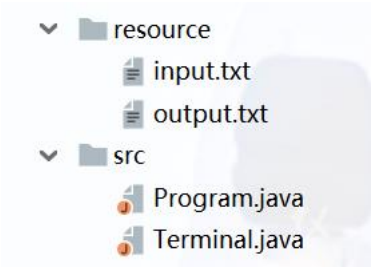
编译原理第二次实验

171250013 常卓

源文件目录截图



产生式资源文件截图

1	P -> C{F}	20	E'' -> = NUM
2	C -> M CLASS ID	21	E'' -> ε
3	C -> CLASS ID	22	E''' -> =E''''
4	F -> MF	23	E''' -> > ID
5	F -> ε	24	E''' -> < ID
6	F -> T ID (B) {E}	25	E'''' -> NUM
7	M -> PUBLIC	26	E'''' -> ID E'''''
8	M -> STATIC	27	E''''' -> +NUM
9	T -> VOID	28	E''''' -> ε
10	T -> STRING	29	E -> ID H
11	T -> INT	30	E -> T ID E'' ; E
12	B -> TB'	31	E -> FOR(E'; E'; E') { E } E
13	B' -> [] ID B''	32	E -> ε
14	B' -> ID B''	33	H -> E'''; E
15	B'' -> ,B	34	H -> F'' (B') ; E
16	B'' -> ε	35	F' -> ID F''
17	E' -> T ID E''	36	F'' -> .F'
18	E' -> ID E'''	37	F'' -> ε
19	E' -> ε	38	B' -> LITERAL

输入文件/流内容截图

```
<PUBLIC>
<CLASS>
<ID, Input>
<BRACKET, LCU>
<PUBLIC>
<STATIC>
<VOID>
<ID, main>
<BRACKET, LRO>
<STRING>
<BRACKET, LSQ>
<BRACKET, RSQ>
<ID, args>
<BRACKET, RRO>
<BRACKET, LCU>
<INT>
<ID, n>
<ASSIGN_OP>
<NUM, 100>
<SEMICOLON>
<FOR>
<BRACKET, LRO>
<INT>
<ID, i>
<ASSIGN_OP>
<NUM, 0>
<SEMICOLON>
<ID, i>
<RELOP, LT>
<ID, n>
<SEMICOLON>
<ID, i>
<ASSIGN_OP>
<ID, i>
<RELOP, ADD>
<NUM, 1>
<BRACKET, RRO>
<BRACKET, LCU>
<ID, System>
<DOT>
<ID, out>
<DOT>
<ID, println>
<BRACKET, LRO>
<LITERAL, Hello World!>
<BRACKET, RRO>
<SEMICOLON>
<BRACKET, RCU>
<BRACKET, RCU>
<BRACKET, RCU>
```

输出推导序列截图

```
1   P -> C { F }
2   C -> M CLASS ID
3   M -> PUBLIC
4   F -> M F
5   M -> PUBLIC
6   F -> M F
7   M -> STATIC
8   F -> T ID ( B ) { E }
9   T -> VOID
10  B -> T B'
11  T -> STRING
12  B' -> [ ] ID B''
13  B'' -> ε
14  E -> T ID E'' ; E
15  T -> INT
16  E'' -> = NUM
17  E -> FOR ( E' ; E' ; E' ) { E } E
18  E' -> T ID E''
19  T -> INT
20  E'' -> = NUM
21  E' -> ID E'''
22  E''' -> < ID
23  E' -> ID E'''
24  E''' -> = E''''
25  E'''' -> ID E'''''
26  E''''' -> + NUM
27  E -> ID H
28  H -> F'' ( B' ) ; E
29  F'' -> . F'
30  F' -> ID F''
31  F'' -> . F'
32  F' -> ID F''
33  F'' -> ε
34  B' -> LITERAL
35  E -> ε
36  E -> ε
```

