编译原理课程实验报告

实验 2：语法分析

姓名	杨军瑗	院系	计算机	学号	1160300427
任课教师	陈鄞		指导教师	廖阔	
实验地点	G214		实验时间	4.21	
实验课表现	出勤、表现得分		实验报告得分		实验总分	
	操作结果得分					

一、需求分析

得分

要求：采用至少一种句法分析技术（LL(1)、SLR(1)、LR(1)或 LALR(1)）对类高级语言中的基本语句进行句法分析。阐述句法分析系统所要完成的功能。

使用 LL1 文法

1、实验目的

（1）巩固对语法分析的基本功能和原理的认识。

（2）通过对语法分析表的自动生成加深语法分析表的认识。

（3）理解并处理语法分析中的异常和错误。

2、实验内容

在词法分析器的基础上设计实现类高级语言的语法分析器，基本功能如下：

能识别以下几类语句：

（1）声明语句（包括变量声明、数组声明、记录声明（record）和过程声明(proc)）

（2）表达式及赋值语句（包括数组元素的引用和赋值）

（3）分支语句：if_then_else

（4）循环语句：do_while

（5）过程调用语句：函数调用 call

3、系统的输入形式：要求通过文件导入测试用例。

4、系统的输出形式：打印输出语法分析结果，以语法树结构给出

5、语法分析器采用的是自顶向下的分析技术，要求编写程序自动计算 FIRST 集和 FOLLOW 集，并根据 SELECT 集自动生成预测分析表。

6、具备语法错误处理能力，能准确给出错误所在位置，并采用可行的错误恢复策略。输出的错误提示信息格式如下：

Error at Line [行号]：[说明文字]

二、文法设计	得分	

要求：给出如下语言成分的文法描述。

```
Program->P          /*程序入口*/
P->D P|S P|empty
```

- 声明语句（包括变量声明、数组声明、记录声明和过程声明）

```
D->T id A;|record id { P }       /*变量声明，记录声明*/
T->X C
X->int|real|char                 /*允许变量声明的类型*/
```

- 表达式及赋值语句（包括数组元素的引用和赋值）

```
A->= F A|empty|, id A          /*声明时赋值，连续声明*/
C>[ num ] C|empty               /*声明数组类型，允许多维*/
L->id L'                        /*对变量或是数组进行赋值*/
L'->[ num ] L'|empty
```

- 分支语句：if_then_else
- 循环语句：do_while

```
S->L = E ;|if B then S else S|while B do S /*赋值语句，分支语句，循环语句*/
E->G E'                  /*表达式*/
E'->+ G E'|empty
G->F G'
G'->* F G'|empty
F->( E )|id|digit|char
```

- 过程调用语句

```
D->proc X id ( M ) { P }   /*函数声明*/
M->X id M'|empty           /*参数类型声明*/
M'->, X id M'|empty
S->call id ( Elist ) ;|return E ;    /*函数调用和返回值*/
Elist->E Elist'
Elist'->, E Elist'|empty
```

其他：

```
B->H B'          /*逻辑表达式*/
B'->or H B'|empty
H->I H'
H'->and I H'|empty
I->not B|( B )|E relop E|true|false
```

