编译原理课程实验报告

实验 3：语义分析

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	2019.04.28	
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：阐述语义分析系统所要完成的功能。

在语法分析器的基础上设计实现类高级语言的语义分析器，基本功能如下：

（1） 能分析以下几类语句，并生成中间代码（三地址指令和四元式形式）：

- 声明语句（包括变量声明、数组声明、记录声明和过程声明）
- 表达式及赋值语句（包括数组元素的引用和赋值）
- 分支语句：if_then_else
- 循环语句：do_while
- 过程调用语句

（2）具备语义错误处理能力，包括变量或函数重复声明、变量或函数引用前未声明、运算符和运算分量之间的类型不匹配（如整型变量与数组变量相加减）等错误，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]

（3）输入为测试用例程序，要求包括以上语句

（4）输出为符号表，三地址指令和四地址码，错误分析。

二、文法设计

得分

要求：给出如下语言成分所对应的语义动作

- 声明语句（包括变量声明、数组声明、记录声明和过程声明）

P→{offset= 0 }D

D→ T id;{ enter(id.lexeme, T.type, offset); offset= offset+ T.width; }D

D->proc X id {type='proc';enterproc(id.lexeme,type,offset);} (M) { P }

D->record id {type='record';enterrecord(id.lexeme,type,offset);} { P }

D→ ε

T→ B{ t =B.type; w =B.width;} C{ T.type= C.type; T.width= C.width;}

B→ int{ B.type= int; B.width= 4; }

B→ real{ B.type= real; B.width= 8; }

B→ real{ B.type= char; B.width= 1; }

M->B id {enter(id.lexeme,B.type.offset);offset=offset+B.width;} M'

M'->ε

C->ε { C.type= t ; C.width= w ;}

C→ [num]C1{ C.type= array(num.val, C1.type); C.width= num.val* C1.width; }

➢ 表达式及赋值语句（包括数组元素的引用和赋值）

普通赋值语句：

S→ id=E; {p=lookup(id.lexeme);if p==nil then error; S.code=E.code|| gen(p '=' E.addr);

E→ E1+E2{E.addr=newtemp();E.code=E1.code||E2.code||gen(E.addr '=' E1.addr '+' E2.addr);

E→ E1*E2{E.addr=newtemp();E.code=E1.code||E2.code|| gen(E.addr '=' E1.addr '*' E2.addr);

E→ -E1{E.addr=newtemp();E.code=E1.code||gen(E.addr '=' 'uminus' E1.addr);

E->(E1){E.addr=E1.addr;E.code=E1.code;

E→ id{E.addr=lookup(id.lexeme);if E.addr==nil then error; E.code= '' ; }

数组元素：

S->id=E;|L=E;{gen(L.array '[' L.offset ']' '=' E.addr);}

E->E1+E2|-E1|(E1)|id|L{E.addr=newtemp();gen(E.addr '=' L.array '[' L.offset ']');}

L->id[E]{L.array=lookup(id.lexeme);

If L.arrar = nil then error;

offset=newtemp();

gen(L.offset '=' E.addr '*' L.type.width);}

|L1[E]{L.array=L1.array;

L.type=L1.type.elem;

t=newtemp();

gen(t ‘=’ E.addr ‘*’ L.type.width);

L.offset=newtemp();

gen(L.offset ‘=’ L1.offset ‘+’ t);}

- 分支语句：if_then_else

S->if{ B.true=newlabel();B.false=newlabel(); }

B then{ label(B.true);S1.next=S.next; }

S1{ gen(‘goto’ S.next) }

Else { label(B.false);S2.next=S.next; }

S2

- 循环语句：do_while

S->while{ B.begin=newlabel();

label(B.begin);

B.true=newlabel();

B.false=S.next; }

B do

{ l abel(B.true);S1.next=B.begin; }

S1 { gen(‘goto’ B.begin); }

- 过程调用语句

S→ call id(Elist) { n=0; forq 中的每个 t
do { gen(‘param’ t);
n = n+1; }
gen(‘call’ id.addr ‘,’ n); }

