



华南师范大学

本科学生实验（实践）报告

测试报告

院 系：计 算 机 学 院

实验课程：编译原理

实验项目：TINY 扩充语言的语法树生成

指导老师：黄煜廉

开课时间：2023~2024 年度第一学期

专 业： 计算机科学与技术

班 级：计算机科学与技术 3 班

学生姓名：卢泓钢

华南师范大学教务处

一、测试数据组织的说明：

根据实验 3 的要求进行测试数据的组织，所编制的源程序需要对以下的功能进行测试。

1.实现改写书写格式的新 if 语句；

2.增加 for 循环；

3.扩充算术表达式的运算符：**+=** 加法赋值运算符(类似于 C 语言的+=)、求余%、乘方^，

4.扩充扩充比较运算符：**>**(大于)、**<=**(小于等于)、**>=**(大于等于)、**<>**(不等于)等运算符，

5.增加正则表达式，其支持的运算符有：**或(|)**、**连接(&)**、**闭包(#)**、**括号()**、**可选运算符(?)**和基本正则表达式。

6.增加位运算表达式，其支持的位运算符有 **and(与)**、**or (或)**、**not(非)**，如果对位运算不熟悉，可以参考 C/C++的位运算。

书写格式的修改

if 语句的修改

对于 if 语句，老师要求的语法规则为 `if_stmt --> if(exp) stmt-sequence else stmt-sequence | if(exp) stmt-sequence`，但是这样的语法规则会导致 if 语句的嵌套出现歧义，例子如下：

```
y = 1;
if (x >= 0)
if (x <= 10)
y = 2
else
y = 3
```

对于以上代码，根据以上语法规则，我们无法确定 **else** 语句是属于哪个 **if** 语句的，有以下两种情况：

```
{情况一}
y = 1;
if (x >= 0)
    if (x <= 10)
```

```

        y = 2
else
    y = 3

{情况二}
y = 1;
if (x >= 0)
    if (x <= 10)
        y = 2
    else
        y = 3

```

因此我将其改写为 `if-stmt --> if (exp) [stmt-sequence] [else [stmt-sequence]]`
 用终结符号方括号 ([]) 括住语句序列，这样就能解决嵌套歧义的问题。

将 if 测试数据由：

```

x:=1;
if (0<x)
    x:=x+1;
    x:= x*x
else
    x:=x+2;
    x:= x*x*x

```

改为：

```

x:=1;
if (0<x) [
    x:=x+1;
    x:= x*x
] else [
    x:=x+2;
    x:= x*x*x
]

```

对正则表达式及其赋值语句的修改

对于正则表达式的赋值，若按照老师的语法规则 `ID:=正则表达式`，会导致正则表达式的赋值与算术表达式的赋值出现歧义，例子如下：

```

exp := a;
reg := a

```

单一字符 `a` 既可以是算术表达式，也可以是正则表达式，无法判断当前赋值语句是正则表达式赋值还是普通表达式赋值，因此我将正则表达式的赋值改写为 `ID ::=`

正则表达式，这样就能解决赋值歧义的问题。

该程序正则表达式的文法规则如下图所示：

正则表达式

reg-exp --> reg-and-exp { | reg-and-exp }

reg-and-exp --> reg-term { & reg-term }

reg-term --> reg-factor { regop }

regop --> # | ?

reg-factor --> (reg-exp) | letter

可见对于该程序来说，正则表达式的元字符仅有英文字母，因此我将给出的正则表达式测试数据进行一些调整：

将原来的： $y:=((+|-)?&d&d\#)&(.&d&d\#)?&((E|e)&((+|-)?&d&d\#))?$

改为： $y::=((a|b)?&d&d\#)&(p&d&d\#)?&((E|e)&((a|b)?&d&d\#))?$

因此，最后组织形成的测试数据（源程序）：

{ 下面程序段是测试修改书写格式后的 if 语句 }

```
x:=1;
if (0<x) [
    x:=x+1;
    x:= x*x
] else [
    x:=x+2;
    x:= x*x*x
]
```

{ 下面程序段是测试 for 循环 }

```
for fact := x downto 1 do
```

```

    fact := fact * x;
    x:=x+1
enddo;

```

```

for fact := 1 to x do
    fact := fact * x;
    x:=x-1
enddo

```

{ 下面程序段是测试 += % ^ }

```

x+=x %2 +3^2;

```

{ 下面程序段是测试>(大于)、<=(小于等于)、>=(大于等于)、<>(不等于)等运算符 }

```

if (x>0) [x:=1];
if (x<=0) [x:=1];
if (x>=0) [x:=1];
if (x<>0) [x:=1]

```

{ 下面程序段是测试正则表达式，其运算符有： 或(|) 、连接(&)、闭包(#)、括号() 、可选运算符(?)和基本正则表达式。 }

```

y::=((a|b)?&d&d#)&(p&d&d#)?&((E|e)&((a|b)?&d&d#))?