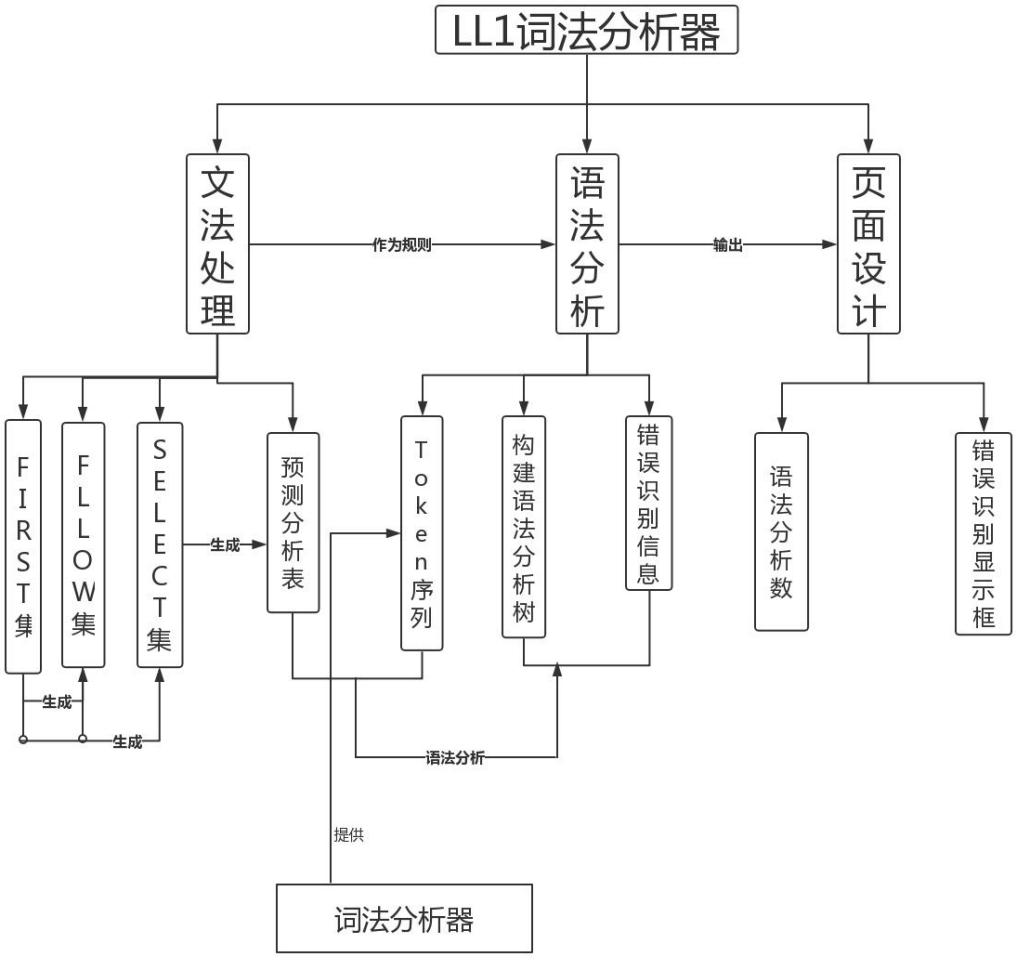
relop-><|<=|==|!=|>|>=

三、系统设计

得分

要求：分为系统概要设计和系统详细设计。

1、系统概要设计：给出必要的系统宏观层面设计图，如系统框架图、数据流图、功能模块结构图等以及相应的文字说明。



（1）语法分析器的输入是测试程序经过词法分析器之后所得到的 token 序列；

（2）输出是语法分析时的语法分析树，以先序遍历的形式输出；

（3）使用的规则是 LL1 文法，需要对这个文法进行预处理，根据文法计算出非终结符的 FIRST 集和 FOLLOW 集，并且根据 FIRST 集和 FOLLOW 集得到 LL1 产生式的 SELECT 集，最终可由 SELECT 集合构造得到预测分析表。预测分析表即是进行语法分析的规则。

（4）语法分析的过程中可能是产生错误信息，记录这些错误信息。

（5）更新页面，输出语法分析书和错误识别信息。

2、系统详细设计：对如下工作进行展开描述

✓ 核心数据结构的设计

（1）YufaAnalyze.java:

这个类中对词法分析得到的 token 序列使用 LL1 预测分析表进行分析，是语法分析的主要逻辑实现。

从 LL1 中得到终结符集合：

```
public ArrayList<String> finallist = new ArrayList<String>();
```

从 LL1 中得到非终结符集合：

```
public ArrayList<String> notfinallist = new ArrayList<String>();
```

输出的树的集合：

```
public ArrayList<YufaNode> treelist = new ArrayList<YufaNode>();
```

自定义树节点类 YuFaNode:

```
public String node;                当前节点
public String nodevalue;           当前节点的属性值
```

当前节点的子节点集合：

```
public ArrayList<YufaNode> chirdlist = new ArrayList<YufaNode>();
public int line;                   当前节点所处的层数：
```

错误信息集合：

```
public ArrayList<String> errorlist = new ArrayList<String>();
```

存储预测分析表：

```
public String[][] all_ll = new String[50][30];
```

可视化生成的语法分析树：

```
public JTree tree;
public JScrollPane jsp_tree1;
DefaultMutableTreeNode top = new DefaultMutableTreeNode("Program");
tree = new JTree(top);
```

识别到的句子集合：

```
public ArrayList<String> senlist = new ArrayList<String>();
```

（2）FirstFollow.java、SelectTable.java

从 FirstFollow.java 中我们根据 LL1 文法计算出各个非终结符的 First 和 Follow 集合，并且将这些集合写入文件 first.txt,follow.txt。

在 SelectTable.java 中，我们根据 FirstFollow.java 得到的 first.txt,follow.txt 进一步得到 select 集，同样存储入 select.txt 中，并且得到预测分析表。

存储读入的文法：in[0]是左部，in[i](i>=1)是使用“|”分割开的一个个右部产生式。

```
public ArrayList<String[]> in=new ArrayList<String[]>();
```

存储计算得到的 first 集合：

```
public ArrayList<String[]> first = new ArrayList<String[]>();
```

存储计算得到的 follow 集合：

```
public ArrayList<String[]> follow = new ArrayList<String[]>();
```

存储计算得到的 select 集合：

```
public ArrayList<String[]> select= new ArrayList<String[]>();
```

存储计算 first 和 follow 时的一条条路径，辅助完成 first 和 follow 的计算,防止计算到重复值：

```
public ArrayList<String[]> track = new ArrayList<String[]>();
```

存储预测分析表：

```
public String[][] all_ll = new String[50][30];
```

✓ 主要功能函数说明

（1）YufaAnalyze.java:

`public void YufaAnalyze()` ：完成语法分析的主函数。主要逻辑实现见程序流程图。

（2）FirstFollow.java

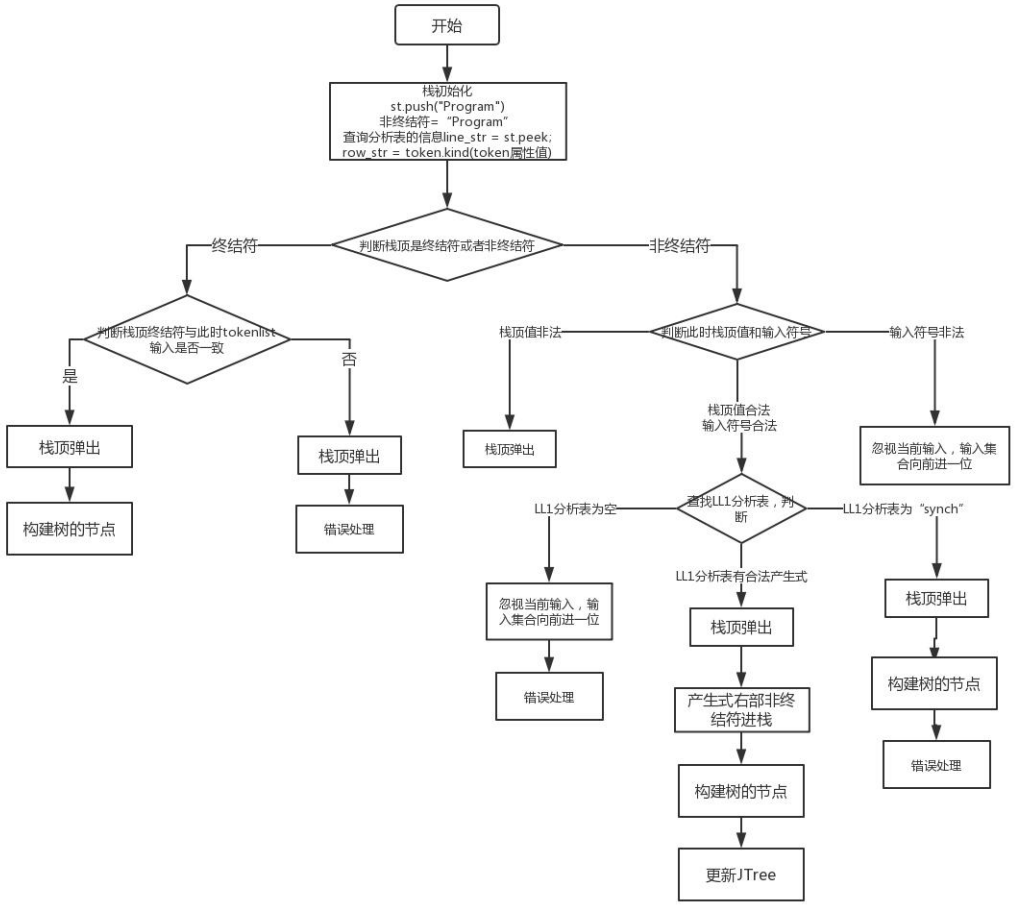
`public ArrayList<String> getFirst(String s,String track1)` ：计算 first 集

`public ArrayList<String> getFollow(String s,String element,String track1)`：根据 follow 集计算 follow 集

`public ArrayList<String> returnFirstofFollow(String s,String element,String track1,String one0,String one1,int index,int t)`：根据 first 集计算 follow 集

✓ 程序核心部分的程序流程图

（1）LL1 分析的流程图



四、系统实现及结果分析

得分

要求：对如下内容展开描述。

1、系统实现过程中遇到的问题；

（1）文法的处理：

该分析器使用的是 LL1 文法，所以需要事先对文法进行处理，消除左递归。

（2）从文法中提取有用信息：

语法分析器的输入来自于词法分析器的 tokenlist；

规则来自于预测分析表，预测分析表由产生式的 select 集合决定，所以需要事先计算 first 集和 follow 集。

2、输出该句法分析器的分析表；

First 集：

first.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

	1	2	3	4	5	6	7	8	9	10	11
Program	proc	int	real	char	record	id	if	while	call	return	ε
P	proc	int	real	char	record	id	if	while	call	return	ε
D	proc	int	real	char	record						
A	=	ε	,								
M	int	real	char	ε							
M'	,	ε									
T	int	real	char								
X	int	real	char								
C	[	ε									
S	id	if	while	call	return						
V	++	--									
N	ε										
E	(	digit	char	id							
E'	+	ε									
G	(	digit	char	id							
G'	*	ε									
F	(	digit	char	id							
L	id										
L'	[	ε									
B	not	(	digit	char	id	true	false				
B'	or	ε									
H	not	(	digit	char	id	true	false				
H'	and	ε									
I	not	(	digit	char	id	true	false				
Relop	<	<=	==	!=	>	>=					
Elist	(	digit	char	id							
Elist'	,	ε									

