(1)文法不能消解的冲突共12个,其中10个能用运算符优先级规则消解,1个能用最近else匹配规则消解,还有一个不能消解.(另有一个由程序设计上带来的实际上不存在的冲突,见下文)

4. SLR(1)所不能消解的冲突

共12个,如下:

```
10:
E -> E + E .
E -> E .+ E
E -> E .* E
```

此状态对终结符+和*都产生shift-reduce冲突,由于乘运算优先级较高,因而

- 对+采取reduce
- 对*采取shift

```
12:
E -> E * E .
E -> E .+ E
E -> E .* E
```

此状态对终结符+和*都产生shift-reduce冲突,由于乘运算优先级较高,因而
对+和*都采取reduce,如同第二项和第三项不存在.

```
15:
E -> d = E .
E -> E .+ E
E -> E .* E
```

此状态对终结符+和*都产生shift-reduce冲突,考虑到应当先对等式右侧表达式求值后再赋值给左侧表达式,因而
对+和*都采取shift.

```
29:
R -> d ( .)
E -> d ( .R' )
R' -> ._
R' -> .R' R ,
```

此状态对终结符)产生shift-reduce冲突,这是由于第三项目是空产生式,因而始终为完成项目,且)在R'的FOLLOW集中,可以执行reduce;但同时他又被第一项目接受执行shift,从而产生冲突.
在查阅资料后,我们认为此处的R和R'是在处理函数参数列表中的逗号表达式,为了允许空参数列表调用(并结合考虑现有分析器的做法),应在此选择shift.

```
38:
S -> if ( B ) S .
```

```
S -> if ( B ) S .else S
```

此状态对终结符**else**产生**shift-reduce**冲突.
这是典型的悬挂**else**冲突.采取最近匹配原则,即对**else**执行**shift**.

```
53:
B -> B AND B .
B -> B .AND B
B -> B .OR B

55:
B -> B OR B .
B -> B .AND B
B -> B .OR B
```

此二状态对终结符**AND**和**OR**产生**shift-reduce**冲突.
查阅资料发现,在**C++**等语言中,逻辑与**(&&,and)**优先级要高于逻辑或**(||,or)**,因而这里解决冲突的办法和上面解决*和+的办法是一样的.
在状态**53**中,对**AND**和**OR**都采取**reduce**;在状态**55**中,对**AND**采取**shift**,对**OR**采取**reduce**.

由于程序设计时的失误,会错误地对终结符#出现`acc/r0`冲突,但不难发现,他们实际上是等价的(对终结符#而言).由于这个冲突可以人工纠正,即选择*acc*作为结果,并且完全不会影响其他状态,因而不在程序中修改,以免产生更多错误.

6 词法分析

选择上一次大作业的样例代码作为输入串.作了一个修改:

1. 由于该文法不接受**print**,因而将**print**改为**return**
   词法分析程序见附件`cifa.cpp`.
   词法分析的结果储存在`codeinput.txt`中,其内容如下

```
int d ( int d ; ) { d = d + i ; return d } ; void d ( int d ; ) { int d ;
void d ( int d ; int d ( ) ; ) { if ( i AND d = i ) d ( i , d ( ) , ) else d
= d ( d , ) ; return d } ; d ( d , d ( ) , ) } ; d ( i , )
```

在文件中,该结果实际为一行.

7 SLR(1)模拟器

该部分读入产生的`trans.csv`和`gram.txt`作为语法部分的输入,读入`codeinput.txt`作为输入串输入.

SLR(1)模拟器代码见附件`DFASimu.cpp`.

其结论为接受,其分析过程如下,用截图展示:

temp

步骤	符号栈	输入	状态栈	ACTION	GOTO
1	D'	intd(intd;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2	R2	2
2	D'int	d(intd;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 73	S73	
3	D'T	d(intd;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61	R7	61
4	D'Td	(intd;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62	S62	
5	D'Td(	intd;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66	S66	
6	D'Td(A'	intd;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67	R9	67
7	D'Td(A'int	d;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 73	S73	
8	D'Td(A'T	d;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 77	R7	77
9	D'Td(A'Td	;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 77 78	S78	
10	D'Td(A'A	;){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 75	R11	75
11	D'Td(A'A;	){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 75 76	S76	
12	D'Td(A'	){d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67	R10	67
13	D'Td(A')	{d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68	S68	
14	D'Td(A'){	d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69	S69	
15	D'Td(A'){D'	d=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70	R2	70
16	D'Td(A'){D'd	=d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6	S6	
17	D'Td(A'){D'd=	d+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7	S7	
18	D'Td(A'){D'd=d	+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7 13	S13	
19	D'Td(A'){D'd=E	+i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7 8	R29	8
20	D'Td(A'){D'd=E+	i;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7 8 9	S9	
21	D'Td(A'){D'd=E+i	;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7 8 9 16	S16	
22	D'Td(A'){D'd=E+E	;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7 8 9 10	R28	10
23	D'Td(A'){D'd=E	;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 7 8	R31	8
24	D'Td(A'){D'S	;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 51	R16	51
25	D'Td(A'){D'S'	;returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71	R14	71
26	D'Td(A'){D'S';	returnd};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71 4	S4	
27	D'Td(A'){D'S';return	d};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71 4 46	S46	
28	D'Td(A'){D'S';returnd	};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71 4 46 13	S13	
29	D'Td(A'){D'S';returnE	};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71 4 46 47	R29	47
30	D'Td(A'){D'S';S	};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71 4 5	R20	5
31	D'Td(A'){D'S'	};voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71	R15	71
32	D'Td(A'){D'S'}	;voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 71 72	S72	
33	D'D	;voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 59	R6	59
34	D'D;	voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 59 60	S60	
35	D'	voidd(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2	R3	2
36	D'void	d(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 74	S74	
37	D'T	d(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61	R8	61
38	D'Td	(intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62	S62	
39	D'Td(	intd;){intd;voidd(intd;intd();)	0 2 61 62 66	S66	