```

{ 下面程序段是测试位运算表达式，其运算符有 and(与)、or (或)、 not(非) }

```

z:= 2 and 3 or 5 and not 5;

```

```

x:=5;
if( x>=1) [
    for fact := x downto 1 do
        fact := fact * x;
        if (fact<>5) [y:=10] else [y:=20];
        for z := 1 to y do
            x:= 2 +3 and 3*4 or 5%6^2 and not 5 * (7+9);
            x+=1-x
        enddo
    enddo
] else [
    w::= ((a|b)?&d&d#)&(p&d&d#)?&((E|e)&((a|b)?&d&d#))?

```

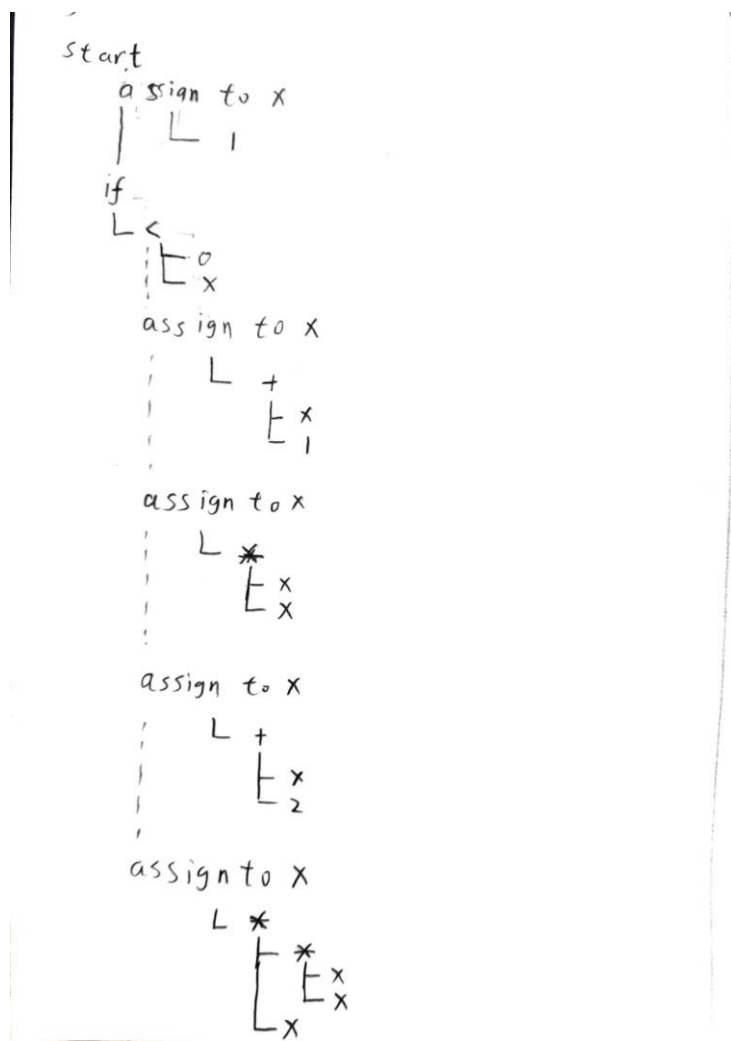
]