Follow 集：

follow.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Program	$																		
P	$	}																	
D	proc	int	real	char	record	id	if	while	call	return	$	}							
A	;																		
M	)																		
M'	)																		
T	id																		
X	id	[																	
C	id																		
S	proc	int	real	char	record	id	if	while	call	return	$	else	}	}	not	(	digit	true	false
V	)																		
N																			
E	;	)	<	<=	==	!=	>	>=	and	or	then	do	,						
E'	;	)	<	<=	==	!=	>	>=	and	or	then	do	,						
G	;	)	<	<=	==	!=	>	>=	and	or	then	do	,	+					
G'	;	)	<	<=	==	!=	>	>=	and	or	then	do	,	+					
F	=	,	;	*	+	)	<	<=	==	!=	>	>=	and	or	then	do			
L	=																		
L'	=																		
B	then	do	;	and	or	)													
B'	then	do	;	and	or	)													
H	or	then	do	;	and	)													
H'	or	then	do	;	and	)													
I	and	or	then	do	;														
Relop	(	digit	char	id															
Elist	)																		
Elist'	)																		

Select 集：

select.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Program->P		$	proc	int	real	char	record	id	if	while	call	return		
P->D P	proc	int	real	char	record									
P->S P	id	if	while	call	return									
P->ε	$	}												
D->proc X id (M) { P }				proc										
D->T id A ;		int	real	char										
D->record id { P }		record												
A->= F A		=												
A->ε	;													
A->, id A		,												
M->X id M'		int	real	char										
M'->, X id M'		,												
M'->ε	)													
T->X C	int	real	char											
X->int	int													
X->real	real													
X->char	char													
C->[digit] C	[													
C->ε	id													
S->L = E		id												
S->if B then S else S		if												
S->while B do S	while													
S->call id (Elist) ;		call												
S->return E ;		return												
V->++	++													
V->--	--													
E->G E'	(	digit	char	id										
E'->+ G E'	+													
E'->ε	:	)	<	<=	==	!=	>	>=	and	or	then	do	,	
G->F G'	digit	(	char	id										
G'->* F G'	*													
G'->ε	+	;	)	<	<=	==	!=	>	>=	and	or	then	do	,
F->(E)	(													
F->digit	digit													
F->char	char													
F->id	id													
L->id L'		id												
L'->[digit] L'		[												
L'->ε	=													
B->H B'	not	(	digit	char	id	true	false							
B'->or H B'		or												
B'->ε	then	do	;	and	)									
H->I H'	not	(	digit	char	id	true	false							
H'->and I H'		and												
H'->ε	or	then	do	;	and	)								
I->not B		not												
I->(B)		(												
I->E Relop E		(	digit	char	id									
I->true	true													
I->false		false												
Relop-><=		<												
Relop->=		<=												
Relop->==		==												
Relop->!=		!=												
Relop->>		>												
Relop->>=		>=												
Elist-> E Elist'		(	digit	char	id									
Elist'->, E Elist'		,												
Elist'->ε		)												

