

哈尔滨工业大学(深圳)

《编译原理》实验报告

学 院: 计算机科学与技术

姓 名: 郭毅安

学 号: 190111026

专 业: 计算机科学与技术

日 期: 2021-11-26

1 实验目的与方法

实验目的：

1.1 词法分析器

- (1) 加深对词法分析程序的功能及实现方法的理解；
- (2) 对类 C 语言的文法描述有更深入的认识，理解有穷自动机、编码表和符号表在编译的整个过程中的应用；
- (3) 设计并编程实现一个词法分析程序，对类 C 语言源程序段进行词法分析，加深对高级语言的认识。

1.2 语法分析

- (1) 深入了解语法分析程序实现原理及方法。
- (2) 理解 LR(1)分析法是严格的从左向右扫描和自底向上的语法分析方法。

1.3 典型语句的语义分析及中间代码生成

- (1) 加深对自顶向下语法制导翻译技术的理解与掌握。
- (2) 加深对自底向上语法制导翻译技术的理解与掌握。
- (3) 巩固对语义分析的基本功能和原理的认识，理解中间代码生成的作用。

1.4 目标代码生成

- (1) 加深对编译器总体结构的理解与掌握；
- (2) 加深对汇编指令的理解与掌握；
- (3) 对指令选择，寄存器分配和计算顺序选择有较深的理解

实验方法：

语言： C++

软件： Visual Studio 2019

2 实验内容及要求

2.1 词法分析器

编写一个词法分析程序，读取代码文件，对文件内自定义的类 C 语言程序段进行词法分析。处理 C 语言源程序，过滤掉无用符号，分解出正确的单词，以二元组形式存输出放在文件中。

2.2 语法分析

- (1) 利用 LR(1)分析法，设计一个语法分析程序，对输入单词符号串进行语法分析；
- (2) 输出推导过程中所用产生式序列并保存在输出文件中。

2.3 典型语句的语义分析及中间代码生成

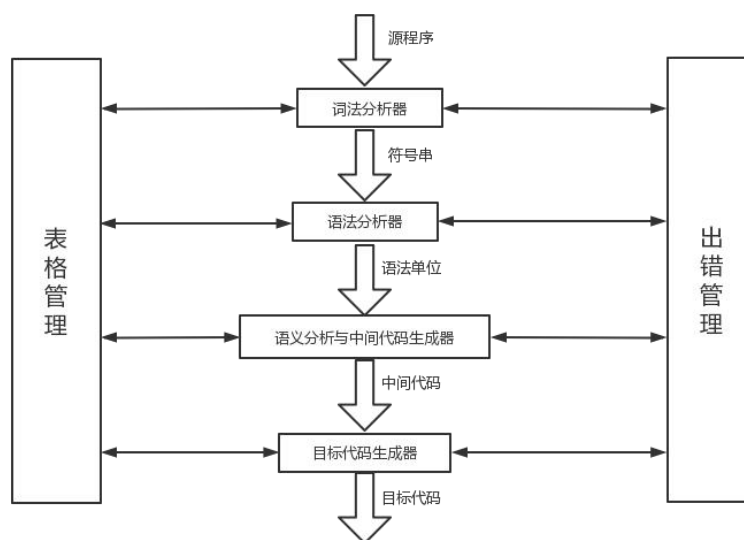
- (1) 针对自顶向下或者自底向上分析法（二选一）中所使用的文法，在完成实验二（语法分析）的基础上为语法正确的单词串设计翻译方案。
- (2) 利用该翻译方案，对所给程序段进行分析，输出生成的中间代码序列和符号表，并保存在相应文件中。