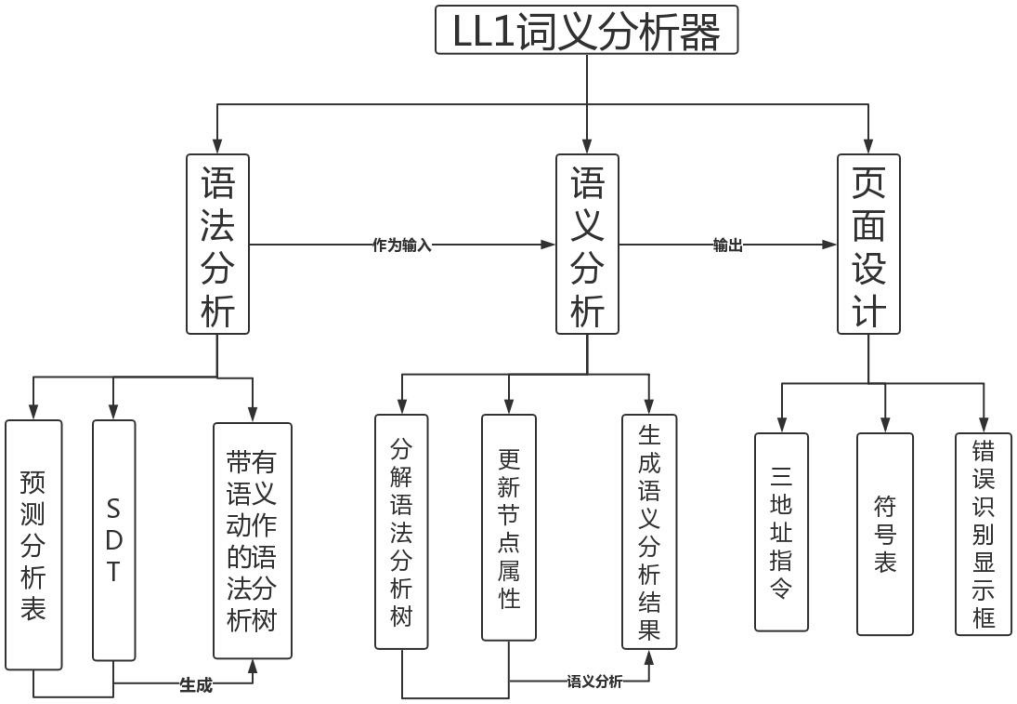
Elist→E { 将 q 初始化为只包含 E.addr; }

Elist→Elist1, E { 将 E.addr 添加到 q 的队尾; }

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。



先前得到的语法分析树并没有带有语义动作，语义分析程序不知道该在何时进行语义分析，所以需要在语法分析阶段获得一棵带有语义动作的语法分析树，这棵树作为语义分析的输入；

值得注意的是，我们语法分析阶段使用 LL1 文法进行的是自顶向上的分析，但是语义分析再更新属性信息时应该是自底向上的规约过程。我们并不可以简单的把整颗语法分析树从最底层开始向上分析，因为这样产生的结果是分析程序从下往上了，比如分析 int a;a=1；的语句，正常情况下应该是先声明 int a，再赋值 a=1，但如果我们把语法分析树彻底颠倒就会导致先分析 a=1,再分析 int a。这将会导致偏移量计算失误、已声明变量永远未声明等错误。

为了解决这个问题，我们需要对语法分析树做一些处理，也就是分解语法分析树，把一整棵语法分析树分解成一棵棵单句规约的小树，也就是两个规约开始符号 P 之间的语法分析树作为语义分析过程的输入。

2、系统详细设计：对如下工作进行展开描述

1、核心数据结构的设计

（1）分析树节点 YufaNode 基类设计：

```
public String node;          节点名
public String nodevalue;     节点属性值：id、digit->value
public ArrayList<YufaNode> chirdlist = new ArrayList<YufaNode>();
子节点序列
public int line;             节点所在的行号
public String type;          语义声明时的 type 属性
public int width;            语义声明时的 width 属性
public String addr;          语义赋值时的 addr 属性
public String code;          语义赋值时的 code 属性(三地址指令)
public String btrue;         语义控制条件为真时的地址
public String bfalse;        语义控制条件为假时的地址
public String snext;         语义控制下一地址
```