步骤	符号栈	输入	状态栈	ACTION	GOTO
		{if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)			
40	D'Td(A'	intd;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67	R9	67
41	D'Td(A'int	d;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 73	S73	
42	D'Td(A'T	d;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 77	R7	77
43	D'Td(A'Td	;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 77 78	S78	
44	D'Td(A'A	;){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 75	R11	75
45	D'Td(A'A;	){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 75 76	S76	
46	D'Td(A'	){intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67	R10	67
47	D'Td(A')	{intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68	S68	
48	D'Td(A'){	intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69	S69	
49	D'Td(A'){D'	intd;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70	R2	70
50	D'Td(A'){D'int	d;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 73	S73	
51	D'Td(A'){D'T	d;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61	R7	61
52	D'Td(A'){D'Td	;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62	S62	
53	D'Td(A'){D'D	;voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 59	R4	59
54	D'Td(A'){D'D;	voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 59 60	S60	
55	D'Td(A'){D'	voidd(intd;intd();) {if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70	R3	70
56	D'Td(A'){D'void	d(intd;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 74	S74	
57	D'Td(A'){D'T	d(intd;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61	R8	61
58	D'Td(A'){D'Td	(intd;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62	S62	
59	D'Td(A'){D'Td(	intd;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66	S66	
60	D'Td(A'){D'Td(A'	intd;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67	R9	67
61	D'Td(A'){D'Td(A'int	d;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 73	S73	
62	D'Td(A'){D'Td(A'T	d;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 77	R7	77
63	D'Td(A'){D'Td(A'Td	;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 77 78	S78	
64	D'Td(A'){D'Td(A'A	;intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 75	R11	75
65	D'Td(A'){D'Td(A'A;	intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 75 76	S76	
66	D'Td(A'){D'Td(A'	intd();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67	R10	67
67	D'Td(A'){D'Td(A'int	d();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 73	S73	
68	D'Td(A'){D'Td(A'T	d();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 77	R7	77
69	D'Td(A'){D'Td(A'Td	();){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 77 78	S78	
70	D'Td(A'){D'Td(A'Td(	);){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 77 78 79	S79	
71	D'Td(A'){D'Td(A'Td()	;){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 77 78 79 80	S80	
72	D'Td(A'){D'Td(A'A	;){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 75	R13	75
73	D'Td(A'){D'Td(A'A;	){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 75 76	S76	
74	D'Td(A'){D'Td(A'	){if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67	R10	67
75	D'Td(A'){D'Td(A')	{if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68	S68	
76	D'Td(A'){D'Td(A'){	if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69	S69	
77	D'Td(A'){D'Td(A'){D'	if(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70	R2	70
78	D'Td(A'){D'Td(A'){D'if	(iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34	S34	

步骤	符号栈	输入	状态栈	ACTION	GOTO
79	D'Td(A'){D'Td(A'){D'if(	iANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35	S35	
80	D'Td(A'){D'Td(A'){D'if(i	ANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 16	S16	
81	D'Td(A'){D'Td(A'){D'if(E	ANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 56	R28	56
82	D'Td(A'){D'Td(A'){D'if(B	ANDd=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36	R26	36
83	D'Td(A'){D'Td(A') {D'if(BAND	d=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52	S52	
84	D'Td(A'){D'Td(A') {D'if(BANDd	=i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52 13	S13	
85	D'Td(A'){D'Td(A') {D'if(BANDd=	i)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52 13 14	S14	
86	D'Td(A'){D'Td(A') {D'if(BANDd=i	)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52 13 14 16	S16	
87	D'Td(A'){D'Td(A') {D'if(BANDd=E	)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52 13 14 15	R28	15
88	D'Td(A'){D'Td(A') {D'if(BANDE	)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52 56	R27	56
89	D'Td(A'){D'Td(A') {D'if(BANDB	)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 52 53	R26	53
90	D'Td(A'){D'Td(A'){D'if(B	)d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36	R23	36
91	D'Td(A'){D'Td(A'){D'if(B)	d(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37	S37	
92	D'Td(A'){D'Td(A') {D'if(B)d	(i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6	S6	
93	D'Td(A'){D'Td(A') {D'if(B)d(	i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31	S31	
94	D'Td(A'){D'Td(A') {D'if(B)d(R'	i,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32	R34	32
95	D'Td(A'){D'Td(A') {D'if(B)d(R'i	,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 16	S16	
96	D'Td(A'){D'Td(A') {D'if(B)d(R'E	,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 25	R28	25
97	D'Td(A'){D'Td(A') {D'if(B)d(R'R	,d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 23	R36	23
98	D'Td(A'){D'Td(A') {D'if(B)d(R'R,	d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 23 24	S24	
99	D'Td(A'){D'Td(A') {D'if(B)d(R'	d(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32	R35	32
100	D'Td(A'){D'Td(A') {D'if(B)d(R'd	(),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 26	S26	
101	D'Td(A'){D'Td(A') {D'if(B)d(R'd(	),)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 26 29	S29	
102	D'Td(A'){D'Td(A') {D'if(B)d(R'd()	,)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 26 29 30	S30	
103	D'Td(A'){D'Td(A') {D'if(B)d(R'R	,)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 23	R38	23
104	D'Td(A'){D'Td(A') {D'if(B)d(R'R,	)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32 23 24	S24	
105	D'Td(A'){D'Td(A') {D'if(B)d(R'	)elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 6 31 32	R35	32
106	D'Td(A'){D'Td(A') {D'if(B)d(R')	elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70	S33	

步骤	符号栈	输入	状态栈	ACTION	GOTO
			34 35 36 37 6 31 32 33		
107	D'Td(A'){D'Td(A') {D'if(B)S	elsed=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38	R22	38
108	D'Td(A'){D'Td(A') {D'if(B)Selse	d=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39	S39	
109	D'Td(A'){D'Td(A') {D'if(B)Selsed	=d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6	S6	
110	D'Td(A'){D'Td(A') {D'if(B)Selsed=	d(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7	S7	
111	D'Td(A'){D'Td(A') {D'if(B)Selsed=d	(d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13	S13	
112	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(	d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20	S20	
113	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R'	d,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21	R34	21
114	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R'd	,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21 26	S26	
115	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R'E	,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21 25	R29	25
116	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R'R	,);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21 23	R36	23
117	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R'R,	);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21 23 24	S24	
118	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R'	);returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21	R35	21
119	D'Td(A'){D'Td(A') {D'if(B)Selsed=d(R')	;returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 13 20 21 22	S22	
120	D'Td(A'){D'Td(A') {D'if(B)Selsed=E	;returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 6 7 8	R30	8
121	D'Td(A'){D'Td(A') {D'if(B)SelseS	;returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 34 35 36 37 38 39 40	R16	40
122	D'Td(A'){D'Td(A'){D'S	;returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 51	R18	51
123	D'Td(A'){D'Td(A'){D'S'	;returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71	R14	71
124	D'Td(A'){D'Td(A'){D'S';	returnd};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71 4	S4	
125	D'Td(A'){D'Td(A') {D'S';return	d};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71 4 46	S46	
126	D'Td(A'){D'Td(A') {D'S';returnd	};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71 4 46 13	S13	
127	D'Td(A'){D'Td(A') {D'S';returnE	};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71 4 46 47	R29	47
128	D'Td(A'){D'Td(A'){D'S';S	};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71 4 5	R20	5
129	D'Td(A'){D'Td(A'){D'S'	};d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71	R15	71
130	D'Td(A'){D'Td(A'){D'S'}	;d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 61 62 66 67 68 69 70 71 72	S72	
131	D'Td(A'){D'D	;d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 59	R6	59
132	D'Td(A'){D'D;	d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 59 60	S60	
133	D'Td(A'){D'	d(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70	R3	70

