**《编译系统设计实践》**

实验三：语法制导翻译与生成中间代码

学号： 031502248

姓名： 左文航

年级： 15级

学院： 数计学院

专业： 计算机

实验时间：2016－2017学年第一学期

任课教师：刘

目录

[一、实验目的 1](#_Toc503549384)

[二、实验内容 1](#_Toc503549385)

[三、设计思路 1](#_Toc503549386)

[1.算数表达式的翻译 1](#_Toc503549387)

[2.布尔表达式的回填 2](#_Toc503549388)

[3.控制流的回填 3](#_Toc503549389)

[四、代码设计 4](#_Toc503549390)

[函数设计 4](#_Toc503549391)

[五、实验结果 5](#_Toc503549392)

[第一组测试样例 5](#_Toc503549393)

[第二组测试样例 6](#_Toc503549394)

[第三组测试样例 6](#_Toc503549395)

[第四组测试样例 7](#_Toc503549396)

[第五组测试样例 8](#_Toc503549397)

[六、实验心得 9](#_Toc503549398)

[七、附 10](#_Toc503549399)

[1. void work3\_trainslate(int state, int k) 10](#_Toc503549400)

[2. void backpatch(string truelist, int M) 16](#_Toc503549401)

## 一、实验目的

通过语法制导或翻译模式生成中间代码。

## 二、实验内容

理解并应用课本的语法制导定义。

**参考课本264页图6-43，课本267页图6-46。**

**或者课本258页图6-36，图6-37。**

在自底向上语法分析基础上设计语义规则（语法制导翻译），将源程序翻译为四元式输出，若有错误将错误信息输出。

实验报告必须包括设计的思路，以及测试报告（输入测试例子，输出结果）。

## 三、设计思路

利用实验二的自底向上的LR(1)文法,当进行归约时,依次判断当前处理的归约文法句子是啥,然后利用实现写好的SDT,再用栈的方式处理当前的number,ident,中间变量，等来与SDT对应,实现算数表达式翻译、布尔表达式的回填、控制流语句的回填。

### 1.算数表达式的翻译

1.(13)

statement → ident := expression

statnment.nextlist = nextlist

statement.code = gen(top.get(ident.lexme) '=' expression.addr)

2.(5)

const-assignment-list → ident = number $

const-assignment-list.code = gen(top.get(ident.lexme) '=' top.get(number.lexme))

3.(6)

const-assignment-list → const-assignment-list , ident = number $

gen(top.get(ident.lexme) '=' top.get(number.lexme))

4.(31)

expression → expression1 adding-operator term

expression.addr = new Temp()

gen(expression.addr ‘=’expression1.addr adding-operator term.addr

5.(35)

term → term1 multiplying-operator factor

term.addr = new Temp()

gen(term.addr = term.addr multiplying-operator factor.addr)

6.(29)

expression → term $

expression.addr = top.get(id.lexeme)

7.(30)

expression → adding-operator term $

expression.addr = new Temp()

gen(E.addr '=' 'minus' term.addr)

8.(34)

term → factor

term.addr = factor.addr

9.(38)

factor → ident $

factor.addr = ident.addr

10.(39)

factor → number $

factor.addr = number.addr

11.(40)

factor → ( expression ) $

factor.addr = expression.addr

12.(32)

adding-operator → + $

adding-operator.addr = +

13.(33)

adding-operator → - $

adding-operator.addr = -

14.(36)

multiplying-operator → \* $

multiplying-operator.addr = \*

15.(37)

multiplying-operator → / $

multiplying-operator.addr = /

### 2.布尔表达式的回填

16.(21)

condition → expression1 relation expression2 $

condition.truelist = makeist()

gen('if' expression1.addr relation.addr expression2.addr 'goto' condition.true)

gen('goto' condition.false)

17.(23)

relation → = $

relation.addr = =

18.(24)

relation → <> $

relation.addr = <>

19.(25)

relation → < $

relation.addr = <

20.(26)

relation → > $

relation.addr = >

21.(27)

relation → <= $

relation.addr = <=

22.(28)

relation → >= $

relation.addr = >=

### 3.控制流的回填

23.(16)

statement → if condition then M2 statement1 M1 $

backpath(condition.truelist,M.instr2)

backpath(condition.falselist,M.instr1)

statement.nextlist = merge(condition.falselist,statement1.nextlist)