（2） 语义分析时存储分析结果的结构

存储三地址指令：

```
public static ArrayList<String>  list3addr = new ArrayList<String>();
```

存储四地址码：

```
public static ArrayList<String>  list4addr = new ArrayList<String>();
```

存储当前的分析语句：

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

存储当前声明变量的类型：

```
public static ArrayList<String>  signkindlist = new ArrayList<String>();
```

存储当前声明变量的 id：（普通变量）

```
public static ArrayList<String>  idlist = new ArrayList<String>();
```

存储当前声明变量的偏移量：

```
public static ArrayList<Integer>  offsetlist = new ArrayList<Integer>();
```

如果当前声明变量是 record，那么 record 与其中声明的其他变量都存储到这个 list 里：

```
public static ArrayList<String>  recordlist = new ArrayList<String>();
```

如果当前声明变量是 proc，那么 proc 与其中声明的其他变量都存储到这个 list 里：

```
public static ArrayList<String>  proclist = new ArrayList<String>();
```

（3） 生成临时变量：

```
public String[] newtemp = {"t0","t1","t2","t3","t4"...};
public String[] newlable = {"L0","L1","L2","L3","L4","L5"...};
```

（4） 更新页面的相关结构

```
Object[][] addr34 = new Object[50][3];
String addr34name[] = {"序号","三地址指令","四元式"};
Object[][] sign = new Object[50][4];
String signname[] = {"识别语句","标识符","类型","偏移量"};
Object[][] error = new Object[50][2];
String errorname[] = {"错误项","错误原因"};
```

2、主要功能函数说明

（1）分析主函数 YuyiAnalyze()

对语法分析树自底向上遍历（已经分割过的小树），如果此时的节点正好是语义节点{an}则执行对应语义动作

（2）变量声明：

```
public int enter(String sentence, String t2, String id1,int w2)
public int enterproc(String sentence, String t2, String id1, int offset2)
public int enterrecord(String sentence, String t2, String id2, int offset2)
```