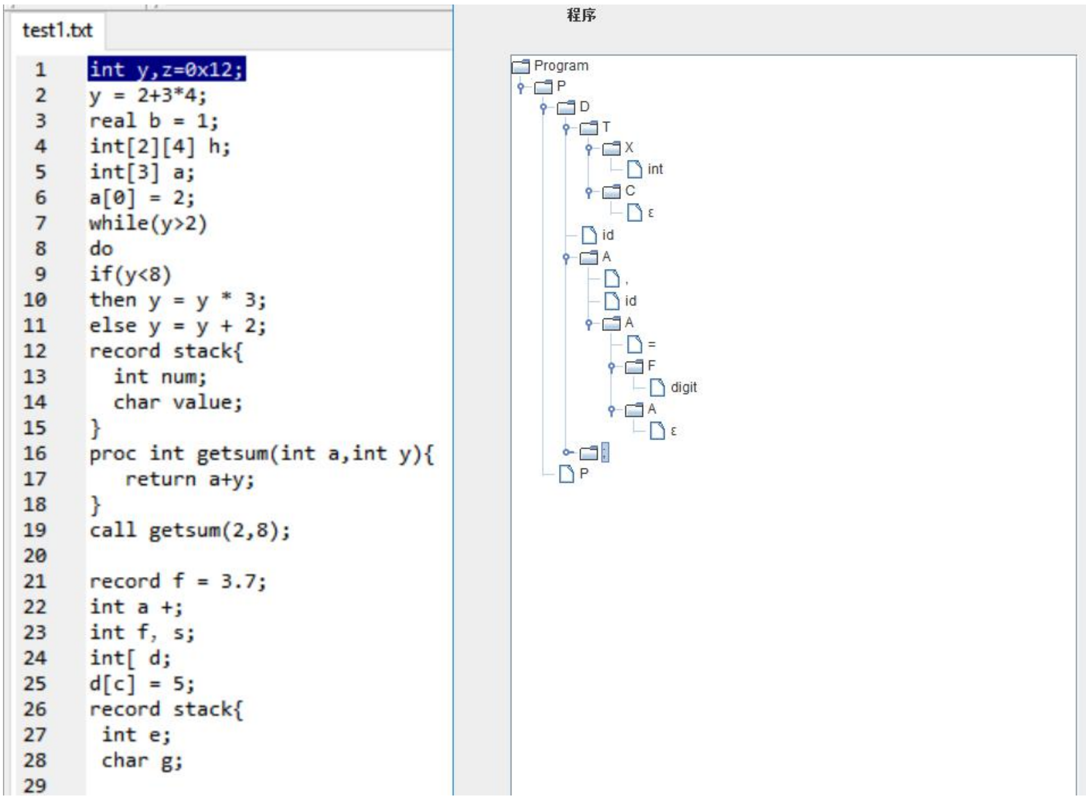
LL1 预测分析表：

	$	proc	int	real	char	record	id	if	while	call	return	}	=	;	,	)	[	++	--	(	digit
Program	Program->ε	Program->D P	Program->D P	Program->D P	Program->D P	Program->D P	Program->D P	Program->S P	Program->S P	Program->S P	Program->S P	Program->S P	Program->ε									
P	P->ε	P->D P	P->D P	P->D P	P->D P	P->D P	P->D P	P->S P	P->S P	P->S P	P->S P	P->S P	P->ε									
D		D->proc	D->T id	D->T id	D->T id	D->T id	D->record	synch														
A														A->= F A	A->ε		A->, id					
M		M->X id	M->X id	M->X id	M->X id	M->X id	M->X id															
M'																M'->, X M'	M'->ε					
T		T->X C	T->X C	T->X C	T->X C	T->X C																
X		X->int	X->real	X->char																		
C							C->ε										C->[dig					
S							S->L = E	S->if B	S->while	S->call	S->retur											
V																		V->++	V->--			
N																						
E						E->G E'	E->G E'															
E'														E'->ε	E'->ε		E'->ε					
G						G->F G'	G->F G'															
G'														G'->ε	G'->ε		G'->ε					
F					F->char	F->id																
L							L->id L'															
L'																	L'->[di					
B					B->H B'	B->H B'																
B'														B'->ε								
H					H->I H'	H->I H'																
H'														H'->ε								
I					I->E Rel	I->E Rel																
Relop																						
Elist					Elist->E	Elist->E																
Elist'																						

+	<	<=	==	!=	>	>=	and	or	then	do	*	not	TRUE	FALSE
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
E'->+ G	E'->ε	E'->ε	E'->ε	E'->ε	E'->ε	E'->ε	E'->ε	E'->ε	E'->ε	E'->ε	-1	-1	-1	-1
synch	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->ε	G'->* F	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	synch	-1	-1	B->H B'	B->H B'	B->H B'
-1	-1	-1	-1	-1	-1	-1	B'->ε	B'->or H	B'->ε	B'->ε	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	synch	-1	-1	-1	H->I H'	H->I H'	H->I H'
-1	-1	-1	-1	-1	-1	-1	H'->and	H'->ε	H'->ε	H'->ε	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	synch	-1	-1	-1	-1	I->not B	I->true	I->false
-1	Relop-><	Relop-><	Relop->=	Relop->!	Relop->>	Relop->>	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