正文

1. Motivation/Aim

通过构造语法分析程序，输出推导序列，对程序的语法正确性进行验证。

2. Content description

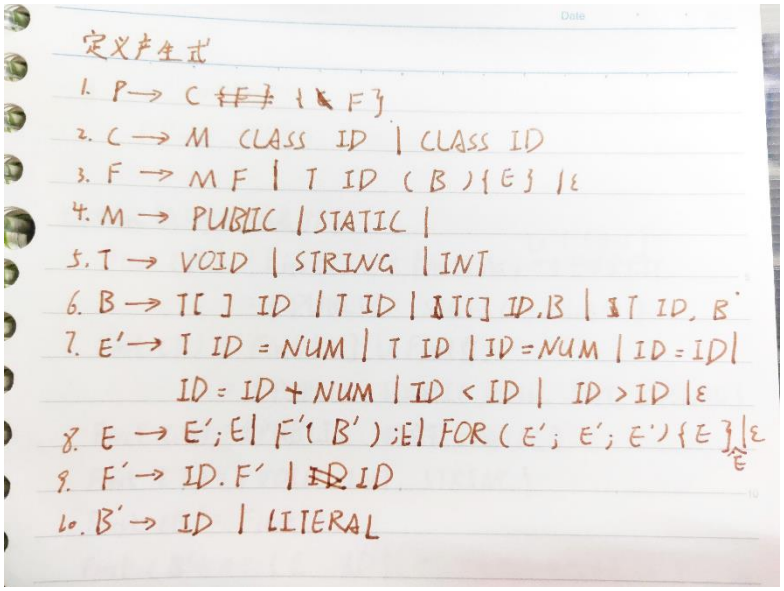
本实验使用 LL(1)分析法对第一次实验产生的 TOKEN 序列进行分析验证，最终输出推导序列。首先自定义出程序的产生式，然后对产生式中的最大公共左因子进行提取，消除左递归后求出非终结符的 First，求产生空的生成式求 Follow，填表，根据产生的预测分析表进行推导，得到推导序列。

3. Description of important Data Structures

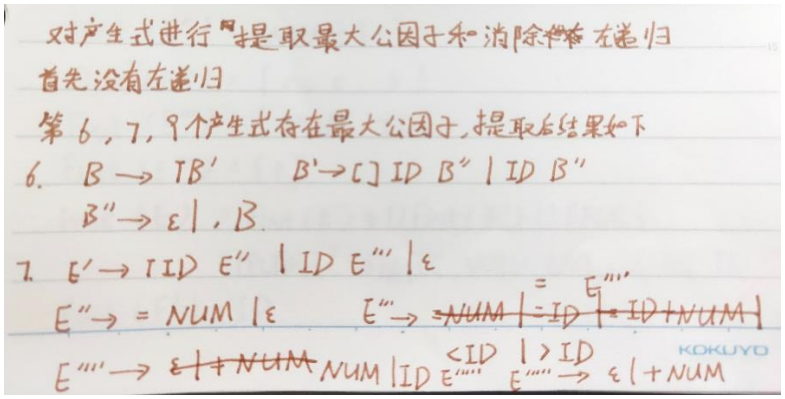
使用二维数组保存预测分析表，使用 map 保存不同非终结符和终结符对应的序号

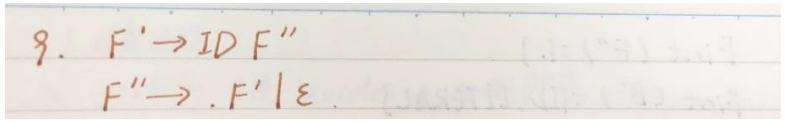
4. Description of core Algorithms

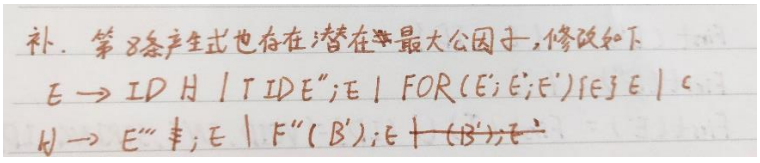
a. 首先定义出产生式（其中 P 代表程序，C 代表类定义，F 代表函数，E 代表表达式，B 代表参数序列，T 代表类型）