二、各功能的测试结论【本次测试只测试正确的源程序】

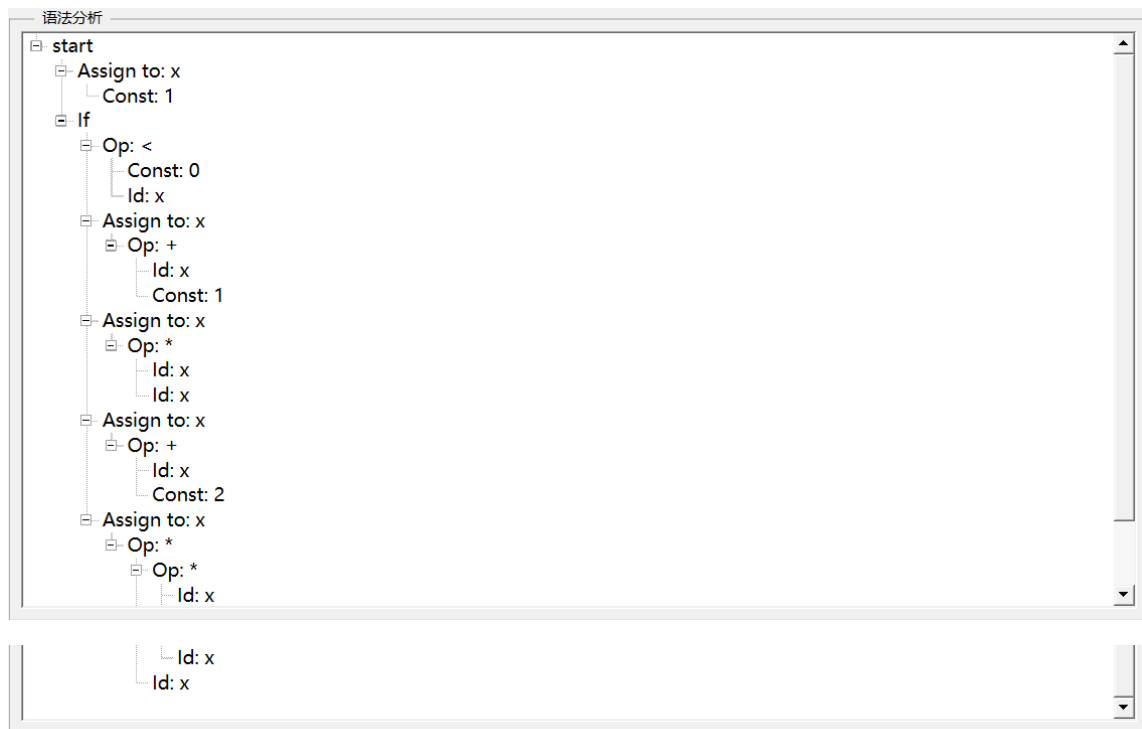
功能 1: 改写书写格式后的新 if 语句

预期结果:

为了省略篇幅, 以下测试的预期结果均使用简写



程序执行结果:



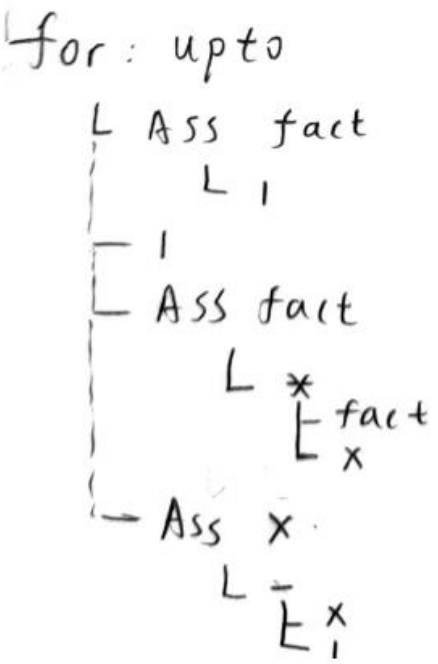
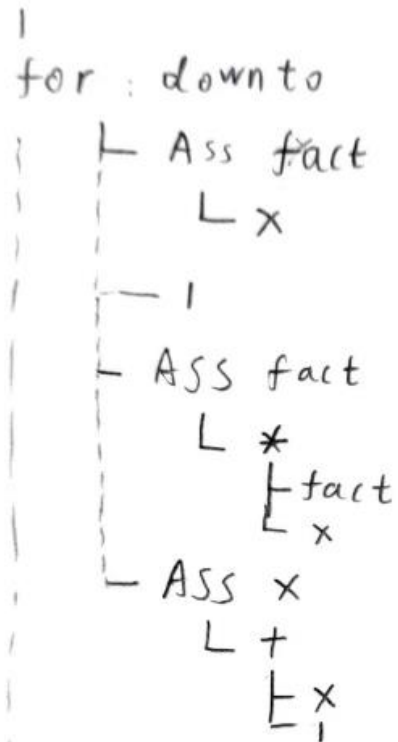
结论：程序基本上能模仿原生 TINY 语言 `printTree()` 函数的语法树生成，但是不足之处是没有显示出 `else` 字段的层次。