2.4 目标代码生成

- (1) 将中间代码所给的地址码生成目标代码（汇编指令）；
- (2) 写出代码序列表；
- (3) 减少程序与指令的开销，进行部分优化。

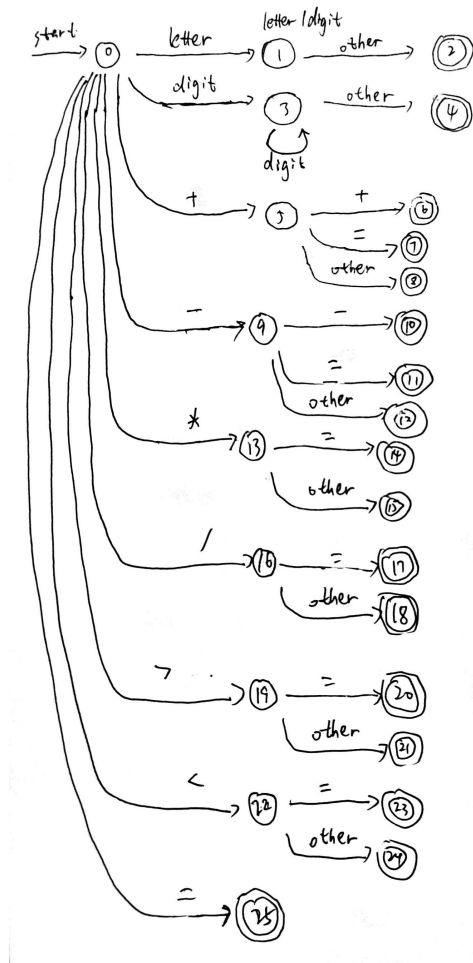
3 实验总体流程与函数功能描述

(1) 整体流程：



(2) 词法分析器：

状态机:



词类编码表:

名称	编号	返回值	名称	编号	返回值
标识符	1	内部字符串	--	47	0
无符号整数	2	整数值	->	48	0
布尔常数	3	0 or 1	*	49	0
字符串常数	4	字符串	*=	50	0
begin	5	0	/	51	0
bool	6	0	/=	52	0
break	7	0	%	53	0
case	8	0	%=	54	0
char	9	0	!	55	0
const	10	0	!=	56	0
continue	11	0	^	57	0
default	12	0	^=	58	0
do	13	0	&	59	0
double	14	0	&=	60	0
else	15	0	&&	61	0
end	16	0		62	0

float	17	0	=	63	0
for	18	0		64	0
goto	19	0	:	65	0
if	20	0	::	66	0
int	21	0	<	67	0
long	22	0	<=	68	0
main	23	0	<<	69	0
return	24	0	<<=	70	0
short	25	0	>	71	0
signed	26	0	>=	72	0
sizeof	27	0	>>	73	0
static	28	0	>>=	74	0
string	29	0	[75	0
struct	30	0]	76	0
switch	31	0	{	77	0
typedef	32	0	}	78	0
unsigned	33	0	\	79	0
until	34	0	(80	0
void	35	0)	81	0
while	36	0	#	82	0
=	40	0	,	83	0
==	41	0	.	84	0
+	42	0	?	85	0
+=	43	0	;	86	0
++	44	0	~	87	0
-	45	0	"	88	0
--	46	0			

函数说明：

```
//扫描指针前进
void go_forward(char& ch);

//扫描输入缓冲区（即源程序 string）
token token_scan();

//将原程序转化为 string 形式
void get_program(char* file);

//符号表打印
void list_printf(int n);

//符号表更新
void link_update(string n, string type);
```

(3) 词法分析器

DFA（有穷状态自动机）：

[https://gitee.com/alchemy-star/compilers/blob/master/LR\(1\)DFA.dfa](https://gitee.com/alchemy-star/compilers/blob/master/LR(1)DFA.dfa)

LR1 分析表：

[https://gitee.com/alchemy-star/compilers/blob/master/LR\(1\)a_table.pdf](https://gitee.com/alchemy-star/compilers/blob/master/LR(1)a_table.pdf)

函数说明：

```
//归约后转移到目标状态
bool goto_state(int top_state, string top_sym);

//移进并完成状态转移
void shift(int next_state);

//归约，同时生成产生式和目标代码
void reduce(int pro_code);

//确定动作，依据动作表和输入区首符号
bool action(int cur_state, string next_sym);
```