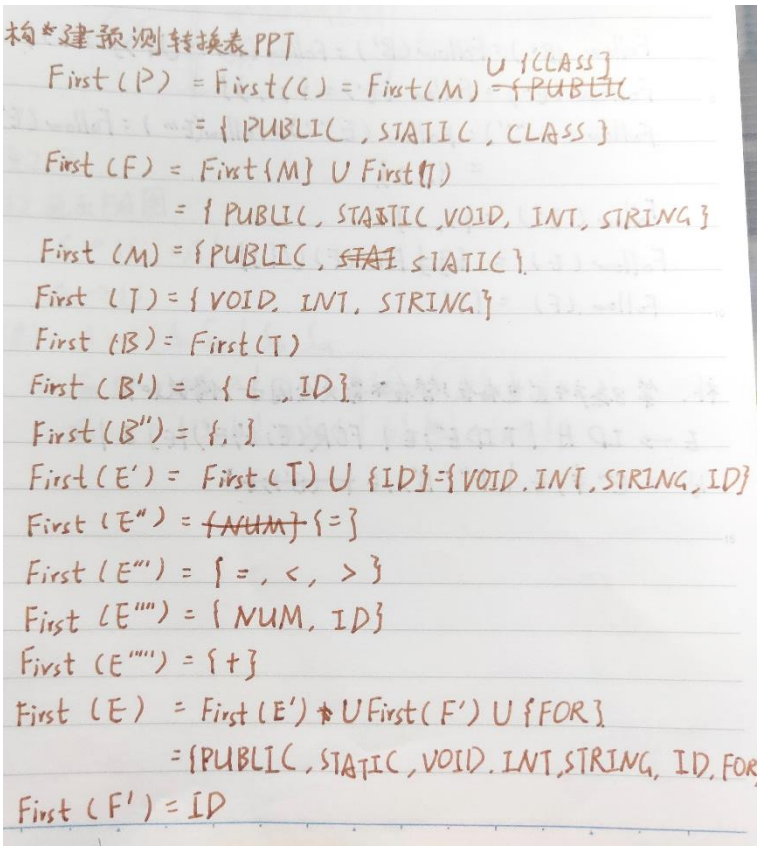
b. 消除左递归 & 提取最大公共左因子

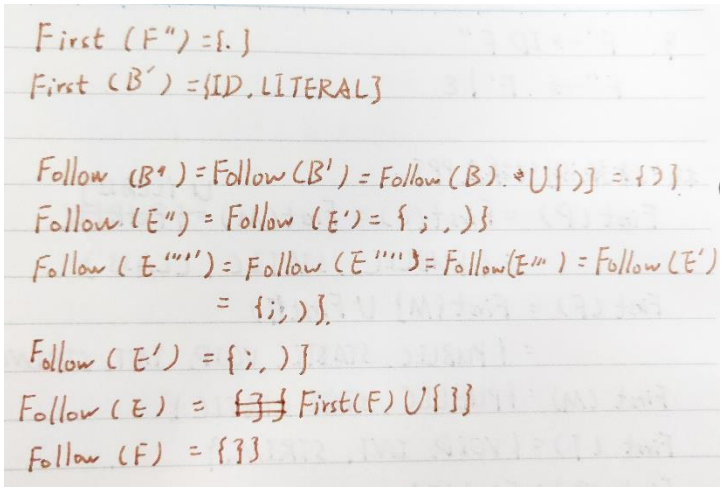






c. 求出 First、Follow





d. 构造预测分析表 PPT（此处对产生式进行了拆分和标号，产生式与标号的对应如下）

	PUBLIC	STATIC	CLASS	VOID	INT	STRING	[	ID	,	NUM	=	<	>	+	FOR	.	LITERAL	(	)	;	}	$
P	1	1	1																			
C	2	2	3																			
F	4	4		6	6	6															5	
F'								35														
M	7	8																				
T				9	11	10																
B				12	12	12																
B'							13	14									39					
B''									15										16			
E				30	30	30		29							31						32	
E'				17	17	17		18											19	19		
E''											20								21	21		
E'''											22	24	23									
E''''								26		25												
E'''''														27					28	28		
F''								35								36		37				
H								34			33	33	33			34						

1	P -> C{F}	20	E'' -> = NUM
2	C -> M CLASS ID	21	E'' -> ε
3	C -> CLASS ID	22	E''' -> =E''''
4	F -> MF	23	E''' -> > ID
5	F -> ε	24	E''' -> < ID
6	F -> T ID (B) {E}	25	E'''' -> NUM
7	M -> PUBLIC	26	E'''' -> ID E'''''
8	M -> STATIC	27	E''''' -> +NUM
9	T -> VOID	28	E''''' -> ε
10	T -> STRING	29	E -> ID H
11	T -> INT	30	E -> T ID E'' ; E
12	B -> TB'	31	E -> FOR(E'; E'; E') { E } E
13	B' -> [] ID B''	32	E -> ε
14	B' -> ID B''	33	H -> E'''; E
15	B'' -> ,B	34	H -> F'' (B') ; E
16	B'' -> ε	35	F' -> ID F''
17	E' -> T ID E''	36	F'' -> .F'
18	E' -> ID E'''	37	F'' -> ε
19	E' -> ε	38	B' -> LITERAL

