

**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（测试报告）**

**指导老师：黄煜廉**

**开课时间：2023 ～ 2024年度第 1学期**

**专 业：计算机科学与技术**

**班 级：计科1班**

**学 生：李达良**

**学 号：20203231004**

**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号：+= 加法赋值运算符号（类似于C语言的+=）、求余%、乘方^，

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

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

6.增加位运算表达式，其支持的位运算符号有 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。

【注意事项：】

1.对于下面的源程序，如果你在实现实验3的时候，相应的书写方法是有不同之处的，请进行相应的详细说明，并根据你的书写方法进行源程序某些书写格式的修改。

2.如果程序的结果显示窗口对于语句比较多的源程序的显示处理不理想（或显示结果浏览不方便）的话，你可以把下面的源程序进行分段式的输入并查看相应的测试结果。

因此，最后组织形成的测试数据（源程序）：【本次测试只测试正确的源程序】

{ 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)等运算符号 }

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，其运算符号有： 或(|) 、连接(&)、闭包(#)、括号( ) 、可选运算符号（?）和基本正则表达式。 }

y:=((+|-)?&d&d#)&(.&d&d#)?&((E|e)&((+|-)?&d&d#))?;

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

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

{下面的程序段是混合式的功能测试 }

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:= ((+|-)?&d&d#)&(.&d&d#)?&((E|e)&((+|-)?&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

说明：if语句在本实验中设置的文法格式是：if\_stmt-->if(exp) [ stmt-sequence [[ else stmt-sequence]] ]（红色的[]代表不是EBNF语法的[]），也就是说，if和else的段里面使用中括号括起来。修改后的源代码如下：

{ 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

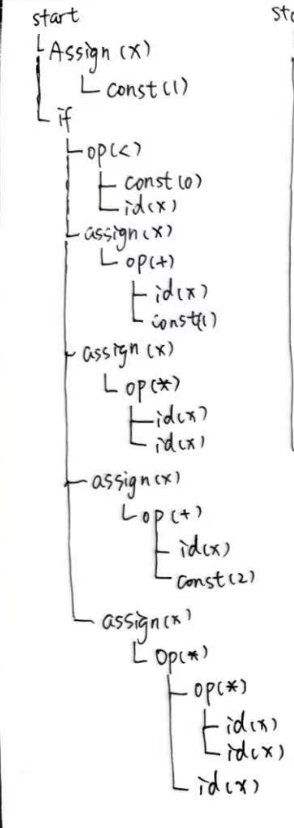
]else[

x:=x+2;

x:= x\*x\*x

]

预期结果：



程序执行结果：



结论：符合预期结果，生成正确

功能2：for循环

说明：样例源程序错误，少了个分号：

{ 下面程序段是测试for循环 }

for fact := x downto 1 do

fact := fact \* x;

x:=x+1

enddo;

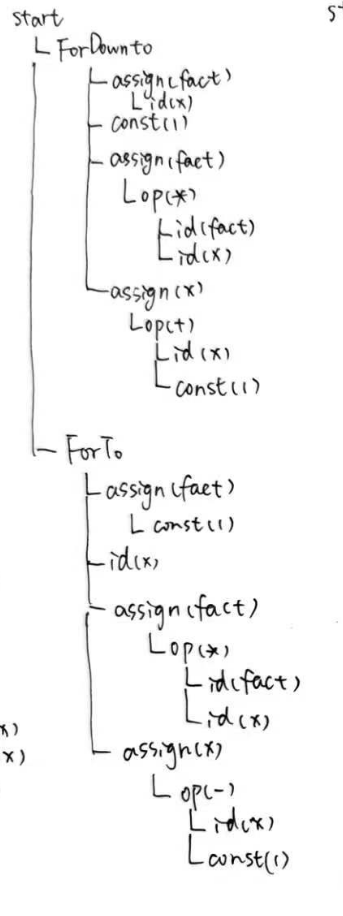
for fact := 1 to x do

fact := fact \* x;