(4) 典型语句的语义分析及中间代码生成

基本步骤和流程同（3）词法分析，主要在归约的同时生成中间代码（三地址码）

函数说明：

```
//根据归约情况和产生式创建三地址码
void make_code(string op, string r1, string r2, string t);

//归约后转移到目标状态
bool goto_state(int top_state, string top_sym);

//移进并完成状态转移
void shift(int next_state);

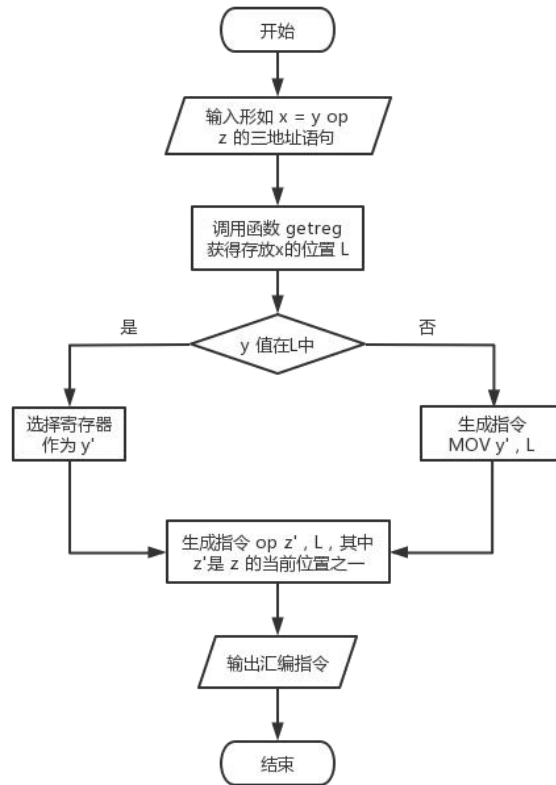
//归约，同时生成产生式和目标代码
void reduce(int pro_code);
```

```
//确定动作，依据动作表和输入区首符号
```

```
bool action(int cur_state, string next_sym);
```

(5) 目标代码生成

流程图：



函数说明：

```
//返回保存 x = y op z 的 x 值的位置 L
```

```
string getreg(string r);
```

```
//形成最终目标代码（汇编代码）
```

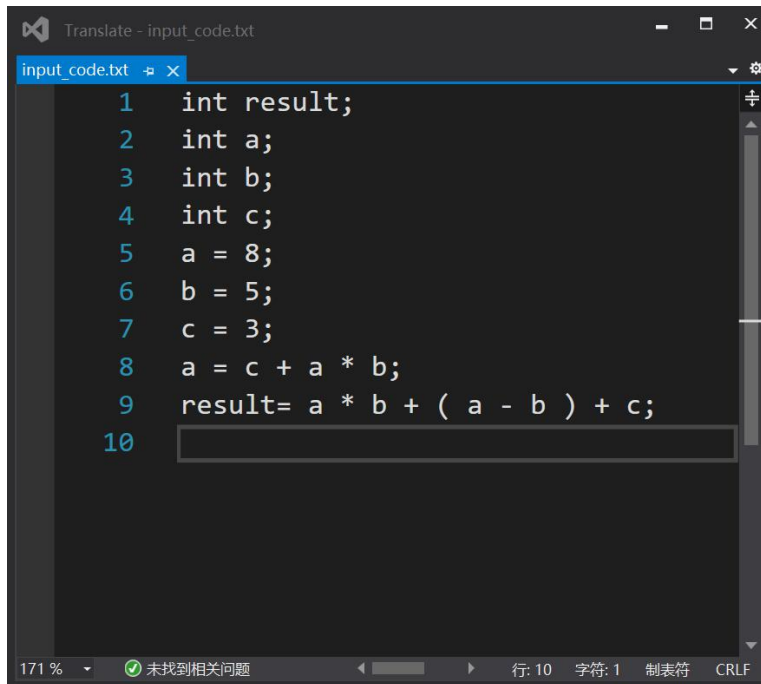
```
void make_aidcode();
```

4 实验结果与分析

(1) 词法分析

输入：

input_code.txt



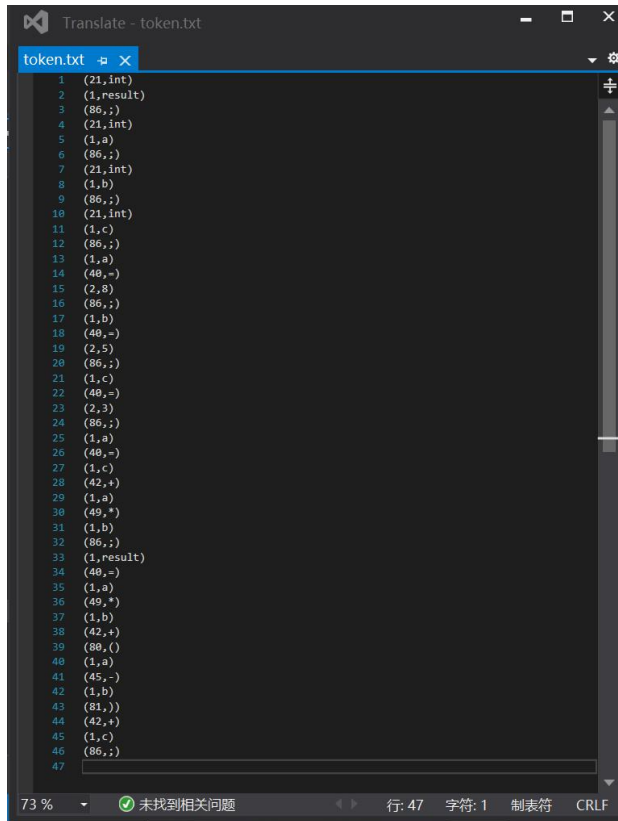
The screenshot shows a code editor window titled "Translate - input_code.txt". The editor contains the following C code:

```
1  int result;  
2  int a;  
3  int b;  
4  int c;  
5  a = 8;  
6  b = 5;  
7  c = 3;  
8  a = c + a * b;  
9  result= a * b + ( a - b ) + c;  
10
```