Sentence ： 当前正在声明的语句

t2 ： 变量的类型

id1 ： 变量 id

w2 ： 变量的偏移量

这个函数首先判断当前符号表是否存在这个 id，如果存在返回-1 主程序进入错误处理报错“重复声明”，否则添加此条记录进入符号表中。

（3）存储三地址指令信息，根据三地址指令计算相应四地址码

```
public void gen(String genstr,String flag)
```

Genstr ： 主程序中组装好的三地址指令

Flag： 当前三地址指令所属的类型，可以为*、+、if、goto、param、call、exit,根据不同类型的规则计算相应四地址码。

将三地址指令存储入 list3addr 中，四地址码存储进 list4addr 中，方便页面更新。

（4）给控制指令中产生的临时跳转地址赋具体的值

```
public String lable(String sbegin)
```

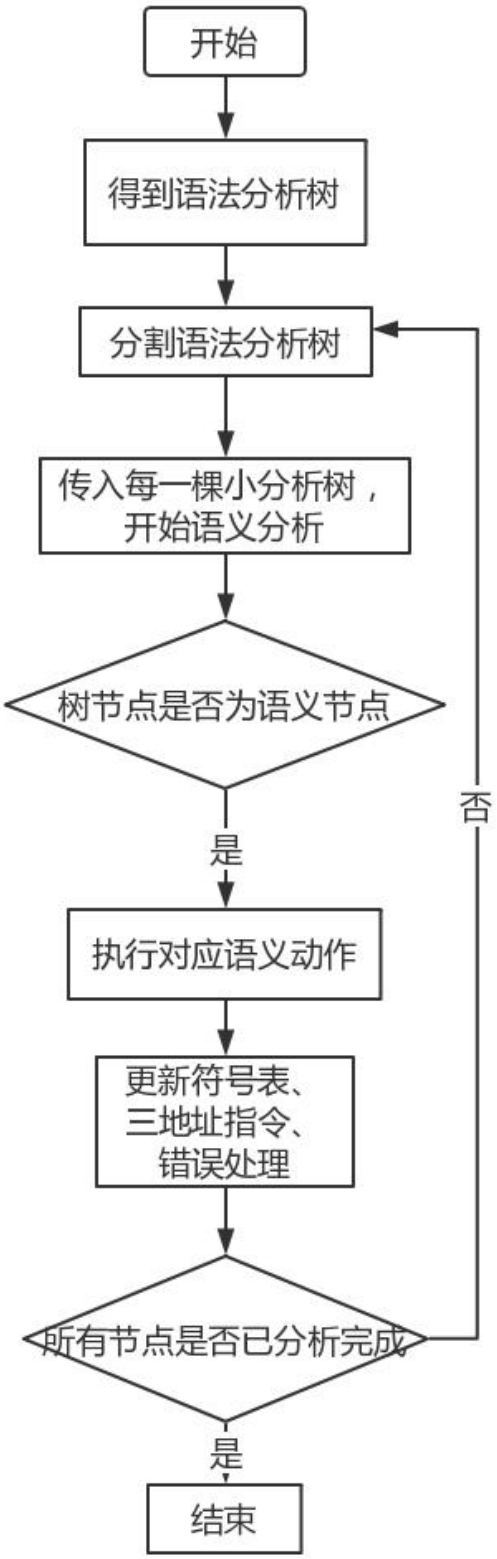
（5）错误处理

private int JudgeIsArray(String id2)：判断是否是数组，如果是数组则返回数组大小方便进行数组是否越界的比较；

public int JudgeSameKind(String i1, String i2, String i3)：判断算术运算时的类型是否匹配,匹配返回 1，否则返回-1

public int JudgeIsState(String id2) 判断此时符号表中是否有该元素，有的话返回 1，否则返回 0 表示 id 未声明

3、程序核心部分的程序流程图



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）各个节点的属性更新很麻烦，应该使用属性栈一边进行语法分析一边进行语义分析；

（2）把语法分析之后得到的结果作为语义分析的输入，直接到分析树进行自底向上的规约是不可行的，这样做的话就会导致程序是从最后一行开始分析了。所以需要进行语法分析树的分割，一个语句的规约过程分割成一棵小树（两个开始符号 P 之间的内容），再对这个小树进行自底向上规约的分析。