功能 2：for 循环

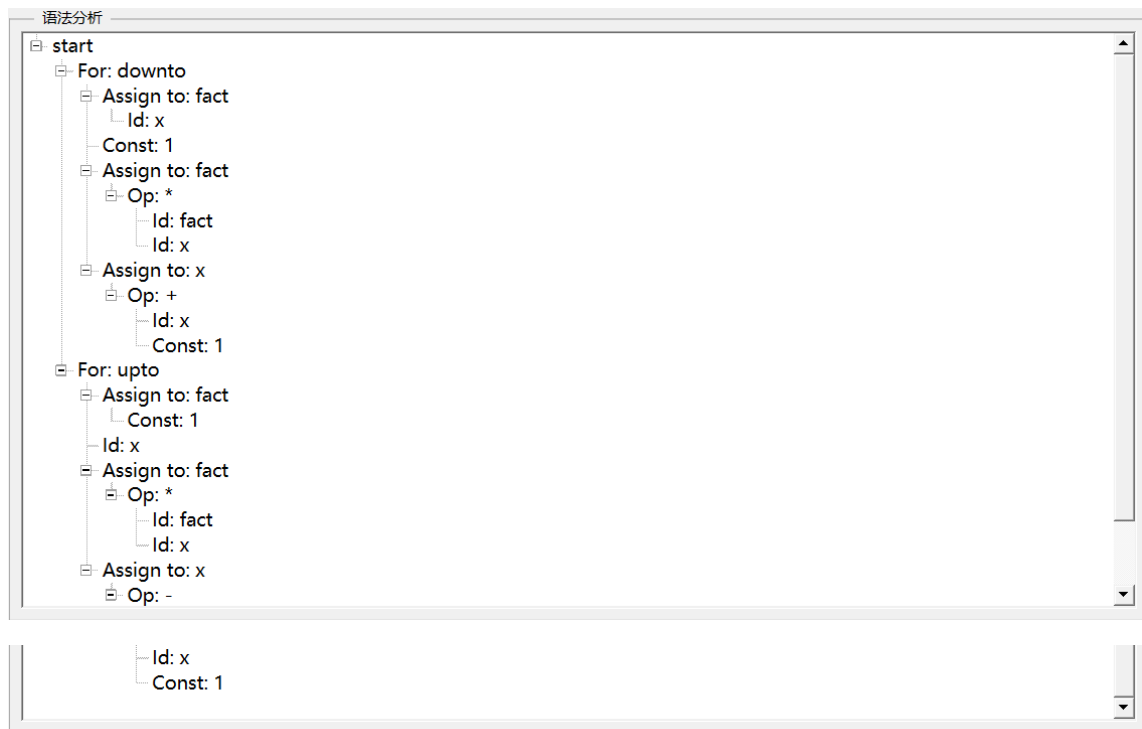
预期结果：

(2)

start



程序执行结果:



结论：程序能正常生成 for 循环语句的语法树。

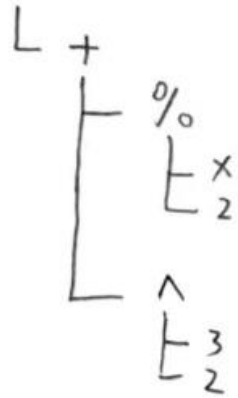
功能 3：扩充算术表达式的运算符：+= 加法赋值运算符（类似于 C 语言的+=）、求余%、乘方^，

预期结果：

13)

start

└ add Assign to x



程序执行结果:



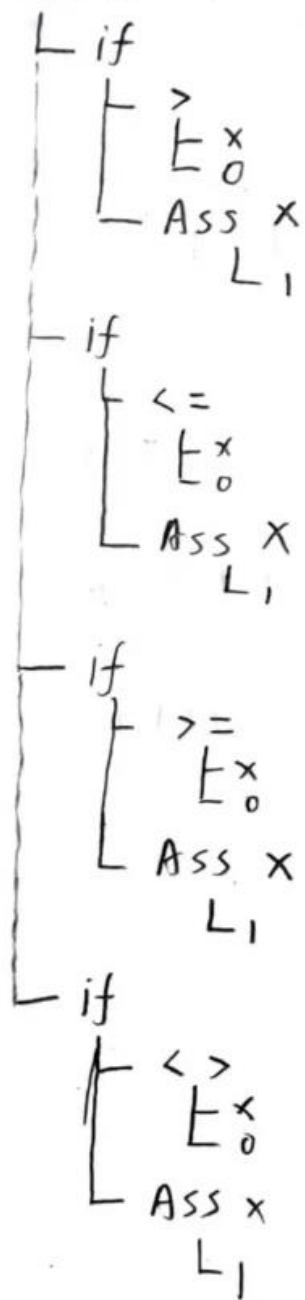
结论：程序能正确生成%与^的语法树。

功能 4：比较运算符的扩充：=（等于），>（大于）、<=（小于等于）、>=（大于等于）、<>（不等于）等运算符

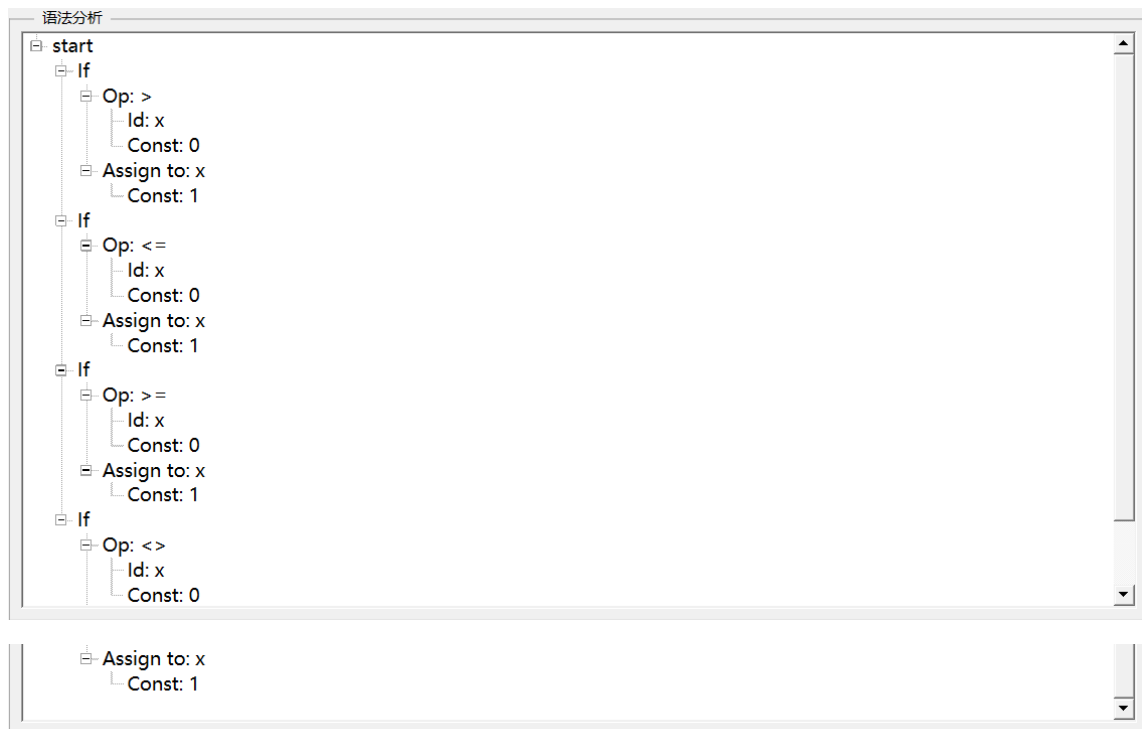
预期结果:

(4)

start



程序执行结果:



结论：程序能正确生成各种比较运算符的语法树。

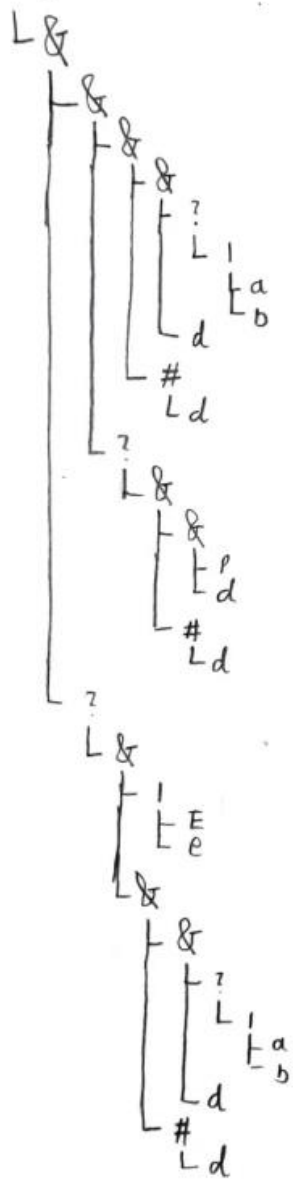
功能 5：正则表达式，其支持的运算符有： 或(|) 、连接(&)、闭包(#)、括号() 、可选运算符(?)和基本正则表达式。