24.(17)

statement → while condition do statement1 $

begin = newlabel()

condition.true = newlabel()

condition.false = statement.label()

statement1.next = begin

gen("goto" begin)

25.(41)

M → ε $

M.instr = nextlist

26.(15)

statement → begin statement-list end $

statement.nextlist = statment-list.nextlist;

27.(19) statement-list → statement $

28.(20) statement-list → statement-list ; statement $

statement-list.nextlist = statement.nextlist

29.(17)

statement → while M1 condition M2 do statement1 M3 $

backpatch(statement1.nextlist,M1.instr)

backpatch(condition.truelist,M2.instr)

backpatch(condition.falselist,M3.instr + 1)

statement.nextlist = condition.falselist

gen('goto' M1.instr)

## 四、代码设计

### 函数设计

void work3\_init()

初始化变量

void work3\_open\_file()

打开本次作业需要用到的文件

void work3\_close\_file()

关闭本次作业需要用到的文件

void work3\_read()

读入需要分析的代码

int Temp()

生成新的中间变量

void insert\_num(char \*str,int num)

对输出的字符串串进行插入操作,

当num<0时,表示插入的是id[-num],

当num>10000时,表示插入的是数字number[num-10000]

其他情况下,插入的是 中间变量 tnum.

void work3\_solve()

对输入的文法进行自低向上分析

void work3\_trainslate(int state,int k)

在规约时,对归约的语句state和当前输入第k个字符进行,进行翻译和中间代码的生成

void backpatch(string truelist,int M)

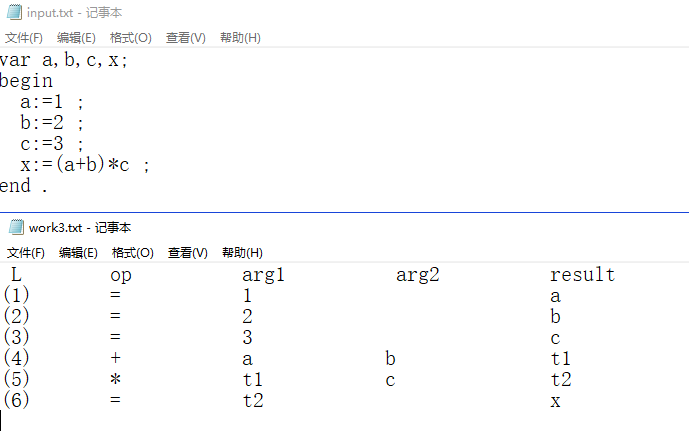
对于链表 truelist 里所有的 四元式进行回填 ,填入M

void work3\_Print()