3、针对一测试程序输出其句法分析结果；

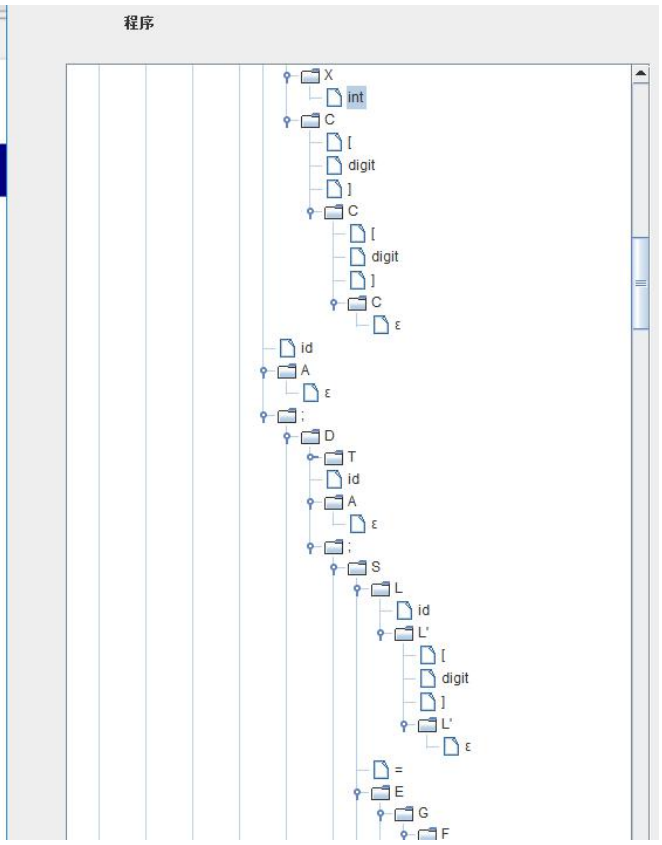
声明赋值语句：



（多维）数组声明：

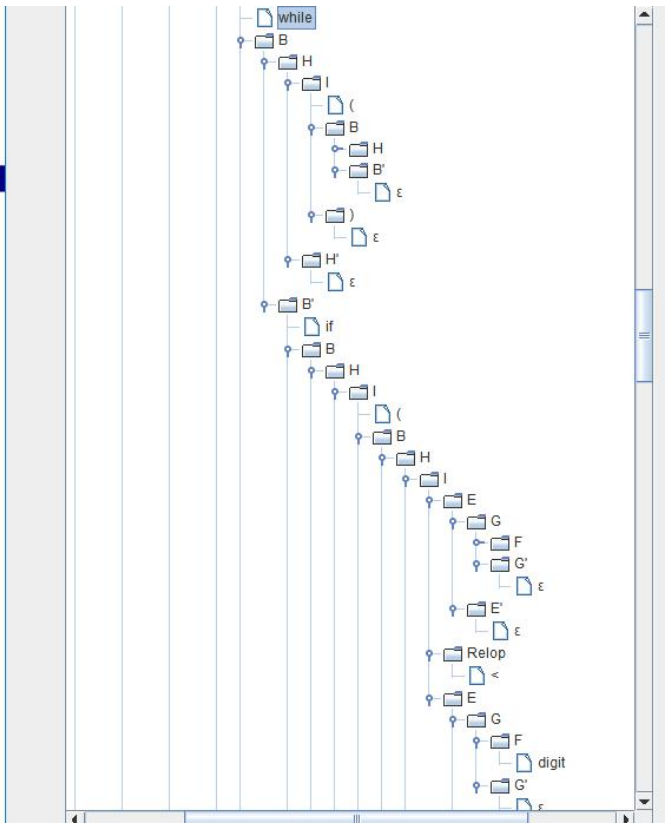
test1.txt

```
1   int y,z=0x12;
2   y = 2+3*4;
3   real b = 1;
4   int[2][4] h;
5   int[3] a;
6   a[0] = 2;
7   while(y>2)
8   do
9   if(y<8)
10  then y = y * 3;
11  else y = y + 2;
12  record stack{
13    int num;
14    char value;
15  }
16  proc int getsum(int a,int y){
17     return a+y;
18  }
19  call getsum(2,8);
20
21  record f = 3.7;
22  int a +;
23  int f, s;
24  int[ d;
25  d[c] = 5;
26  record stack{
27   int e;
28   char g;
29
```



循环语句：

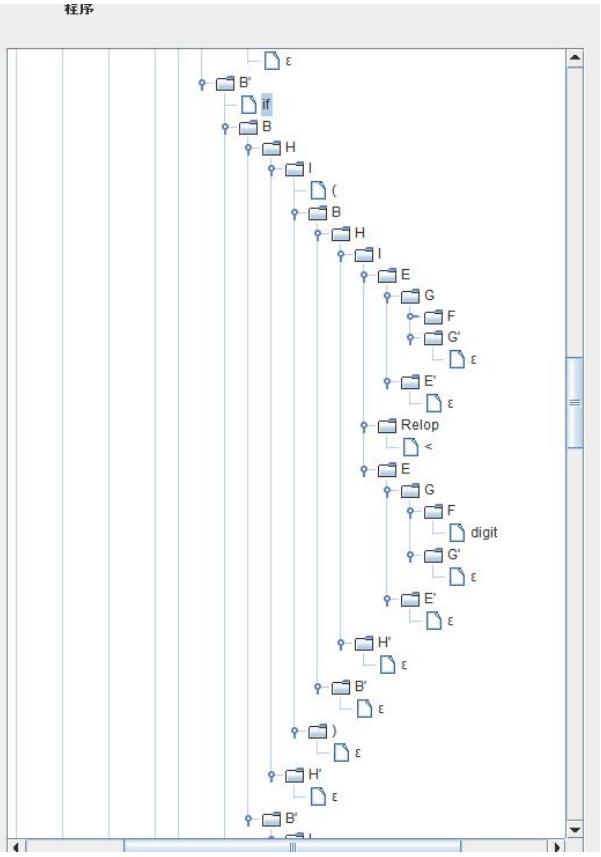
```
1   int y,z=0x12;
2   y = 2+3*4;
3   real b = 1;
4   int[2][4] h;
5   int[3] a;
6   a[0] = 2;
7   while(y>2)
8   do
9   if(y<8)
10  then y = y * 3;
11  else y = y + 2;
12  record stack{
13    int num;
14    char value;
15  }
16  proc int getsum(int a,int y){
17     return a+y;
18  }
19  call getsum(2,8);
20
21  record f = 3.7;
22  int a +;
23  int f, s;
24  int[ d;
25  d[c] = 5;
26  record stack{
27   int e;
28   char g;
29
```