预期结果：

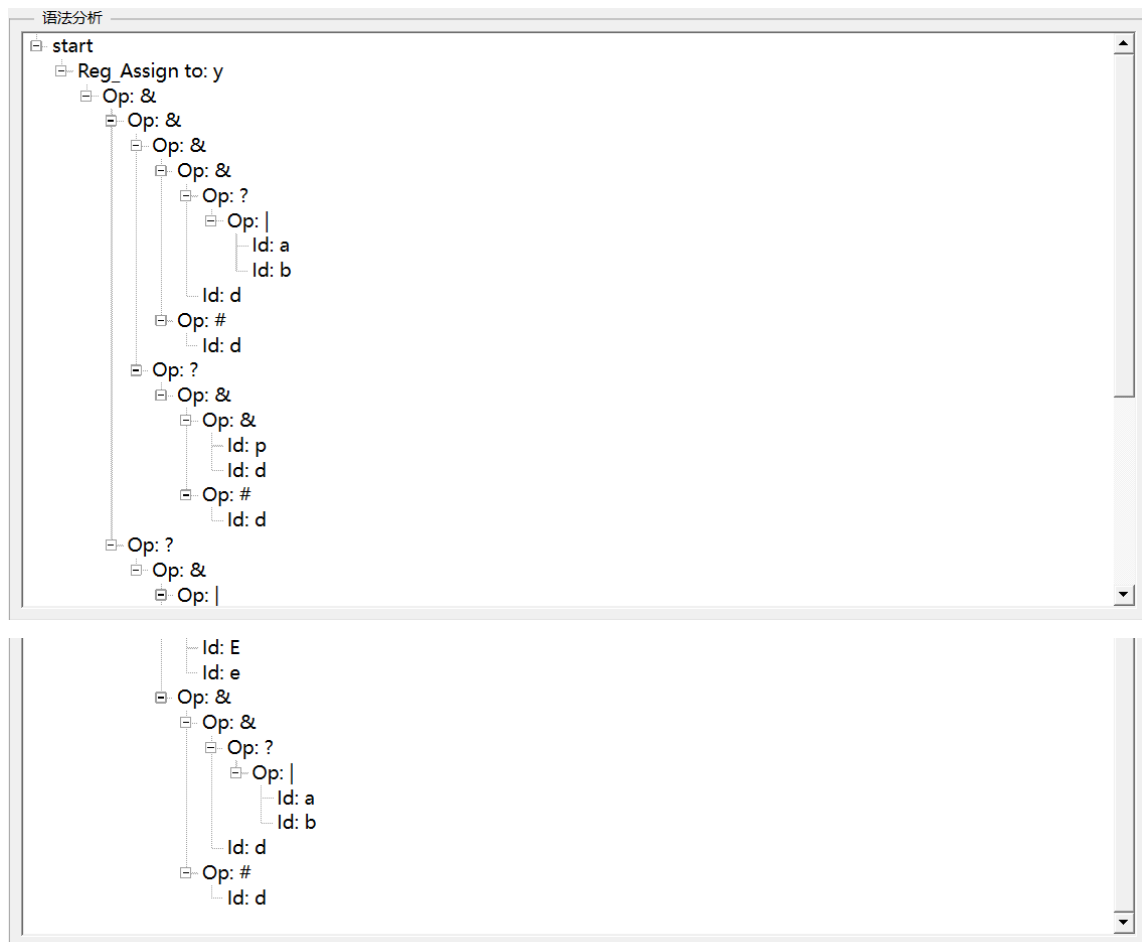
(4)

start

L Reg Assign to: y



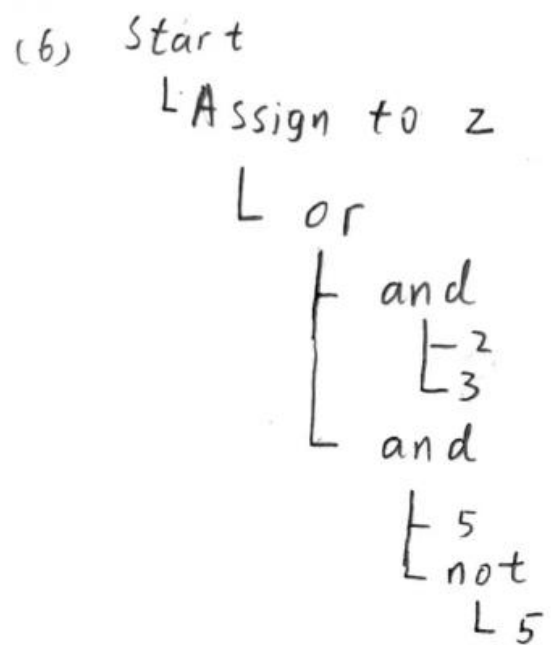
程序执行结果:



结论：经仔细比对，可发现生成的语法树正确，能正确地反映各符号间的优先级顺序。

功能 6：位运算表达式，其支持的位运算符号有 and(与)、or（或）、 not(非)

预期结果：



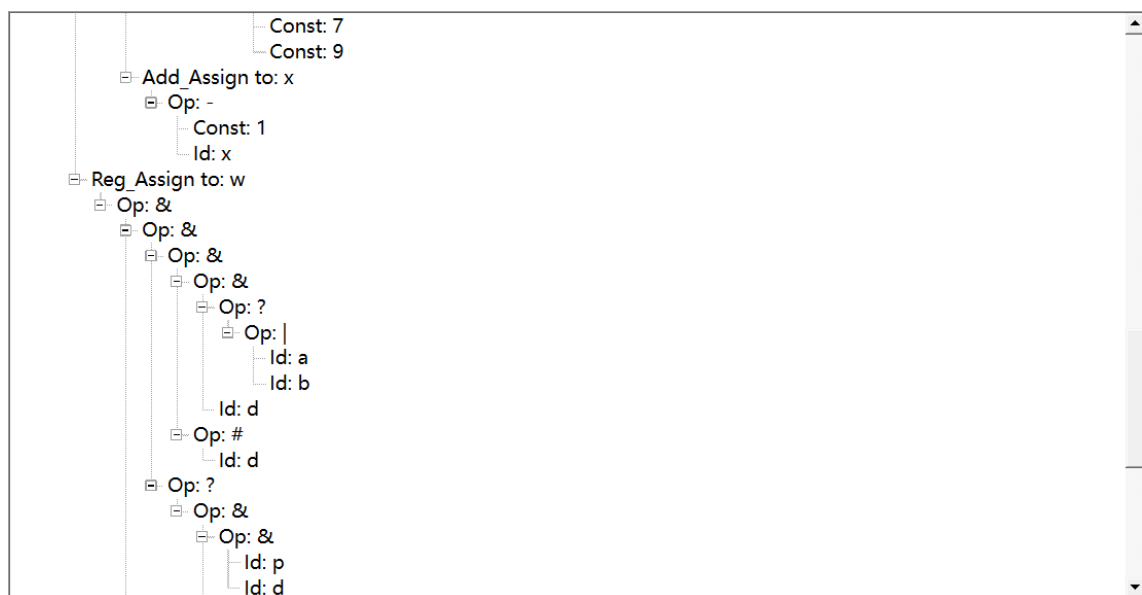
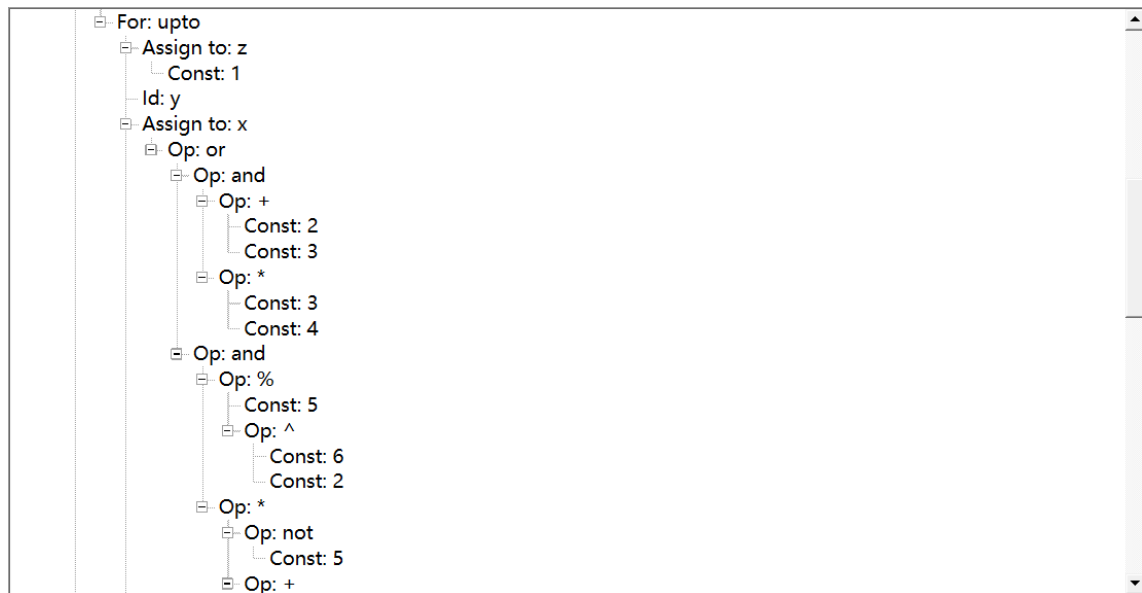
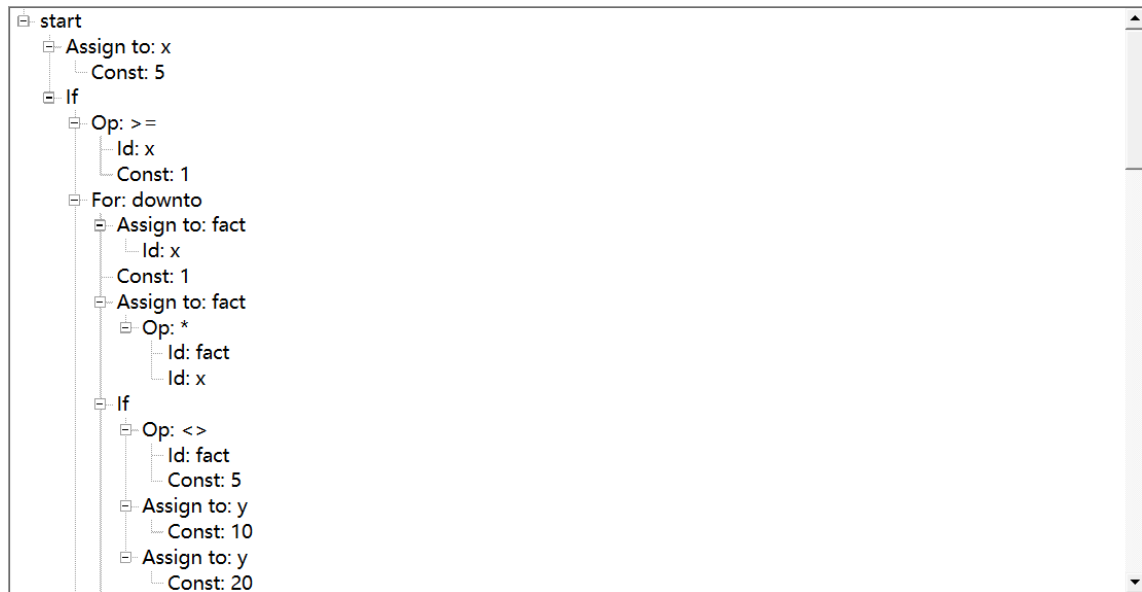
程序执行结果：