The status bar at the bottom indicates "171 %", "未找到相关问题", "行: 10", "字符: 1", "制表符", and "CRLF".

输出:

token.txt

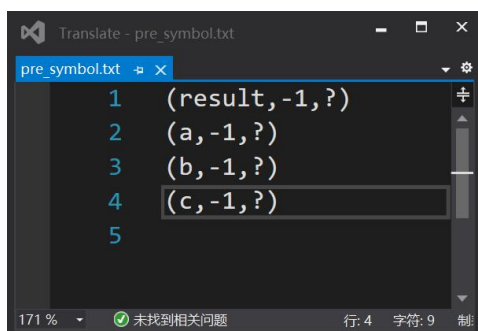


The screenshot shows a code editor window titled "Translate - token.txt". The editor contains the following token stream:

```
1  (21,int)  
2  (1,result)  
3  (86,;)  
4  (21,int)  
5  (1,a)  
6  (86,;)  
7  (21,int)  
8  (1,b)  
9  (86,;)  
10 (21,int)  
11 (1,c)  
12 (86,;)  
13 (1,a)  
14 (40,-)  
15 (2,8)  
16 (86,;)  
17 (1,b)  
18 (40,-)  
19 (2,5)  
20 (86,;)  
21 (1,c)  
22 (40,-)  
23 (2,3)  
24 (86,;)  
25 (1,a)  
26 (40,-)  
27 (1,c)  
28 (42,+)  
29 (1,a)  
30 (49,*)  
31 (1,b)  
32 (86,;)  
33 (1,result)  
34 (40,-)  
35 (1,a)  
36 (49,*)  
37 (1,b)  
38 (42,+)  
39 (80,())  
40 (1,a)  
41 (45,-)  
42 (1,b)  
43 (81,))  
44 (42,+)  
45 (1,c)  
46 (86,;)  
47
```

The status bar at the bottom indicates "73 %", "未找到相关问题", "行: 47", "字符: 1", "制表符", and "CRLF".

pre_symbol.txt



```
1 (result, -1, ?)
2 (a, -1, ?)
3 (b, -1, ?)
4 (c, -1, ?)
5
```