步骤	符号栈	输入	状态栈	ACTION	GOTO
134	D'Td(A'){D'd	(d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6	S6	
135	D'Td(A'){D'd(	d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31	S31	
136	D'Td(A'){D'd(R'	d,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32	R34	32
137	D'Td(A'){D'd(R'd	,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 26	S26	
138	D'Td(A'){D'd(R'E	,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 25	R29	25
139	D'Td(A'){D'd(R'R	,d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 23	R36	23
140	D'Td(A'){D'd(R'R,	d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 23 24	S24	
141	D'Td(A'){D'd(R'	d(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32	R35	32
142	D'Td(A'){D'd(R'd	(),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 26	S26	
143	D'Td(A'){D'd(R'd(	),)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 26 29	S29	
144	D'Td(A'){D'd(R'd()	,)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 26 29 30	S30	
145	D'Td(A'){D'd(R'R	,)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 23	R38	23
146	D'Td(A'){D'd(R'R,	)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 23 24	S24	
147	D'Td(A'){D'd(R'	)};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32	R35	32
148	D'Td(A'){D'd(R')	};d(i,)	0 2 61 62 66 67 68 69 70 6 31 32 33	S33	
149	D'Td(A'){D'S	};d(i,)	0 2 61 62 66 67 68 69 70 51	R22	51
150	D'Td(A'){D'S'	};d(i,)	0 2 61 62 66 67 68 69 70 71	R14	71
151	D'Td(A'){D'S'}	;d(i,)	0 2 61 62 66 67 68 69 70 71 72	S72	
152	D'D	;d(i,)	0 2 59	R6	59
153	D'D;	d(i,)	0 2 59 60	S60	
154	D'	d(i,)	0 2	R3	2
155	D'd	(i,)	0 2 6	S6	
156	D'd(	i,)	0 2 6 31	S31	
157	D'd(R'	i,)	0 2 6 31 32	R34	32
158	D'd(R'i	,)	0 2 6 31 32 16	S16	
159	D'd(R'E	,)	0 2 6 31 32 25	R28	25
160	D'd(R'R	,)	0 2 6 31 32 23	R36	23
161	D'd(R'R,	)	0 2 6 31 32 23 24	S24	
162	D'd(R'	)	0 2 6 31 32	R35	32
163	D'd(R')		0 2 6 31 32 33	S33	
164	D'S		0 2 51	R22	51
165	D'S'		0 2 3	R14	3
166	P		0 1	R1	1
167	P		0 1	acc	