分支语句：

test1.txt

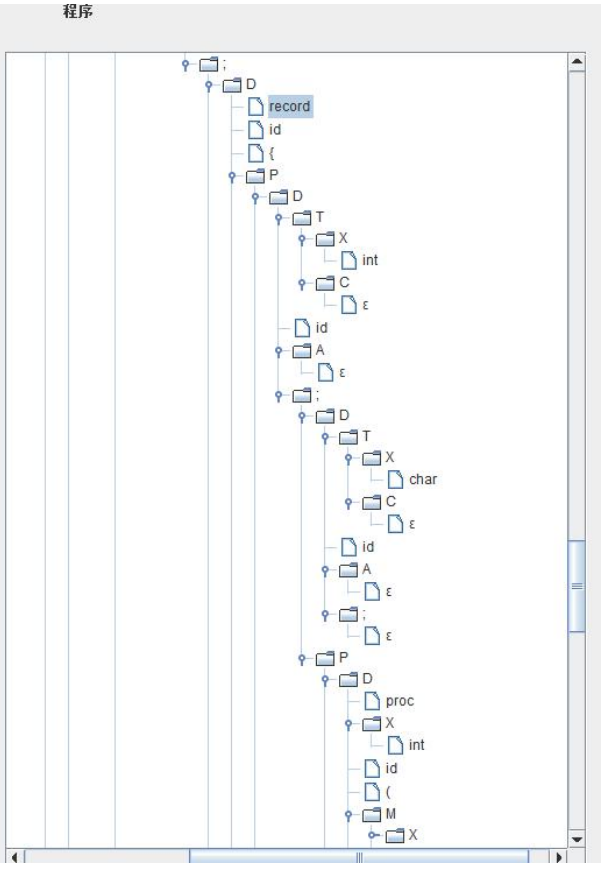
```
1   int y,z=0x12;
2   y = 2+3*4;
3   real b = 1;
4   int[2][4] h;
5   int[3] a;
6   a[0] = 2;
7   while(y>2)
8   do
9   if(y<8)
10  then y = y * 3;
11  else y = y + 2;
12  record stack{
13    int num;
14    char value;
15  }
16  proc int getsum(int a,int y){
17     return a+y;
18  }
19  call getsum(2,8);
20
21  record f = 3.7;
22  int a +;
23  int f, s;
24  int[ d;
25  d[c] = 5;
26  record stack{
27   int e;
28   char g;
29
```



记录声明：

test1.txt

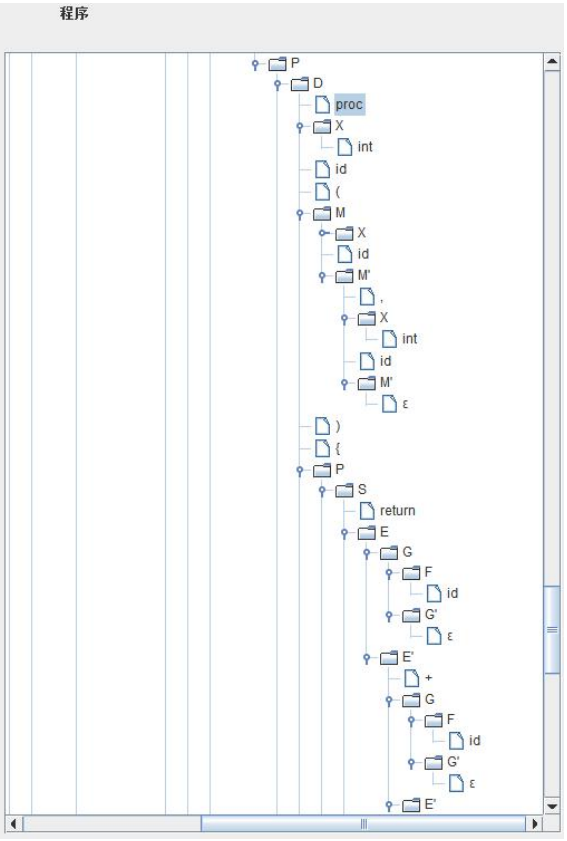
```
1   int y,z=0x12;
2   y = 2+3*4;
3   real b = 1;
4   int[2][4] h;
5   int[3] a;
6   a[0] = 2;
7   while(y>2)
8   do
9   if(y<8)
10  then y = y * 3;
11  else y = y + 2;
12  record stack{
13    int num;
14    char value;
15  }
16  proc int getsum(int a,int y){
17     return a+y;
18  }
19  call getsum(2,8);
20
21  record f = 3.7;
22  int a +;
23  int f, s;
24  int[ d;
25  d[c] = 5;
26  record stack{
27   int e;
28   char g;
29
```



过程声明：

test1.txt

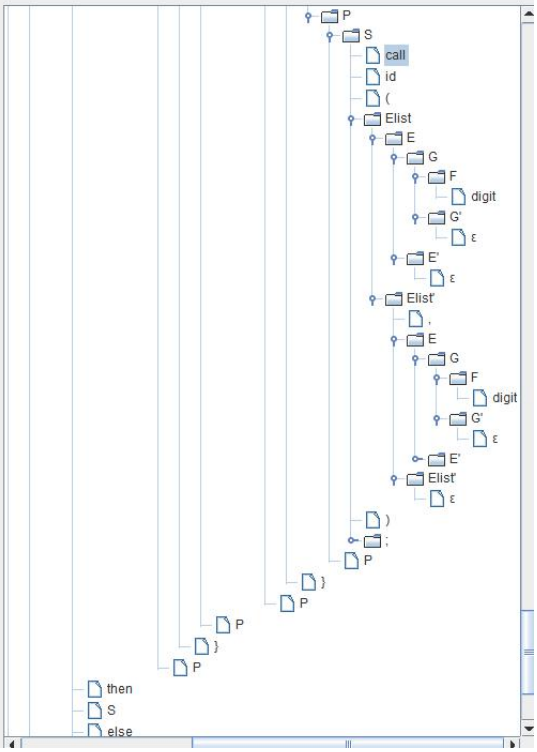
```
1   int y,z=0x12;
2   y = 2+3*4;
3   real b = 1;
4   int[2][4] h;
5   int[3] a;
6   a[0] = 2;
7   while(y>2)
8   do
9   if(y<8)
10  then y = y * 3;
11  else y = y + 2;
12  record stack{
13    int num;
14    char value;
15  }
16  proc int getsum(int a,int y){
17     return a+y;
18  }
19  call getsum(2,8);
20
21  record f = 3.7;
22  int a +;
23  int f, s;
24  int[ d;
25  d[c] = 5;
26  record stack{
27   int e;
28   char g;
29
```