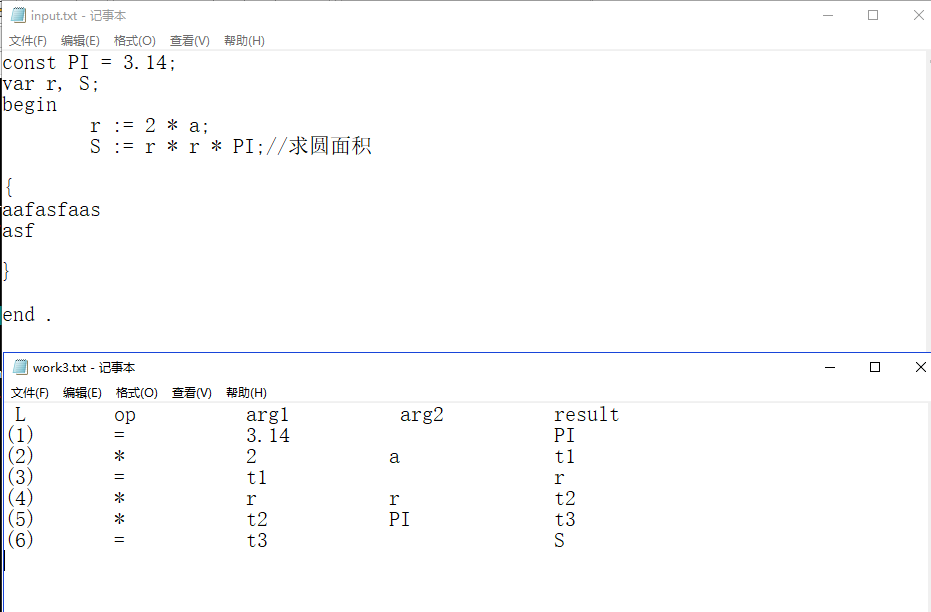
输出四元式

## 五、实验结果

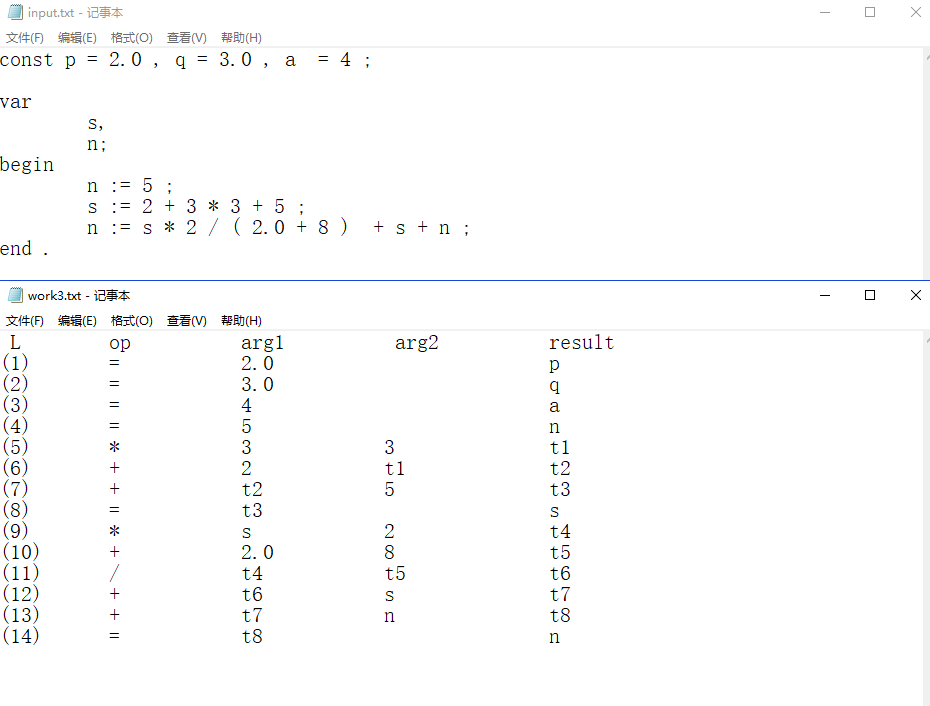
### 第一组测试样例



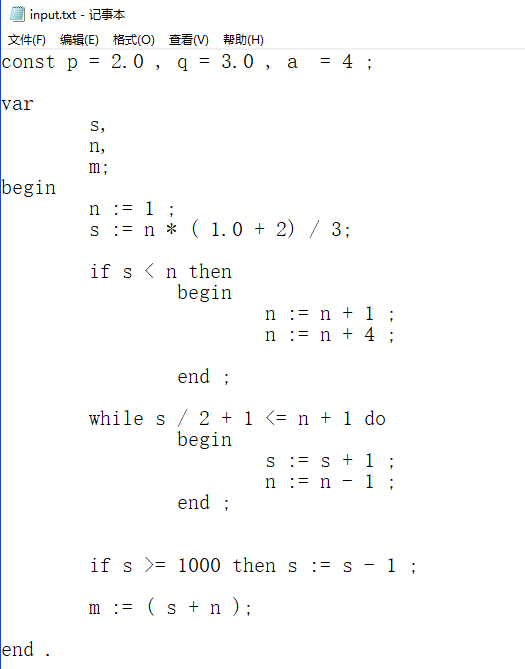
### 第二组测试样例

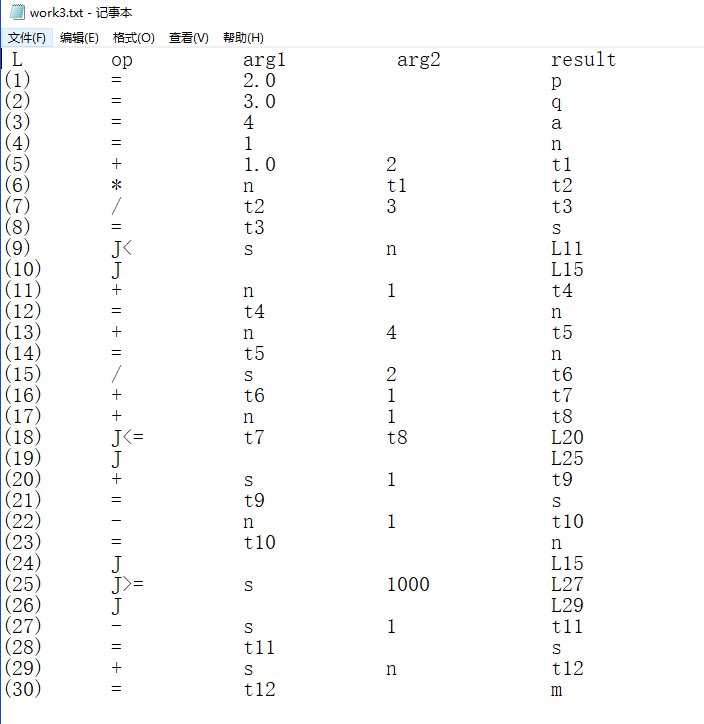


### 第三组测试样例

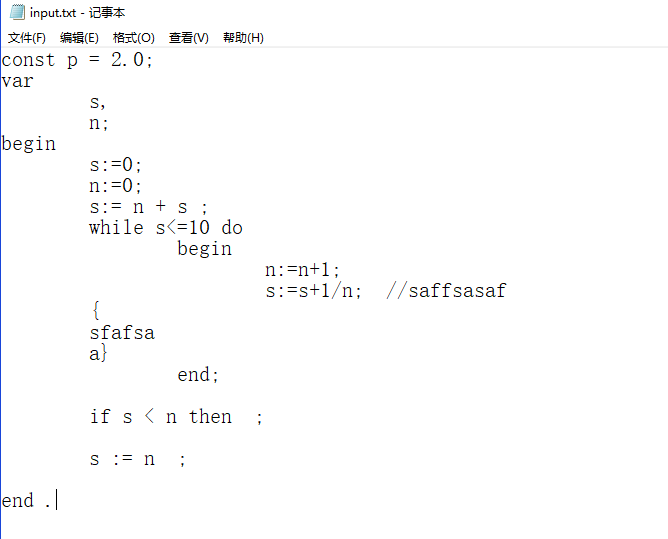


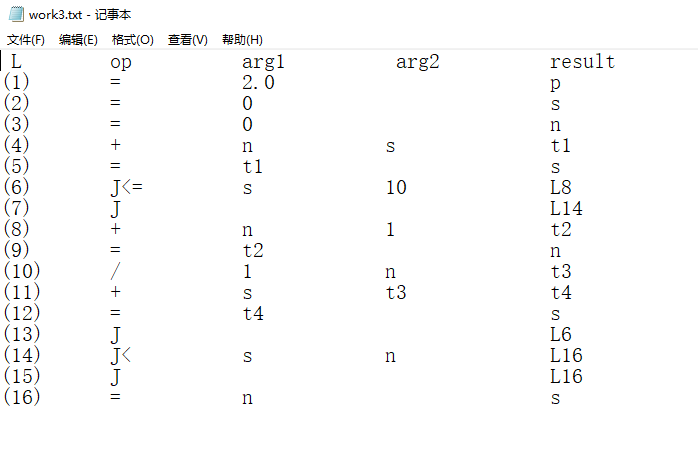
### 第四组测试样例





### 第五组测试样例





## 六、实验心得

这个实验相较前两个实验虽然在代码量上没有第二个实验大,但是思考难度更大，特别是在理论课还没有教的情况下，硬着头皮把需要用到的理论知识预习完，有因为上课主要讲的是SDD,然而在LR(1)文法的情况下,SDD开发难度更大,因此在做实验时,先尝试了SDD，但是始终无法实现code属性的连接,又因为是一个人做实验,越做心里越急,好在后面回头是岸,采用了SDT的方法,利用回填和C++的string,很简单的就实现了字符串的连接,在一步一步摸索和测试终于将这个实验完成。

这次实验让我对SDD与SDT,继承属性和综合属性有了更深一步的了解,再加上能把三次实验的代码整合起来,大概写了1700+行的代码,输入一个PASCAL语言的代码,就能直接生成一个四元式,全程实验都是自己独立完成的,也让自己觉得实验做的十分的有意义和满足感。

## 七、附

核心代码如下:

## 1. void work3\_trainslate(int state, int k)

void work3\_trainslate(int state, int k)

{

int op = 10, arg1 = 22, arg2 = 35, result = 50;

char \*str = work3\_p[p\_len];

int id = k, num = k, len;

if (state == 5 || state == 6) //赋值 id = num

{

while (strlen(read\_id[id]) == 0)--id;

while (strlen(read\_num[num]) == 0) --num;