结论：生成语法树正确，能正确反映位运算符优先级。

功能 7：混合功能测试

预期结果：





结论：经仔细比对，程序可对混合功能代码生成正确的语法树，不足之处同样是没有显示出 **else** 字段的层次。

最终结论：程序能对各种情况的代码生成正确的语法树，不足之处为没有显示出 **if** 语句中 **else** 字段的层次。

三、通过测试结论对实验 3 的自评

根据测试的结论来对实验 3 的实现情况的自评分数以及原因说明

【上述功能测试，功能 1~6 的测试各占 12 分，共 72 分；功能 7 的测试占 28 分（其中 2 个 **if** 语句功能共占 6 分，2 个 **for** 语句共功能占 6 分，其他的 4 个子功能各 4 分）】

功能 1：7 分，**if** 语句没有显示出 **else** 字段的层次。

功能 2：12 分，正确生成 **for** 语句语法树

功能 3：12 分，正确生成 **%** 与 **^** 运算符的语法树

功能 4：12 分，正确生成比较运算符语法树

功能 5: 12 分, 正确生成正则表达式及其赋值的语法树, 能正确显示其优先级顺序

功能 6: 12 分, 正确生成位运算符语法树

功能 7: if 语句功能 4 分, for 语句功能 6 分, 其他子功能都能完整实现, 16 分,
 $4+6+16=26$ 分

总分: 93