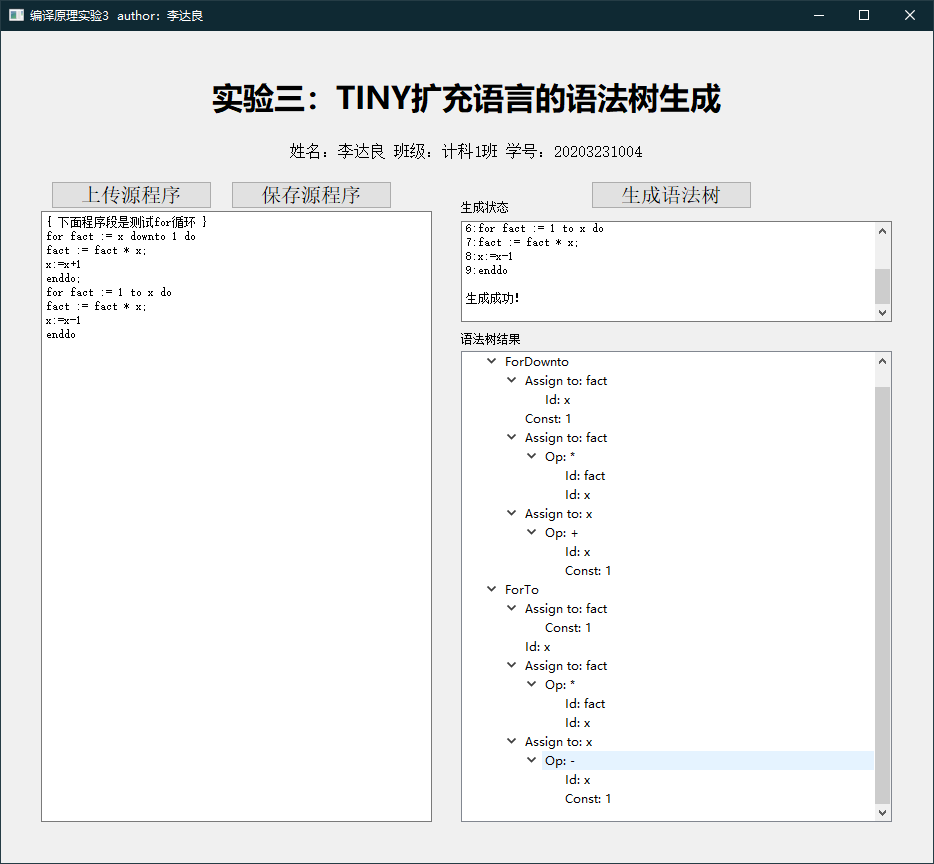
x:=x-1

enddo

预期结果：



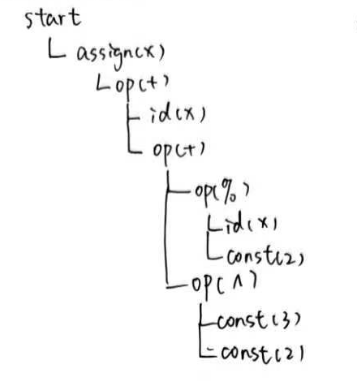
程序执行结果：



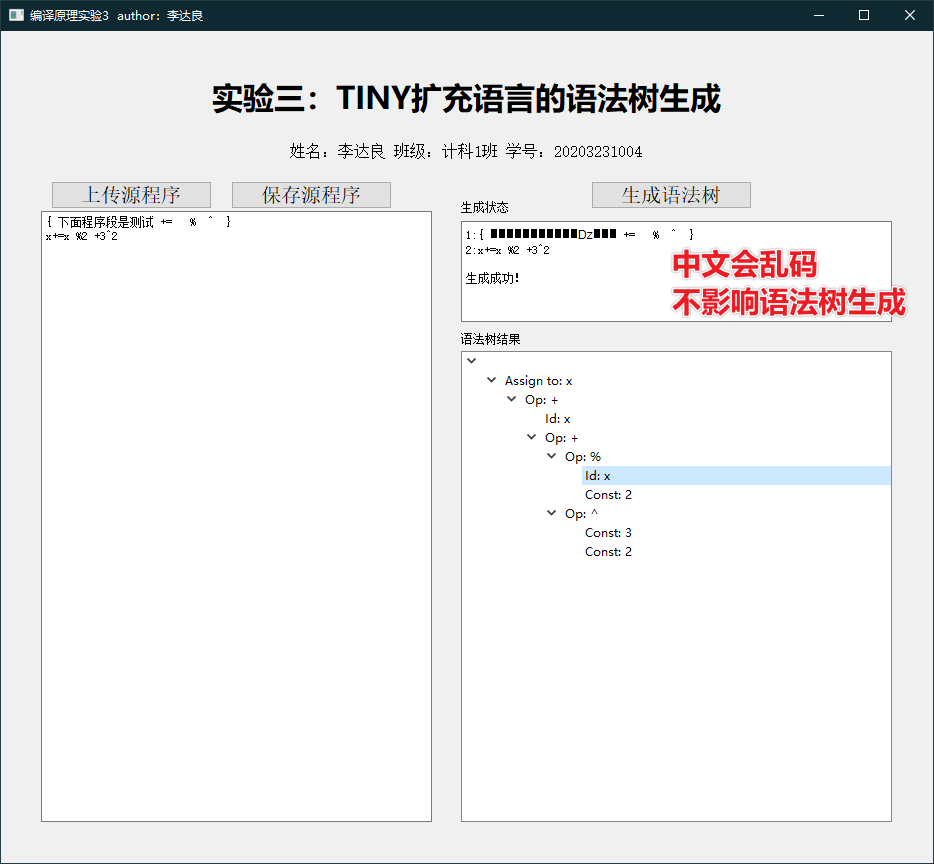
结论：符合预期结果，生成正确

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

预期结果：



程序执行结果：



结论：符合预期结果，生成正确

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

说明：if语句需要加中括号：

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

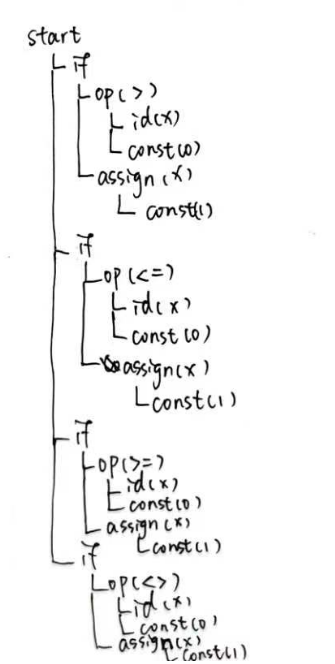
if (x>0) [x:=1];

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