//op

str[op] = '=';

//arg1

len = strlen(read\_num[num]);

for (int i = 0; i<len; ++i) str[i + arg1] = read\_num[num][i];

//result

len = strlen(read\_id[id]);

for (int i = 0; i<len; ++i) str[i + result] = read\_id[id][i];

read\_id[id][0] = '\0';

p\_len++;

}

else if (state == 13) // id = expression;

{

string next = ""; next += p\_len;

statement\_nextlist.push(next);

//op

str[op] = '=';

//arg1

int num = expr.top(); expr.pop();

str[arg1] = 't';

insert\_num(str + arg1 + 1, num);

//result

while (strlen(read\_id[id]) == 0)--id;

len = strlen(read\_id[id]);

for (int i = 0; i<len; ++i) str[i + result] = read\_id[id][i];

read\_id[id][0] = '\0';

p\_len++;

}

else if (state == 31) // expression1 adding-operator term

{

int t1 = Temp();

int t2 = expr.top(); expr.pop();

int o = oper.top(); oper.pop();

int t3 = term.top(); term.pop();

str[op] = String[o][0];

//arg1

str[arg1] = 't';

insert\_num(str + arg1 + 1, t2);

//arg2

str[arg2] = 't';

insert\_num(str + arg2 + 1, t3);

//result

str[result] = 't';

insert\_num(str + result + 1, t1);

expr.push(t1);

p\_len++;

}

else if (state == 35) //term1 multiplying-operator factor

{

int t1 = Temp();

int t2 = term.top(); term.pop();

int o = oper.top(); oper.pop();

int t3 = factor.top(); factor.pop();

str[op] = String[o][0];

//arg1

str[arg1] = 't';

insert\_num(str + arg1 + 1, t2);

//arg2

str[arg2] = 't';

insert\_num(str + arg2 + 1, t3);

//result

str[result] = 't';

insert\_num(str + result + 1, t1);

term.push(t1);

p\_len++;

}

else if (state == 29) //expression = term

{

int t = term.top(); term.pop();

expr.push(t);

}

else if (state == 30) //expression = adding-operator term

{

int t1 = Temp();

int o = oper.top(); oper.pop();

int t2 = term.top(); term.pop();

str[op] = String[o][0];

//arg1

str[arg1] = 't';

insert\_num(str + arg1 + 1, t2);

//result

str[result] = 't';

insert\_num(str + result + 1, t1);

expr.push(t1);

strcpy(work3\_p[p\_len++], str); put\_i++;

}

else if (state == 34)// term.addr = factor.addr

{

int t = factor.top(); factor.pop();

term.push(t);