接受
接受

8 附件

Generator.cpp:

```
#include<bits/stdc++.h>
using namespace std;

struct SingleGener
{
    string left;
    vector<string> right;
};
```

```
struct SingleGenerWithDot : public SingleGener
{
    int dotIndex; // Index of the dot in the right side:
    // 0 means before the first symbol, right.size() means after the last
symbol
    SingleGenerWithDot(string left, vector<string> right, int index) :
SingleGener{left, right}, dotIndex(index) {}

    bool operator==(const SingleGenerWithDot& other) const
    {
        return left == other.left && right == other.right && dotIndex ==
other.dotIndex; // Compare the left side, right side and dot index
    }

    friend bool operator==(const SingleGener& lhs, const SingleGenerWithDot&
rhs)
    {
        return lhs.left == rhs.left && lhs.right == rhs.right; // Compare
the left side, right side and dot index
    }
};

struct GenerTable
{
    vector<SingleGener> table; // Hash by its index in the vector
    // Guarantee index 0 to be argumented production
    unordered_map<string, vector<string>> first;
    unordered_map<string, vector<string>> follow;
    vector<string> nonTerminal;
    vector<string> terminal;
    string startSymbol;
    string ArgumentedStartSymbol; // Augmented grammar start symbol
    void ParseTableFromFile(ifstream& file)
    {
        string line;
        getline(file, line); // Read the first line for the start symbol
        stringstream ss(line);
        ss >> startSymbol; // Read the start symbol
        // Argumented Grammer
        ArgumentedStartSymbol = startSymbol + "'"; // Augmented grammar
start symbol
        table.push_back({ArgumentedStartSymbol, {startSymbol}}); // Add
augmented grammar production
        nonTerminal.push_back(ArgumentedStartSymbol); // Add to non-terminal
list
        while (getline(file, line))
        {
            if (line.empty()) continue; // Skip empty lines
            SingleGener gener;
            stringstream ss(line);
```

```
            ss >> gener.left; // Read the left side of the production
            if(find(nonTerminal.begin(), nonTerminal.end(), gener.left) ==
nonTerminal.end())
                nonTerminal.push_back(gener.left); // Add to non-terminal
list if not already present

            string rightPart;
            while (ss >> rightPart) // Read the right side of the production
            {
                if(rightPart == "_") break; // Empty production
                gener.right.push_back(rightPart);
            }
            table.push_back(gener);
        }

        for(auto&& item : table)
        {
            for(auto&& right : item.right)
            {
                if(find(nonTerminal.begin(), nonTerminal.end(), right) ==
nonTerminal.end() && find(terminal.begin(), terminal.end(), right) ==
terminal.end())
                {
                    terminal.push_back(right); // Add to terminal list if
not already present
                }
            }
        }
    }

    void ParseFollowFromFile(ifstream& file)
    {
        string Line;
        while(getline(file, Line))
        {
            stringstream ss(Line);
            string str;
            string temp;
            vector<string> right;
            ss >> str; // Left
            while(ss >> temp)
            {
                right.push_back(temp);
            }
            follow.insert({str, right});
        }

    }
};
```

```
struct ActionType
{
    enum Type
    {
        Uninit,
        Shift,
        Reduce,
        Accept,
    };
    Type type = Uninit; // Type of action
    int num;
    // For Shift, num is the state number to shift to
    // For Reduce, num is the production index to reduce by
    // For Goto, num is the state number to go to
    // For Accept, num is not used
};

struct ItemDFAState
{
    vector<SingleGenerWithDot> itemSet; // Set of items in this state

    bool operator==(const ItemDFAState& other) const
    {
        return itemSet == other.itemSet; // Compare the item sets
    }

    void print(ostream& os = cout)
    {
        os<<"Set Start:"<<endl;
        for(auto&& item : itemSet)
        {
            os << item.left << " -> ";
            if(item.right.size() == 0)
            {
                os << "._"<<endl;
                continue;
            }
            for(size_t i = 0; i < item.right.size(); ++i)
            {
                if(i == item.dotIndex) os << "."; // Print the dot at the
correct position
                os << item.right[i] << " ";
            }
            if(item.dotIndex == item.right.size()) os << "."; // Print the
dot at the end if needed
            os << endl;
        }
        os<<"Set End."<<endl;
    }
};
```

```
struct ItemDFA
{
    GenerTable& table; // Reference to the grammar table
    int SetCount; // Number of sets in the DFA
    vector<ItemDFAState> states; // States of the DFA
    vector<vector<vector<ActionType>>> ActionTable; // Action table for the
DFA, Y indexed by state number, X indexed by index of terminal symbol
    vector<vector<int>> GotoTable; // Goto table for the DFA, Y indexed by
state number, X indexed by index of non-terminal symbol

    void Closure(ItemDFAState& itemSet, vector<string>& nextSymbols)
    {
        vector<SingleGenerWithDot> temp = itemSet.itemSet;
        here:
        bool hasChanged = false;
        itemSet.itemSet = temp; // Update the item set with the new items
        for(auto&& item : itemSet.itemSet)
        {
            // If the dot is at the end of the production, skip it
            if(item.dotIndex == item.right.size()) continue;
            string nextSymbol = item.right[item.dotIndex];
            if(find(nextSymbols.begin(), nextSymbols.end(), nextSymbol) ==
nextSymbols.end())
            {
                nextSymbols.push_back(nextSymbol); // Add the next symbol to
the list
            }
            // If the next symbol is a non-terminal, find its productions
            if(find(table.nonTerminal.begin(), table.nonTerminal.end(),
nextSymbol) != table.nonTerminal.end())
            {
                for(auto&& gener : table.table)
                {
                    if(gener.left == nextSymbol)
                    {
                        // Create a new item with the dot at the beginning
of the production
                        SingleGenerWithDot newItem(gener.left, gener.right,
0);
                        // Check if the item is already in the set
                        if(find(temp.begin(), temp.end(), newItem) ==
temp.end())
                        {
                            temp.push_back(newItem);
                            hasChanged = true;
                        }
                    }
                }
            }
```

```
        }
        // If the item set has changed, repeat the closure process
        if(hasChanged)
        {
            goto here; // Repeat the closure process
        }
    }

    long int getIndexOfStates(ItemDFAState& itemSet)
    {
        for(size_t i = 0; i < states.size(); ++i)
        {
            if(states[i].itemSet == itemSet.itemSet) return i; // Return the
index of the state if found
        }
        return -1; // Return -1 if not found
    }

    void updateTable(long int current_state_index, string symbol,
ActionType&& action)
    {
        // Update the action table with the new action
        auto it = find(table.terminal.begin(), table.terminal.end(),
symbol);
        if(it == table.terminal.end()) return; // If the symbol is not found
in the terminal list, return
        auto index = distance(table.terminal.begin(), it); // Get the index
of the symbol in the terminal list
        auto&& Place = ActionTable.at(current_state_index).at(index); // Get
the action at the current state and symbol index
        // if(Place.type != ActionType::Uninit)
        // {
        //     // println("Conflict detected for state {} and symbol {}: now
{}, previous {}", current_state_index, symbol, (int)action.type,
(int)Place.type);
        //     cout<<"Conflict detected: for state "<<current_state_index
        //     <<", Symbol "<<symbol
        //     <<", Previous"<<(int)Place.type<<"_"<<Place.num
        //     <<", Now"<<(int)action.type<<"_"<<action.num
        //     <<endl;
        // }
        Place.push_back(action); // Update the action at the current state
and symbol index
    }

    void dfs(ItemDFAState itemSet,const vector<string>& nextSymbols)
    {
        vector<SingleGenerWithDot> temp = itemSet.itemSet; // Copy the
current item set
        long int currentStateIndex = getIndexOfStates(itemSet); // Get the
```

```
index of the current state
        for(auto&& nxtChar : nextSymbols)
        {
            // cout<<nxtChar<<endl;
            ItemDFAState newItemSet; // New item set for the new state
            for(auto&& item : temp)
            {
                // If the dot is at the end of the production, skip it
                if(item.dotIndex == item.right.size()) continue;
                // If the next symbol matches the symbol after the dot,
create a new item with the dot moved to the right
                if(item.right[item.dotIndex] == nxtChar)
                {
                    newItemSet.itemSet.emplace_back(item.left, item.right,
item.dotIndex + 1); // Add the new item to the new item set
                }
            }
            if(!newItemSet.itemSet.empty())
            {
                vector<string> newNextSymbols;
                Closure(newItemSet, newNextSymbols); // Perform closure on
the new state

                if(find(states.begin(), states.end(), newItemSet) ==
states.end())
                {
                    states.push_back(newItemSet); // Add the new state to
the DFA states
                    SetCount++; // Increment the set count
                    dfs(newItemSet, newNextSymbols); // Perform DFS on the
new state
                }
                long int newStateIndex = getIndexOfStates(newItemSet); //