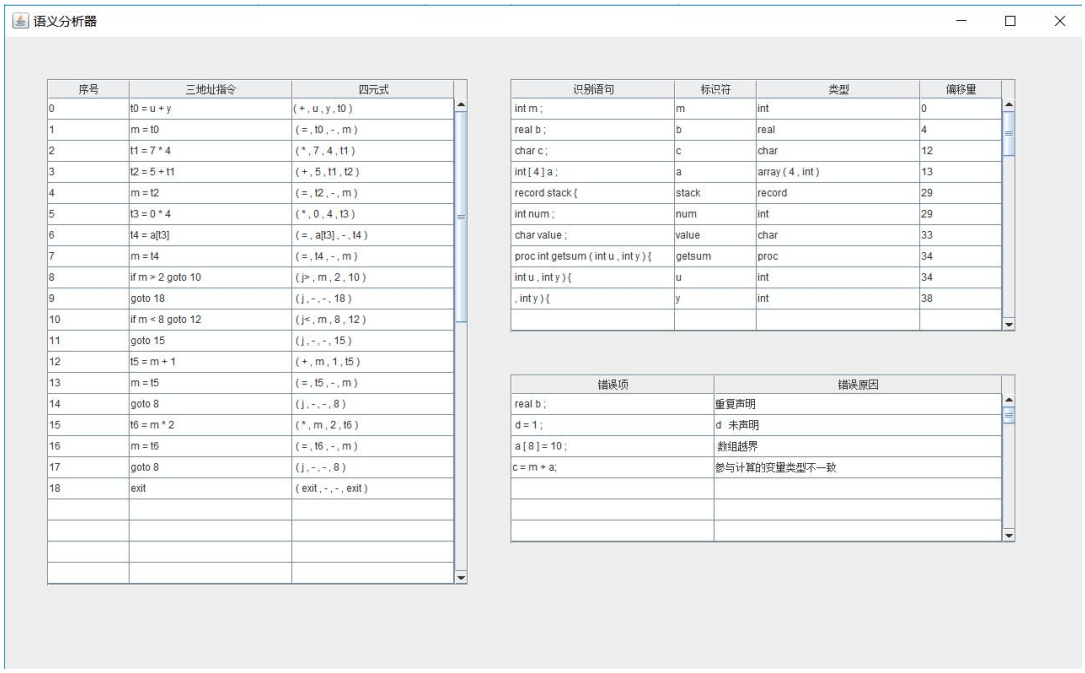
（3）Record,proc,数组需要单独有一张符号表

2、针对一测试程序输出其语义分析结果；

3、输出针对此测试程序经过语义分析后的符号表；

4、输出针对此测试程序对应的语义错误报告；

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



序号	三地址指令	四元式
0	t0 = u + y	(+ , u , y , t0)
1	m = t0	(= , t0 , - , m)
2	t1 = 7 * 4	(* , 7 , 4 , t1)
3	t2 = 5 + t1	(+ , 5 , t1 , t2)
4	m = t2	(= , t2 , - , m)
5	t3 = 0 * 4	(* , 0 , 4 , t3)
6	t4 = a[t3]	(= , a[t3] , - , t4)
7	m = t4	(= , t4 , - , m)
8	if m > 2 goto 10	(j> , m , 2 , 10)
9	goto 18	(j , - , - , 18)
10	if m < 8 goto 12	(j< , m , 8 , 12)
11	goto 15	(j , - , - , 15)
12	t5 = m + 1	(+ , m , 1 , t5)
13	m = t5	(= , t5 , - , m)
14	goto 8	(j , - , - , 8)
15	t6 = m * 2	(* , m , 2 , t6)
16	m = t6	(= , t6 , - , m)
17	goto 8	(j , - , - , 8)
18	exit	(exit , - , - , exit)

识别语句	标识符	类型	偏移量
int m ;	m	int	0
real b ;	b	real	4
char c ;	c	char	12
int [4] a ;	a	array (4 , int)	13
record stack {	stack	record	29
int num ;	num	int	29
char value ;	value	char	33
proc int getsum (int u , int y) {	getsum	proc	34
int u , int y) {	u	int	34
, int y) {	y	int	38

错误项	错误原因
real b ;	重复声明
d = 1 ;	d 未声明
a [8] = 10 ;	数组越界
c = m + a;	参与计算的变量类型不一致

5、对实验结果进行分析。
普通变量声明 int m ;real b;char c；符号表中添加 m,b,c 的相应记录；
数组声明 int[4] a，符号表中添加 array(数组大小，数组类型)；
Record、proc 记录声明，程序中单开 recordlist、proc 记录,整个 record 相当于一个变量。

普通赋值 m = u + y，m = 5+7*4，三地址指令第 0——4 条正确显示；
数组引用 m=a[0]在三地址指令第 5——7 条正确显示；
控制流语句 whie do 和 if then else 在三地址指令第 8——17 条正确显示。

错误处理中，程序第 2、3 行都声明了 real b，所以报错重复声明；
程序第 24 行 d=1，d 并未在符号表中声明，报错未声明变量；
程序第 25 行 a[8] = 10，查找符号表后发现 a 为数组且大小为 4，所以 a[8]发生数组越界；
程序第 26 行 c = m + a，c 为 char 类型，m 为 int 类型，a 是数组，不可进行算术运算，所以报错参与计算的变量类型不一致。

从词法分析到语法分析再到语义分析，程序可以成功生成 token 序列，使用 LL1 文法分析得到正确的语法分析树，一些基本语句可以得到语义识别。三个过程中的常见错误都可以识别报错。

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：