分析：每个单词和符号都以一个二元组形式输出，左元素为符号编码，右符号为变量名。临时符号表暂时只有变量名，按照实验要求变量类型为 -1，地址为？。

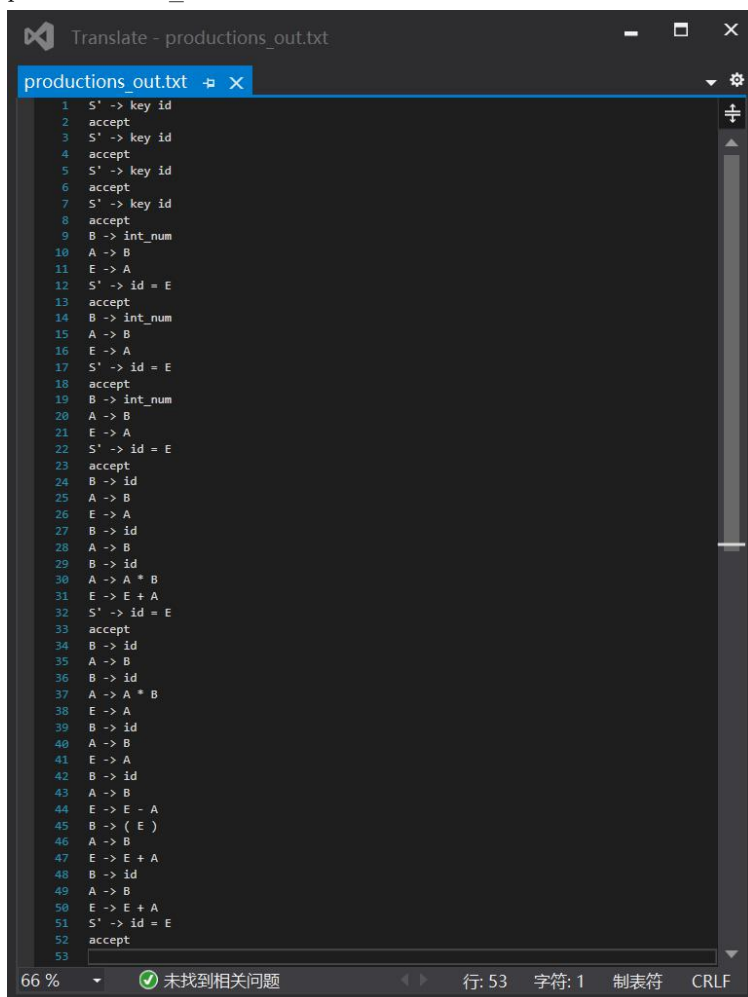
(2) 词法分析

输入：

(1) 的输出 token.txt

输出：

productions_out.txt



```
1 S' -> key id
2 accept
3 S' -> key id
4 accept
5 S' -> key id
6 accept
7 S' -> key id
8 accept
9 B -> int_num
10 A -> B
11 E -> A
12 S' -> id = E
13 accept
14 B -> int_num
15 A -> B
16 E -> A
17 S' -> id = E
18 accept
19 B -> int_num
20 A -> B
21 E -> A
22 S' -> id = E
23 accept
24 B -> id
25 A -> B
26 E -> A
27 B -> id
28 A -> B
29 B -> id
30 A -> A * B
31 E -> E + A
32 S' -> id = E
33 accept
34 B -> id
35 A -> B
36 B -> id
37 A -> A * B
38 E -> A
39 B -> id
40 A -> B
41 E -> A
42 B -> id
43 A -> B
44 E -> E - A
45 B -> ( E )
46 A -> B
47 E -> E + A
48 B -> id
49 A -> B
50 E -> E + A
51 S' -> id = E
52 accept
53
```

分析：由（1）的符号串，经过 LR（1）分析后可以得出系列归约式，
每个句子归约成功后输出 accept

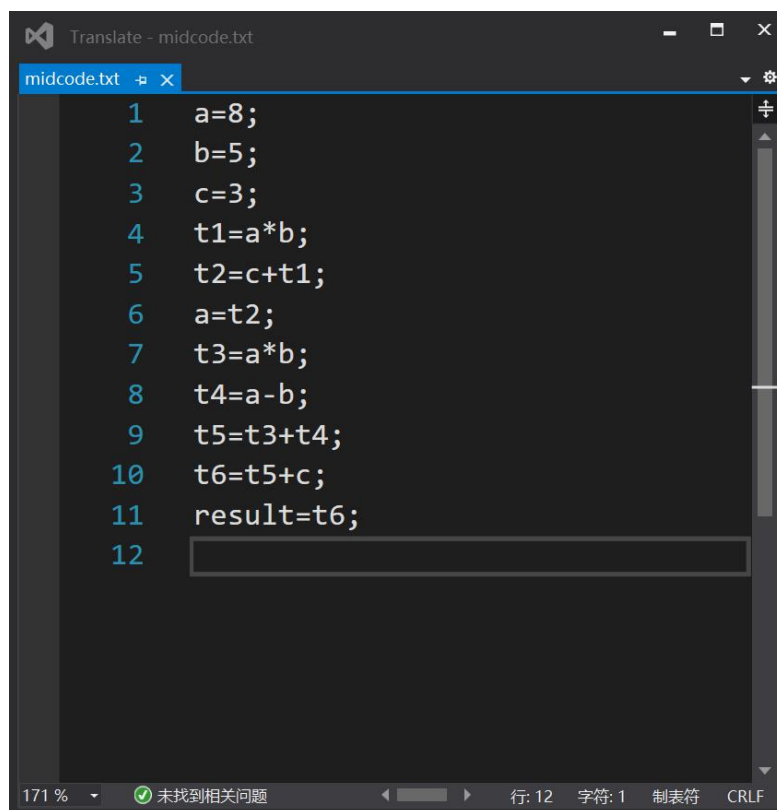
（3）中间代码生成

输入：

（1）的输出 token.txt 和 pre_symbol.txt

输出：

midcode.txt



```
Translate - midcode.txt
midcode.txt
1  a=8;
2  b=5;
3  c=3;
4  t1=a*b;
5  t2=c+t1;
6  a=t2;
7  t3=a*b;
8  t4=a-b;
9  t5=t3+t4;
10 t6=t5+c;
11 result=t6;
12
```