Get the index of the new state
                if(newStateIndex != -1)
                {
                    // Update the action table with the new action
                    if(find(table.terminal.begin(), table.terminal.end(),
nxtChar) != table.terminal.end()) // terminal, shift
                    {
                        ActionType action;
                        action.type = ActionType::Shift; // Set the action
type to shift
                        action.num = newStateIndex; // Set the state number
to shift to
                        updateTable(currentStateIndex, nxtChar,
move(action)); // Update the action table
                    }
                    else if(find(table.nonTerminal.begin(),
table.nonTerminal.end(), nxtChar) != table.nonTerminal.end()) // non-
```

```
terminal, goto
                    {
                        auto it = find(table.nonTerminal.begin(),
table.nonTerminal.end(), nxtChar); // Find the index of the non-terminal
symbol
                        auto index = distance(table.nonTerminal.begin(),
it); // Get the index of the non-terminal symbol
                        GotoTable.at(currentStateIndex).at(index) =
newStateIndex; // Update the goto table with the new state index
                    }
                }
            }
        }
        // cout<<"---"<<endl;
        // Check accept state
        for(auto&& item : temp)
        {
            if(item.dotIndex == item.right.size() && item.left ==
table.ArgumentedStartSymbol) // If the item is an accept state
            {
                ActionType action;
                action.type = ActionType::Accept; // Set the action type to
accept
                action.num = -1; // Set to -1 for accept action

ActionTable.at(currentStateIndex).at(table.terminal.size()).push_back(move(a
ction)); // Update the action table for the accept state
            }
        }

        // Check reduce state
        for(auto&& item : temp)
        {
            if(item.dotIndex == item.right.size()) // If the item is a
reduce state
            {
                auto it = find(table.table.begin(), table.table.end(),
item); // Find the index of the item in the grammar table
                auto index = distance(table.table.begin(), it); // Get the
index of the item in the grammar table
                ActionType action;
                action.type = ActionType::Reduce; // Set the action type to
reduce
                action.num = index; // Set the production index to reduce by
                //updateTable(currentStateIndex, item.left, move(action));
// Update the action table for the reduce state
                auto& Form = table.follow.at(item.left);
                for(int i = 0; i <= table.terminal.size(); i++)
                {
                    auto&& c = (i == table.terminal.size() ? "#" :
```

```
table.terminal.at(i));
                    if(find(Form.begin(), Form.end(), c) != Form.end())
                    {
                        auto& Modi =
                            ActionTable.at(currentStateIndex).at(i);
                        // if(Modi.type != ActionType::Uninit)
                        // {
                        //     cout<<"Conflict detected: for state "
<<currentStateIndex
                        //         <<", Symbol "<<c
                        //         <<", Previous"<<(int)Modi.type<<"_"
<<Modi.num
                        //         <<", Now"<<(int)action.type<<"_"
<<action.num
                        //         <<endl;
                        // }
                        Modi.push_back(action); // Update the action table
for the reduce state
                    }
                }
            }
        }
    }

    void BuildDFA()
    {
        ItemDFAState startState;
        SingleGenerWithDot startItem(table.ArgumentedStartSymbol,
{table.startSymbol}, 0);
        startState.itemSet.push_back(startItem); // Add the start item to
the start state
        vector<string> nextSymbols; // List of next symbols to process
        Closure(startState, nextSymbols); // Perform closure on the start
state
        states.push_back(startState); // Add the start state to the DFA
states
        SetCount = 1; // Initialize the set count to 1
        ActionTable.resize(10000, vector<vector<ActionType>>
(table.terminal.size() + 1)); // Resize the action table to accommodate the
start state
        GotoTable.resize(10000, vector<int>(table.nonTerminal.size())); //
Resize the goto table to accommodate the start state
        startState.print(); // Print the start state
        dfs(startState, nextSymbols); // Perform DFS to build the DFA
    }
};

int main()
{
    GenerTable generTable;
```

```
    ifstream file("gram.txt");
    if (!file.is_open())
    {
        cerr << "Error opening file" << endl;
        return 1;
    }
    generTable.ParseTableFromFile(file);
    file.close();

    ifstream file2("follow.txt");
    if(! file2.is_open())
    {
        cerr << "Error opening file" << endl;
        return 1;
    }
    generTable.ParseFollowFromFile(file2);
    file2.close();

    ItemDFA itemDFA(generTable);
    itemDFA.BuildDFA(); // Build the DFA from the grammar table

    // println("DFA states count: {}", itemDFA.SetCount);
    cout << "DFA states count: " << itemDFA.SetCount << endl; // Output the
number of DFA states

    ofstream fout("trans.csv");
    fout<<setw(10)<<"@";
    for(auto&& item : generTable.terminal)
    {
        fout<<setw(10)<<item<<"@";
    }
    fout<<setw(10)<<"#"<<"@";
    for(auto&& item : generTable.nonTerminal)
    {
        fout<<setw(10)<<item<<"@";
    }
    fout<<endl;
    int ConflictCount = 0;
    for(int i = 0; i < itemDFA.SetCount; i++)
    {
        fout<<setw(10)<<i<<"@";
        for(int j = 0; j <= generTable.terminal.size(); j++)
        {
            string op;
            auto& item = itemDFA.ActionTable.at(i).at(j);
            for(auto&& item2 : item)
            {
                if(item2.type == ActionType::Shift) op += "s" +
to_string(item2.num); // Shift action
                else if(item2.type == ActionType::Reduce) op += "r" +
```

```
to_string(item2.num); // Reduce action
                else if(item2.type == ActionType::Accept) op += "acc"; //
Accept action
                else op = " "; // Uninitialized action
                if(&item2 != &item.back()) op += "/"; // If not the last
item, add a comma
            }
            // fout<<setw(10)<<op<<"@";
            if(item.size() >= 2)
            {
                cout<<"Conflict No." << (++ConflictCount )
                    <<"detected: for state "<<i
                    <<", Symbol "<<(j == generTable.terminal.size() ? "#" :
generTable.terminal.at(j))<<", "
                    <<op
                    <<endl;

                cout<<"Please select action for this conflict: "<<endl;
                for(int k = 0; k < item.size(); k++)
                {
                    cout<<k<<": ";
                    if(item.at(k).type == ActionType::Shift) cout<<"s" <<
item.at(k).num <<endl; // Shift action
                    else if(item.at(k).type == ActionType::Reduce) cout<<"r"
<< item.at(k).num <<endl; // Reduce action
                    else if(item.at(k).type == ActionType::Accept)
cout<<"acc"<<endl; // Accept action
                    else cout<<"Uninit"<<endl; // Uninitialized action
                }
                cout<<item.size()<<": ignore" <<endl;
                int choice;
                cin>>choice; // Get user input for the action to take
                if(choice == item.size()) goto here2;
                if(choice < 0 || choice >= item.size())
                {
                    cout<<"Invalid choice, please try again."<<endl;
                    j--;
                    continue; // If invalid choice, repeat the action for
the same symbol
                }
                auto& selectedAction = item.at(choice); // Get the selected
action
                if(selectedAction.type == ActionType::Shift) // Shift action
                {
                    op = "s" + to_string(selectedAction.num); // Set the
action to shift
                }
                else if(selectedAction.type == ActionType::Reduce) // Reduce
action
                {
```

```
                    op = "r" + to_string(selectedAction.num); // Set the
action to reduce
                }
                else if(selectedAction.type == ActionType::Accept) // Accept
action
                {
                    op = "acc"; // Set the action to accept
                }
            }
            else if(item.size() == 1)
            {
                auto& selectedAction = item.at(0); // Get the selected
action
                if(selectedAction.type == ActionType::Shift) // Shift action
                {
                    op = "s" + to_string(selectedAction.num); // Set the
action to shift
                }
                else if(selectedAction.type == ActionType::Reduce) // Reduce
action
                {
                    op = "r" + to_string(selectedAction.num); // Set the
action to reduce
                }
                else if(selectedAction.type == ActionType::Accept) // Accept
action
                {
                    op = "acc"; // Set the action to accept
                }
            }
            else op = " "; // Uninitialized action
            here2:
            fout<<setw(10)<<op<<"@"; // Output the action to the file
        }
        for(int j = 0; j < generTable.nonTerminal.size(); j++)
        {
            auto& item = itemDFA.GotoTable.at(i).at(j);
            if(item == 0) fout<<setw(10)<<"@";
            else fout<<setw(10)<<item<<"@";
        }
        fout<<endl;
    }

    ofstream fout2("res2.csv");
    for(int i = 0; i < itemDFA.SetCount; i++)
    {
        fout2<<i<<": "<<endl;
        itemDFA.states[i].print(fout2);
    }
    fout2.close();
```

```
    return 0;
}
```