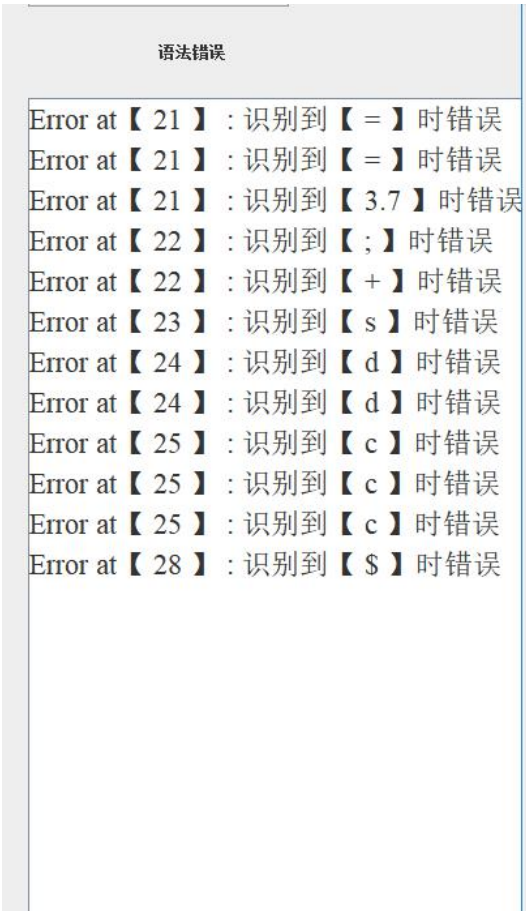
过程调用：

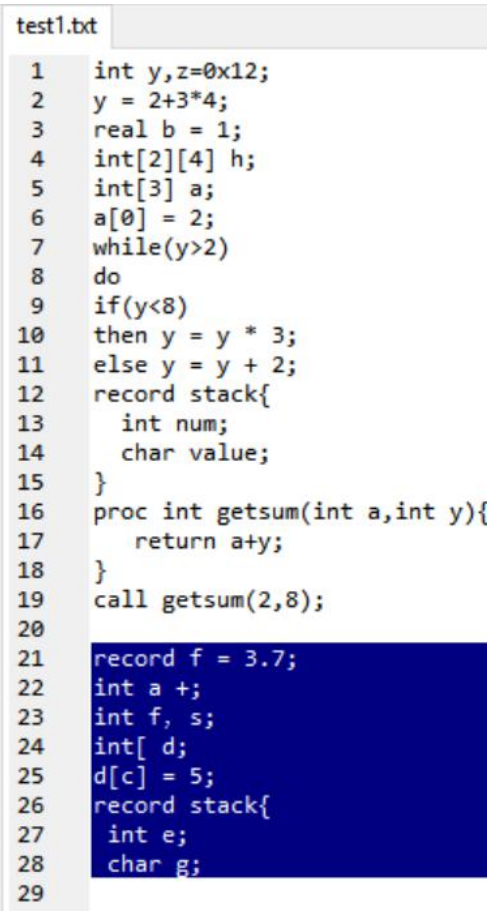
test1.txt

```
1   int y,z=0x12;
2   y = 2+3*4;
3   real b = 1;
4   int[2][4] h;
5   int[3] a;
6   a[0] = 2;
7   while(y>2)
8   do
9   if(y<8)
10  then y = y * 3;
11  else y = y + 2;
12  record stack{
13    int num;
14    char value;
15  }
16  proc int getsum(int a,int y){
17     return a+y;
18  }
19  call getsum(2,8);
20
21  record f = 3.7;
22  int a +;
23  int f, s;
24  int[ d;
25  d[c] = 5;
26  record stack{
27   int e;
28   char g;
29
```



4、输出针对此测试程序对应的语法错误报告；





5、对实验结果进行分析。

实验在分析的过程中以先跟顺序生成语法树，使用 JTree 结构进行输出，并且将节点信息存储进 treelist 中，以便语义分析时使用。以先跟顺序遍历得到的 treelist 存储入 tree.txt 中。

错误分析：

第 21 行，record 声明失败，报错后进入恐慌模式，一直丢弃非法的输入值直到找到合法值为止，如果直到最后识别到标识符“;”时都未找到合法值，则将 21 行整行丢弃。

第 22 行，“a+”错误，丢弃非法；

第 23 行，“，”为中文字符，识别错误，丢弃；

第 25 行，数组声明错误；

第 26 行，数组赋值错误，[]只可为 digit

第 28 行，声明 record 时缺少“}”。

总之，语法树生成正确，正确语句可成功识别，错误项可以找到并报错。语法分析程序执行正确。

注：其中的测试样例需先用已编写的词法分析程序进行处理。

指导教师评语：   日期：