分析：每行中间代码对应一个产生式，临时变量 tx 由产生式左部决定，结果实际上以四元式三地址码形式存储，输出时按实验要求写成赋值语句形式。

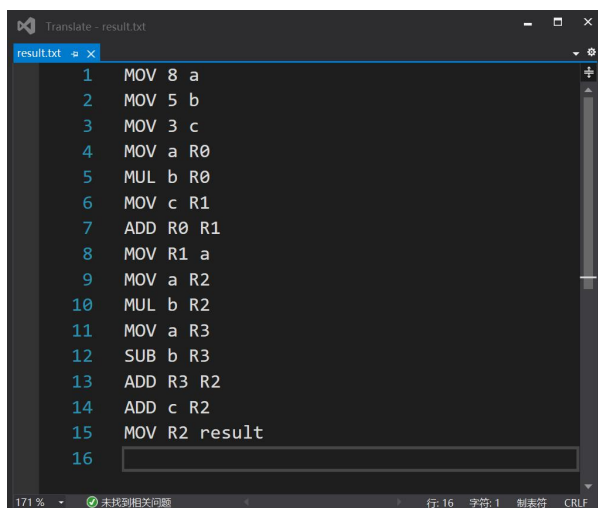
（4）目标代码生成

输入：

（3）的输出 midcode.txt

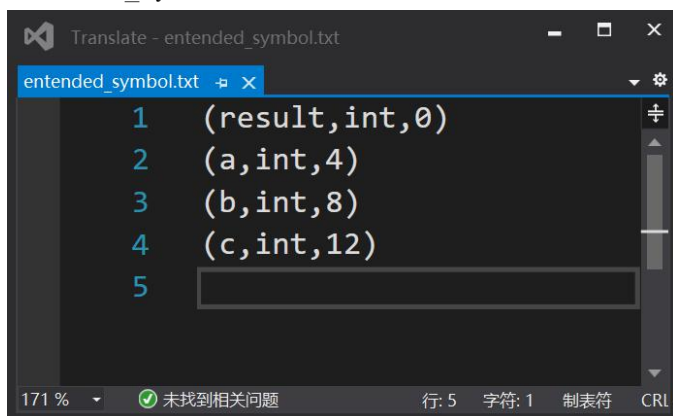
输出：

result.txt



```
1 MOV 8 a
2 MOV 5 b
3 MOV 3 c
4 MOV a R0
5 MUL b R0
6 MOV c R1
7 ADD R0 R1
8 MOV R1 a
9 MOV a R2
10 MUL b R2
11 MOV a R3
12 SUB b R3
13 ADD R3 R2
14 ADD c R2
15 MOV R2 result
16
```

extended_symbol.txt



```
1 (result,int,0)
2 (a,int,4)
3 (b,int,8)
4 (c,int,12)
5
```

分析：每行中间代码可以写成一个或多行汇编代码语句。其中 Rx 表示寄存器，基本用于存储历史变量 tx。

拓展符号表较临时符号表多了变量类型和内存地址。

5 实验中遇到的困难与解决办法

(1) 遇到的困难及解决方法：

- a. 设置文法时，选择了语句嵌套和声明嵌套，导致产生了 90 多个状态；

解决方法：经过分析，发现语句嵌套没有必要，遂删去，后来由于时间关系也放弃了声明嵌套。

- b. 语法分析是，出现 $E \rightarrow A$ 无法归约的 bug

解决方法：经过反复调试，最终排查出是某状态多弹出一个元素。后修正。

(2) 收获：

- a. 对编译原理的知识和内容理解更加深刻，掌握更加牢固，较好地将理论付诸实践
- b. 对编译器的实现原理了解更加全面，并初步成功实现了一个简单的编译器
- c. 编程能力和调试能力有了更进一步的提高，代码经验和调试经验更加丰富
- d. 对 c++的使用和代码编写更加熟练

(3) 意见建议:

暂无