此程序产生的结果是.csv格式的,需要手动导入Excel软件中,选择@为分隔符(因为文法中出现了逗号),才能得到附件中的结果.

gram.txt

```
P
P   D' S'
D'  _
D'  D' D ;
D   T  d
D   T  d [ i ]
D   T  d ( A' ) { D' S' }
T   int
T   void
A'  _
A'  A' A ;
A   T  d
A   d  [ ]
A   T  d ( )
S'  S
S'  S' ; S
S   d = E
S   if ( B ) S
S   if ( B ) S else S
S   while ( B ) S
S   return E
S   { S' }
S   d ( R' )
B   B AND B
B   B OR B
B   E r E
B   E
E   d = E
E   i
E   d
E   d ( R' )
E   E + E
E   E * E
E   ( E )
R'  _
```

```
R'  R' R ,
R   E
R   d [ ]
R   d ( )
```

follow.txt

```
P' #
P #
S' ; } #
D d if while return { int void #
D' d if while return { int void #
T d
A' d int void )
A d int void )
S ; } # else
B ) OR AND
E ; } # else ) + * d i r AND OR (
R' ) d i , (
R ) d i , (
```

生成的结果(状态表):

```
0:
Set Start:
P' -> .P
P -> .D' S'
D' -> ._
D' -> .D' D ;
Set End.
1:
Set Start:
P' -> P .
Set End.
2:
Set Start:
P -> D' .S'
D' -> D' .D ;
S' -> .S
S' -> .S' ; S
D -> .T d
D -> .T d [ i ]
D -> .T d ( A' ) { D' S' }
```

```
S -> .d = E
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
T -> .int
T -> .void
Set End.
3:
Set Start:
P -> D' S' .
S' -> S' .; S
Set End.
4:
Set Start:
S' -> S' ; .S
S -> .d = E
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
Set End.
5:
Set Start:
S' -> S' ; S .
Set End.
6:
Set Start:
S -> d .= E
S -> d .( R' )
Set End.
7:
Set Start:
S -> d = .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
```

```
E -> .E * E
E -> .( E )
Set End.
8:
Set Start:
S -> d = E .
E -> E .+ E
E -> E .* E
Set End.
9:
Set Start:
E -> E + .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
10:
Set Start:
E -> E + E .
E -> E .+ E
E -> E .* E
Set End.
11:
Set Start:
E -> E * .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
12:
Set Start:
E -> E * E .
E -> E .+ E
E -> E .* E
Set End.
```

```
13:
Set Start:
E -> d .= E
E -> d .
E -> d .( R' )
Set End.
14:
Set Start:
E -> d = .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
15:
Set Start:
E -> d = E .
E -> E .+ E
E -> E .* E
Set End.
16:
Set Start:
E -> i .
Set End.
17:
Set Start:
E -> ( .E )
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
18:
Set Start:
E -> ( E .)
E -> E .+ E
E -> E .* E
```

```
Set End.
19:
Set Start:
E -> ( E ) .
Set End.
20:
Set Start:
E -> d ( .R' )
R' -> ._
R' -> .R' R ,
Set End.
21:
Set Start:
E -> d ( R' .)
R' -> R' .R ,
R -> .E
R -> .d [ ]
R -> .d ( )
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
22:
Set Start:
E -> d ( R' ) .
Set End.
23:
Set Start:
R' -> R' R .,
Set End.
24:
Set Start:
R' -> R' R , .
Set End.
25:
Set Start:
R -> E .
E -> E .+ E
E -> E .* E
```

```
Set End.
26:
Set Start:
R -> d .[ ]
R -> d .( )
E -> d .= E
E -> d .
E -> d .( R' )
Set End.
27:
Set Start:
R -> d [ .]
Set End.
28:
Set Start:
R -> d [ ] .
Set End.
29:
Set Start:
R -> d ( .)
E -> d ( .R' )
R' -> ._
R' -> .R' R ,
Set End.
30:
Set Start:
R -> d ( ) .
Set End.
31:
Set Start:
S -> d ( .R' )
R' -> ._
R' -> .R' R ,
Set End.
32:
Set Start:
S -> d ( R' .)
R' -> R' .R ,
R -> .E
R -> .d [ ]
R -> .d ( )
E -> .d = E
E -> .i
```

```
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
33:
Set Start:
S -> d ( R' ) .
Set End.
34:
Set Start:
S -> if .( B ) S
S -> if .( B ) S else S
Set End.
35:
Set Start:
S -> if ( .B ) S
S -> if ( .B ) S else S
B -> .B AND B
B -> .B OR B
B -> .E r E
B -> .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
36:
Set Start:
S -> if ( B .) S
S -> if ( B .) S else S
B -> B .AND B
B -> B .OR B
Set End.
37:
Set Start:
S -> if ( B ) .S
S -> if ( B ) .S else S
S -> .d = E
```

```
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
Set End.
38:
Set Start:
S -> if ( B ) S .
S -> if ( B ) S .else S
Set End.
39:
Set Start:
S -> if ( B ) S else .S
S -> .d = E
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
Set End.
40:
Set Start:
S -> if ( B ) S else S .
Set End.
41:
Set Start:
S -> while .( B ) S
Set End.
42:
Set Start:
S -> while ( .B ) S
B -> .B AND B
B -> .B OR B
B -> .E r E
B -> .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
```

```
E -> .E * E
E -> .( E )
Set End.
43:
Set Start:
S -> while ( B .) S
B -> B .AND B
B -> B .OR B
Set End.
44:
Set Start:
S -> while ( B ) .S
S -> .d = E
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
Set End.
45:
Set Start:
S -> while ( B ) S .
Set End.
46:
Set Start:
S -> return .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
47:
Set Start:
S -> return E .
E -> E .+ E
E -> E .* E
Set End.
48:
Set Start:
```

```
S -> { .S' }
S' -> .S
S' -> .S' ; S
S -> .d = E
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
Set End.
49:
Set Start:
S -> { S' .}
S' -> S' .; S
Set End.
50:
Set Start:
S -> { S' } .
Set End.
51:
Set Start:
S' -> S .
Set End.
52:
Set Start:
B -> B AND .B
B -> .B AND B
B -> .B OR B
B -> .E r E
B -> .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
53:
Set Start:
B -> B AND B .
B -> B .AND B
```

```
B -> B .OR B
Set End.
54:
Set Start:
B -> B OR .B
B -> .B AND B
B -> .B OR B
B -> .E r E
B -> .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
55:
Set Start:
B -> B OR B .
B -> B .AND B
B -> B .OR B
Set End.
56:
Set Start:
B -> E .r E
B -> E .
E -> E .+ E
E -> E .* E
Set End.
57:
Set Start:
B -> E r .E
E -> .d = E
E -> .i
E -> .d
E -> .d ( R' )
E -> .E + E
E -> .E * E
E -> .( E )
Set End.
58:
Set Start:
```

```
B -> E r E .
E -> E .+ E
E -> E .* E
Set End.
59:
Set Start:
D' -> D' D .;
Set End.
60:
Set Start:
D' -> D' D ; .
Set End.
61:
Set Start:
D -> T .d
D -> T .d [ i ]
D -> T .d ( A' ) { D' S' }
Set End.
62:
Set Start:
D -> T d .
D -> T d .[ i ]
D -> T d .( A' ) { D' S' }
Set End.
63:
Set Start:
D -> T d [ .i ]
Set End.
64:
Set Start:
D -> T d [ i .]
Set End.
65:
Set Start:
D -> T d [ i ] .
Set End.
66:
Set Start:
D -> T d ( .A' ) { D' S' }
A' -> ._
A' -> .A' A ;
Set End.
67:
```

```
Set Start:
D -> T d ( A' .) { D' S' }
A' -> A' .A ;
A -> .T d
A -> .d [ ]
A -> .T d ( )
T -> .int
T -> .void
Set End.
68:
Set Start:
D -> T d ( A' ) .{ D' S' }
Set End.
69:
Set Start:
D -> T d ( A' ) { .D' S' }
D' -> ._
D' -> .D' D ;
Set End.
70:
Set Start:
D -> T d ( A' ) { D' .S' }
D' -> D' .D ;
S' -> .S
S' -> .S' ; S
D -> .T d
D -> .T d [ i ]
D -> .T d ( A' ) { D' S' }
S -> .d = E
S -> .if ( B ) S
S -> .if ( B ) S else S
S -> .while ( B ) S
S -> .return E
S -> .{ S' }
S -> .d ( R' )
T -> .int
T -> .void
Set End.
71:
Set Start:
D -> T d ( A' ) { D' S' .}
S' -> S' .; S
Set End.
```

```
72:
Set Start:
D -> T d ( A' ) { D' S' } .
Set End.
73:
Set Start:
T -> int .
Set End.
74:
Set Start:
T -> void .
Set End.
75:
Set Start:
A' -> A' A .;
Set End.
76:
Set Start:
A' -> A' A ; .
Set End.
77:
Set Start:
A -> T .d
A -> T .d ( )
Set End.
78:
Set Start:
A -> T d .
A -> T d .( )
Set End.
79:
Set Start:
A -> T d ( .)
Set End.
80:
Set Start:
A -> T d ( ) .
Set End.
81:
Set Start:
A -> d .[ ]
Set End.
82:
```

```
Set Start:
A -> d [ .]
Set End.
83:
Set Start:
A -> d [ ] .
Set End.
```

转移表`trans.csv`(`trans.xlsx`)太大不便在此列出,请参考邮件附件.