e. 根据 PPT 编写程序，得到推导序列

5. Use cases on running

输入 TOKEN 序列

```
1  <PUBLIC>
2  <CLASS>
3  <ID, Input>
4  <BRACKET, LCU>
5  <PUBLIC>
6  <STATIC>
7  <VOID>
8  <ID, main>
9  <BRACKET, LRO>
10 <STRING>
11 <BRACKET, LSQ>
12 <BRACKET, RSQ>
13 <ID, args>
14 <BRACKET, RRO>
15 <BRACKET, LCU>
16 <INT>
17 <ID, n>
18 <ASSIGN_OP>
19 <NUM, 100>
20 <SEMICOLON>
21 <FOR>
22 <BRACKET, LRO>
23 <INT>
24 <ID, i>
25 <ASSIGN_OP>
26 <NUM, 0>
27 <SEMICOLON>
28 <ID, i>
29 <RELOP, LT>
30 <ID, n>
31 <SEMICOLON>
32 <ID, i>
33 <ASSIGN_OP>
34 <ID, i>
35 <RELOP, ADD>
36 <NUM, 1>
37 <BRACKET, RRO>
38 <BRACKET, LCU>
39 <ID, System>
40 <DOT>
41 <ID, out>
42 <DOT>
43 <ID, println>
44 <BRACKET, LRO>
45 <LITERAL, Hello World!>
46 <BRACKET, RRO>
47 <SEMICOLON>
48 <BRACKET, RCU>
49 <BRACKET, RCU>
50 <BRACKET, RCU>
```

输出的推导序列

```
P -> C { F }
C -> M CLASS ID
M -> PUBLIC
F -> M F
M -> PUBLIC
F -> M F
M -> STATIC
F -> T ID ( B ) { E }
T -> VOID
B -> T B'
T -> STRING
B' -> [ ] ID B''
B'' -> ε
E -> T ID E'' ; E
T -> INT
E'' -> = NUM
E -> FOR ( E' ; E' ; E' ) { E } E
E' -> T ID E''
T -> INT
E'' -> = NUM
E' -> ID E'''
E''' -> < ID
E' -> ID E'''
E''' -> = E''''
E'''' -> ID E'''''
E''''' -> + NUM
E -> ID H
H -> F'' ( B' ) ; E
F'' -> . F'
F' -> ID F''
F'' -> . F'
F' -> ID F''
F'' -> ε
B' -> LITERAL
E -> ε
E -> ε
```

6. 错误处理

在遇到 1.没有记录的 TOKEN，2.查表为空，3.指针指向终结符和栈顶的终结符不同，三种情况时，程序会打印出报错。

7. Problems occurred and related solutions

本次实验遇到的最大的问题在于如何用数据结构保存 PPT 表，并把产生式右边的非终结符/终结符从右向左压入栈内。最开始的想法是使用一个二维矩阵，每个项都是一个列表，保存待压入栈的非终结符/终结符，但由于 PPT 表是一个稀疏的矩阵，而产生式的右边也大小不定，所以这个方法会浪费大量内存，初始化也很困难。最后采用了两种数据结构共同解决了这个问题，首先用一个整型的矩阵保存产生式的编号，然后用 map 保存编号到产生式的对应关系，最后分析产生式字符串，逐个压栈。

另一个问题是需不需要区分非终结符和终结符，因为都需要压栈，所以应该是同一种数据类型，但是两种的处理方法完全不同，需要加以区分。最后使用了同一个类来表示，但用一个布尔值来区分两种，并用两个不同的 map 来索引。

8. My feelings and comments

通过这次实验，我深入了解了 LL(1)的分析过程，并且尝试了自己构造一组可以分析完整程序的生成式。由于真正的程序语法比较复杂，所以我简化了可能的程序结构，在适用于本程序的情况下使之尽可能适用于更多的程序。另一方面，在进行提取最大公共左因子时，由于没有深入分析每一个生成式，所以最后出现了遗漏的情况，在编程结束后报错才发现，然后进行了修改。

感想就是分析真正的程序的工作量还是很大的，即使是一个简化的版本也需要超出预期的工作。