}

else if (state == 38)// factor.addr = ident.addr

{

while (strlen(read\_id[id]) == 0)--id;

factor.push(-id);

}

else if (state == 39)

{

while (strlen(read\_num[num]) == 0)--num;

factor.push(num + 10000);

}

else if (state == 40)

{

int t = expr.top(); expr.pop();

factor.push(t);

}

else if (state == 32 || state == 33 || state == 36 || state == 37) //op = +-/\*

{

int o = k;

while (read\_op[o] == 0) --o;

oper.push(read\_op[o]);

read\_op[o] = 0;

}

else if (state >= 23 && state <= 28)

{

int re = G[state][1];

rela.push(re);

}

else if (state == 21) // E1 rep E2

{

string truelist = "";

truelist += p\_len;

string falselist = "";

falselist += p\_len + 1;

condition\_truelist.push(truelist);

condition\_falselist.push(falselist);

int rel = rela.top(); rela.pop();

int t2 = expr.top(); expr.pop();

int t1 = expr.top(); expr.pop();

//op

str[op] = 'J';

int len = strlen(String[rel]);

for (int i = 0; i < len; ++i) str[op + i + 1] = String[rel][i];

//arg1

str[arg1] = 't';

insert\_num(str + arg1 + 1, t1);

//arg2

str[arg2] = 't';

insert\_num(str + arg2 + 1, t2);

//result

str[result] = 'L';

p\_len++;

// return;

//goto L

str = work3\_p[p\_len];

str[op] = 'J';

str[result] = 'L';

// insert\_num(str+result+1,p\_len);

p\_len++;

}

else if (state == 41) //M.instr = nextlist

{

M.push(p\_len);

}

else if (state == 16)//statement = if condition then M statement1 M $

{

string truelist = condition\_truelist.top(); condition\_truelist.pop();

string falselist = condition\_falselist.top(); condition\_falselist.pop();

int M\_list1 = M.top(); M.pop();

int M\_list2 = M.top(); M.pop();

backpatch(truelist, M\_list2);

backpatch(falselist, M\_list1);

string s\_nextlist1 = statement\_nextlist.top(); statement\_nextlist.pop();

string s\_nextlist = falselist + s\_nextlist1;//merge

statement\_nextlist.push(s\_nextlist);

}

else if (state == 13 || state == 16)

{

string next = ""; next += p\_len;

statement\_nextlist.push(next);

}

else if (state == 15)

{

string statement\_list = statement\_list\_nextlist.top(); statement\_list\_nextlist.pop();

statement\_nextlist.push(statement\_list);

}

else if (state == 19 || state == 20)

{

string list = statement\_nextlist.top(); statement\_nextlist.pop();

statement\_list\_nextlist.push(list);

}

else if (state == 18)

{

string ss = "";

int a = 0;

ss += a;

statement\_nextlist.push(ss);

}

else if (state == 17) //statement = while M1 condition M2 do statement1 M3 $

{

string statement1 = statement\_nextlist.top(); statement\_nextlist.pop();

string ctruelist = condition\_truelist.top(); condition\_truelist.pop();

string cfalselist = condition\_falselist.top(); condition\_falselist.pop();

int M3 = M.top(); M.pop();

int M2 = M.top(); M.pop();

int M1 = M.top(); M.pop();

backpatch(statement1, M1);

backpatch(ctruelist, M2);;

backpatch(cfalselist, M3 + 1);

statement\_nextlist.push(cfalselist);

cout << "M1 = " << M1 << endl;

char \*str = work3\_p[p\_len];

str[op] = 'J';

str[result] = 'L';

insert\_num(str + result + 1, M1);

p\_len++;

}

}

## 2. void backpatch(string truelist, int M)

//回填,truelist填入 goto M

void backpatch(string truelist, int M)

{

int q = M;

int op = 10, arg1 = 22, arg2 = 35, result = 50;

for (int i = 0; i < truelist.length(); ++i)

{

int p = (int)truelist[i];

char \*str = work3\_p[p] + result;

insert\_num(str + 1, q);

}

}