DFASimu.cpp

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <stack>
#include <map>
#include <sstream>
#include <fstream>
#include <algorithm>

using namespace std;

// 定义产生式结构体，表示文法规则
struct Production {
    string left;  // 产生式的左部（非终结符）
    vector<string> right;  // 产生式的右部（符号序列）
};

// 定义Action结构体，用来表示移进、规约、转移等操作
struct Action {
    enum Type { SHIFT, REDUCE, GOTO, ACC, ERROR } type;  // 动作类型
    int value;  // 关联的状态编号或规约规则的编号
    Action() : type(ERROR), value(-1) {}  // 默认构造函数，表示无效操作
};

vector<Production> productions;  // 存储文法规则
vector<string> terminals;        // 存储终结符
vector<string> non_terminals;    // 存储非终结符
map<int, map<string, Action>> action_table;  // 存储状态和符号对应的动作（移进、规约等）
map<int, map<string, int>> goto_table;       // 存储状态和非终结符对应的转移状态

// 分割字符串函数，按指定分隔符拆分字符串，去掉空格并返回分割后的字符串向量
vector<string> split(const string &s, char delimiter) {
    vector<string> tokens;
    string token;
    istringstream tokenStream(s);
```

```
    while (getline(tokenStream, token, delimiter)) {
        // 去掉每个单元格中的空格
        token.erase(remove_if(token.begin(), token.end(), ::isspace),
token.end());
        if (!token.empty()) tokens.push_back(token);
    }
    return tokens;
}

// 检查符号类型，返回0表示是终结符，返回1表示是非终结符，返回-1表示未知符号
int check_symbol_type(const string& symbol) {
    if (find(terminals.begin(), terminals.end(), symbol) != terminals.end())
{
        return 0;  // 是终结符
    }
    if (find(non_terminals.begin(), non_terminals.end(), symbol) !=
non_terminals.end()) {
        return 1;  // 是非终结符
    }
    return -1;  // 既不是终结符也不是非终结符
}

// 解析动作字符串，将动作字符串解析为对应的 Action 对象
Action parse_action(const string &s) {
    Action a;
    if (s.empty()) return a;
    if (s == "acc") {
        a.type = Action::ACC;
    } else if (s[0] == 's') {
        a.type = Action::SHIFT;
        a.value = stoi(s.substr(1));  // 获取状态编号
    } else if (s[0] == 'r') {
        a.type = Action::REDUCE;
        a.value = stoi(s.substr(1));  // 获取规约规则编号
    } else if (isdigit(s[0])) {
        a.type = Action::GOTO;
        a.value = stoi(s);  // 获取 GOTO 状态编号
    }
    return a;
}

// 去除字符串中的空格，并按分隔符拆分字符串
vector<string> rm_space(const string &s, char delimiter) {
    vector<string> tokens;
    string token;
    istringstream tokenStream(s);
    while (getline(tokenStream, token, delimiter)) {
        token.erase(remove_if(token.begin(), token.end(), ::isspace),
token.end());  // 去掉空格
        tokens.push_back(token);  // 即使是空格也保留为空字符串
```

```
    }
    return tokens;
}

// 从 CSV 文件读取数据并转换为二维字符串向量
vector<vector<string>> read_csv_to_vector(const string& filename) {
    ifstream file(filename);  // 打开文件
    string line;
    vector<vector<string>> result;

    if (!file.is_open()) {  // 文件打开失败
        cerr << "Error: Could not open file " << filename << endl;
        return result;
    }

    // 逐行读取文件
    while (getline(file, line)) {
        vector<string> row;
        vector<string> tokens = rm_space(line, '@');  // 使用 '@' 符号分割列

        // 将每个单元格加入当前行
        for (const auto& token : tokens) {
            row.push_back(token);
        }

        // 将当前行加入到二维向量
        result.push_back(row);
    }

    file.close();  // 关闭文件
    return result;
}

// 读取文法文件并解析成生产式
void read_grammar(const string &filename) {
    ifstream file(filename);
    string line;
    while (getline(file, line)) {
        vector<string> parts = split(line, ' ');
        if (parts.empty()) continue;
        Production prod;
        prod.left = parts[0];  // 产生式的左部
        for (size_t i = 1; i < parts.size(); ++i) {
            if (parts[i] == "_") continue;  // 跳过空符号
            prod.right.push_back(parts[i]);  // 产生式的右部
        }
        productions.push_back(prod);  // 将生产式加入列表
    }
}
```

```
// 读取分析表并填充 action_table 和 goto_table
void read_trans_table(const string &filename) {
    // 使用二维vector读取文件
    vector<vector<string>> table = read_csv_to_vector(filename);

    // 如果解析失败，返回
    if (table.empty()) {
        cerr << "Error: Failed to read the CSV file." << endl;
        return;
    }

    // 第一行是符号行，分为终结符和非终结符
    vector<string> symbols = table[0];
    size_t split_pos = find(symbols.begin(), symbols.end(), "#") -
symbols.begin() + 1;

    terminals = vector<string>(symbols.begin(), symbols.begin() +
split_pos);  // 提取终结符
    non_terminals = vector<string>(symbols.begin() + split_pos,
symbols.end());  // 提取非终结符

    // 解析每一行状态数据
    for (size_t row = 1; row < table.size(); ++row) {  // 从第二行开始是状态数据
        vector<string> tokens = table[row];
        if (tokens.empty()) continue;

        int state = stoi(tokens[0]);  // 当前状态编号
        for (int i = 1; i < tokens.size(); i++) {  // 解析每一列的动作
            if (tokens[i] != "") {
                Action a = parse_action(tokens[i]);
                if (a.type != Action::ERROR) {
                    if (i <= 23) {  // 前24列是终结符的动作
                        action_table[state][terminals[i]] = a;
                    } else {  // 后面的列是非终结符的转移状态
                        goto_table[state][non_terminals[i - 24]] =
stoi(tokens[i]);
                    }
                }
            }
        }
    }
}

// 打印每一步解析过程
void print_step(int step, const stack<string> &sym_stack, const stack<int>
&state_stack,
                const vector<string> &input, int pos, const string &action,
int goto_state = -1) {
    cout << step << "\t| ";
    stack<string> sym_tmp = sym_stack;
```

```
    vector<string> syms;
    while (!sym_tmp.empty()) {
        syms.push_back(sym_tmp.top());
        sym_tmp.pop();
    }
    reverse(syms.begin(), syms.end());
    for (const auto &s : syms) cout << s;
    cout << "\t| ";
    for (size_t i = pos; i < input.size(); ++i) cout << input[i];
    cout << "\t| ";
    stack<int> state_tmp = state_stack;
    vector<int> states;
    while (!state_tmp.empty()) {
        states.push_back(state_tmp.top());
        state_tmp.pop();
    }
    reverse(states.begin(), states.end());
    for (int s : states) cout << s << " ";
    cout << "\t| " << action;
    if (goto_state != -1) cout << "\t| " << goto_state;
    else cout << "\t| ";
    cout << endl;
}

// 解析函数，执行移进、规约等操作
bool parse(const vector<string> &input) {
    stack<string> sym_stack;
    stack<int> state_stack;
    state_stack.push(0);
    int pos = 0, step = 1;

    cout << "步骤\t| 符号栈\t| 输入\t| 状态栈\t| ACTION\t| GOTO" << endl;
    cout << "---|---|---|---|---|---" << endl;

    while (true) {
        int state = state_stack.top();
        string sym = pos < input.size() ? input[pos] : "#";  // 获取当前符号

        if (find(terminals.begin(), terminals.end(), sym) ==
terminals.end()) {
            cerr << "错误：未知符号 " << sym << endl;
            return false;
        }

        if (action_table[state].find(sym) == action_table[state].end()) {
            cerr << sym << endl;
            cerr << "错误：状态 " << state << " 无动作" << endl;
            return false;
        }
```

```
        Action a = action_table[state][sym];
        string action_str;

        // 移进操作
        if (a.type == Action::SHIFT) {
            sym_stack.push(sym);
            state_stack.push(a.value);
            action_str = "S" + to_string(a.value);
            pos++;
            print_step(step++, sym_stack, state_stack, input, pos,
action_str);
        }
        else if (a.type == Action::REDUCE)
        {   // 规约操作
            Production &prod = productions[a.value];
            int rhs_len = prod.right.size();
            for (int i = 0; i < rhs_len; ++i) {
                if (sym_stack.empty())
                {
                    cerr << "错误：符号栈为空" << endl;
                    return false;
                }
                sym_stack.pop();
                state_stack.pop();
            }
            sym_stack.push(prod.left);
            int new_state = state_stack.top();
            int goto_state = goto_table[new_state][prod.left];
            state_stack.push(goto_state);
            action_str = "R" + to_string(a.value);
            print_step(step++, sym_stack, state_stack, input, pos,
action_str, goto_state);
        }
        else if (a.type == Action::ACC)
        {  // 接受操作
            print_step(step++, sym_stack, state_stack, input, pos, "acc");
            cout << "接受" << endl;
            return true;
        }
        else
        {
            cerr << "错误：无效动作" << endl;
            return false;
        }
    }
}

int main() {
    read_grammar("gram.txt");  // 读取文法
    read_trans_table("trans.csv");  // 读取分析表
```

```
    // 示例输入：d = i #
    // vector<string> input = {"d", "=", "i", "#"};
    vector<string> input;
    ifstream file("codeinput.txt");
    if (!file.is_open()) {
        cerr << "Error opening input file" << endl;
        return 1;
    }
    string line;
    getline(file, line);
    stringstream ss(line);
    while(ss >> line) {
        input.push_back(line);  // 将输入的符号加入到输入向量中
    }
    bool result = parse(input);  // 解析输入
    cout << (result ? "接受" : "拒绝") << endl;
    return 0;
}
```

cifa.cpp

```
#include<iostream>
#include<fstream>
#include<cstring>
#include<map>

using namespace std;

string answer = "";
map<string, string> match1 = {
        {"int","int"},{"void","void"},{"if","if"},{"else","else"},
        {"return","return"},{"print","print"}
};
map<char,string> match2 = {
        {'(',"("},{')',")"},{'[',"["},{']',"]"},{'{',"{"},{'}',"}"},
        {';',";"},{',',","},{'=',"="},{'/',"DIV"},{'&',"AND"},{'|',"OR"},
{'+',"+"}
};

void fail(){
        cout<<"error: invalid input"<<endl;
}

void end(string a, string b){
        string c = " " + b;
        answer += c;
}
```

```
bool case0(int &flag, string &temp,char ch){
        temp = "";
        if(ch >= '0' && ch <= '9'){
                temp += ch;
                flag = 1;
        }
        else if((ch>='a'&&ch<='z')||(ch >= 'A'&&ch<='Z')){
                flag = 2;
                temp += ch;
        }
        else{
                auto it = match2.find(ch);
                if(it != match2.end()){
                        string a = " " + it->second;
                        answer += a;
                        flag = 0;
                }
                else if( ch == '<'||ch == '>' ){
                        flag = 3;
                        temp += ch;
                }
                else if( ch == ' '|| ch =='\n'|| ch == '\r' || ch == '\t'){
                        flag = 0;
                }
                else{
                        return false;
                }
        }
        return true;
}

bool scanner(ifstream &file){
        char ch;
        int flag = 0;
        string temp = "";
        cout<<"the original code is:"<<endl;
        while(file.get(ch)){
                cout<<ch;
                switch (flag){
                        case 0:
                                if(!case0(flag,temp,ch)) return false;
                                break;
                        case 1:
                                if(ch>='0'&&ch<='9') temp += ch;
                                else {
                                        end(temp,"i");
                                        if(!case0(flag,temp,ch)) return
false;
                                }
                                break;
```

```
                        case 2:
                                if((ch>='a'&&ch<='z')||(ch >=
'A'&&ch<='Z')||(ch>='0'&&ch<='9')) temp += ch;
                                else{
                                        auto it = match1.find(temp);
                                        if(it != match1.end()) end("_", it-
>second);
                                        else end(temp, "d");
                                        if(!case0(flag,temp,ch)) return
false;
                                }
                                break;
                        case 3:
                                if(ch == '=') {
                                        temp += ch;
                                        end(temp, "ROP");
                                }
                                else{
                                        end(temp, "ROP");
                                        if(!case0(flag,temp,ch)) return
false;
                                }
                                break;
                }
        }
        return true;
}


int main () {
        ifstream file("code.txt");
        if(!file.is_open()){
                cout<<"error: cannot open the file"<<endl;
                return 1;
        }
        if(!scanner(file)) return 1;
        cout<<endl<<"the lexical analysis result is:"<<endl;
        cout<<answer<<endl;
}
```

词法分析器的输入如下：

```
int raw(int x;) {
        y=x+5;
        return y};
void foo(int y;) {
        int z;
```

```
        void bar(int x; int soo();) {
        if(3&z=1) bar(3, soo(),)
        else z = soo(x,);
        return z};
    bar(y, raw(),)};
foo(6,)
```