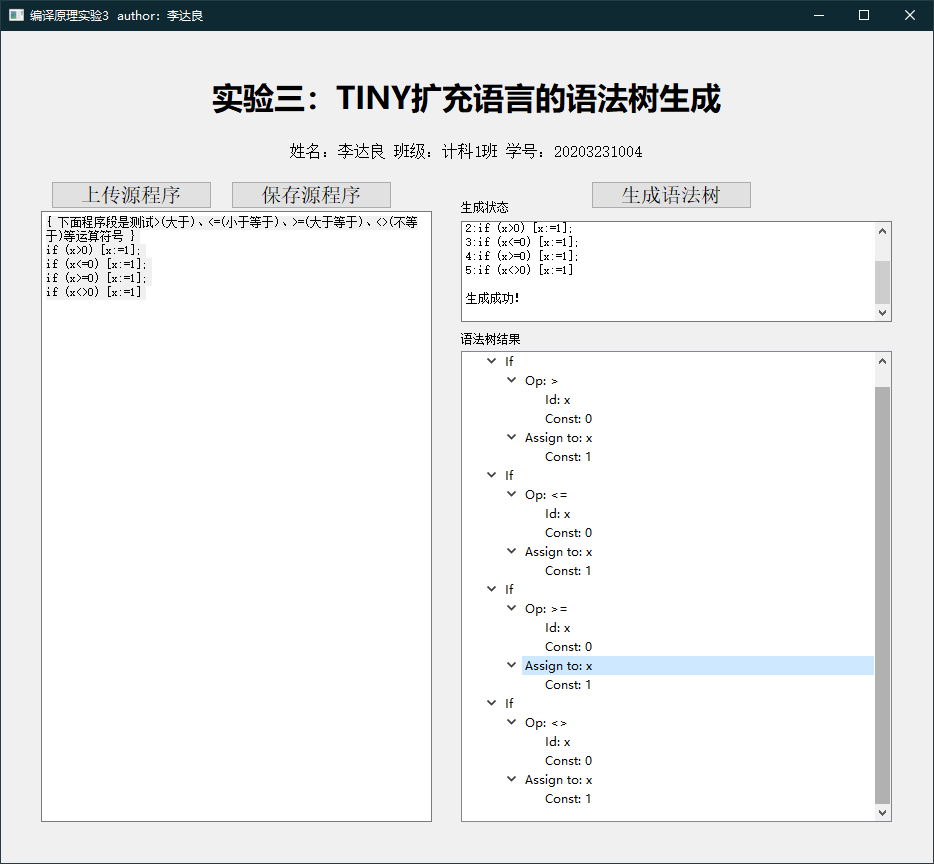
if (x>=0) [x:=1];

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

预期结果：



程序执行结果：



结论：符合预期结果，生成正确

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

说明：由于本实验定义的文法规则：regex-stmt->identifier ::= regex\_exp

regex\_exp-> regex\_exp rorop andreg | andreg

rorop -> |

andreg -> andreg randop topreg | topreg

randop -> &

topreg -> topreg topop | reg\_factor

topop -> # | ?

reg\_factor -> (regex\_exp) | ideifier | number

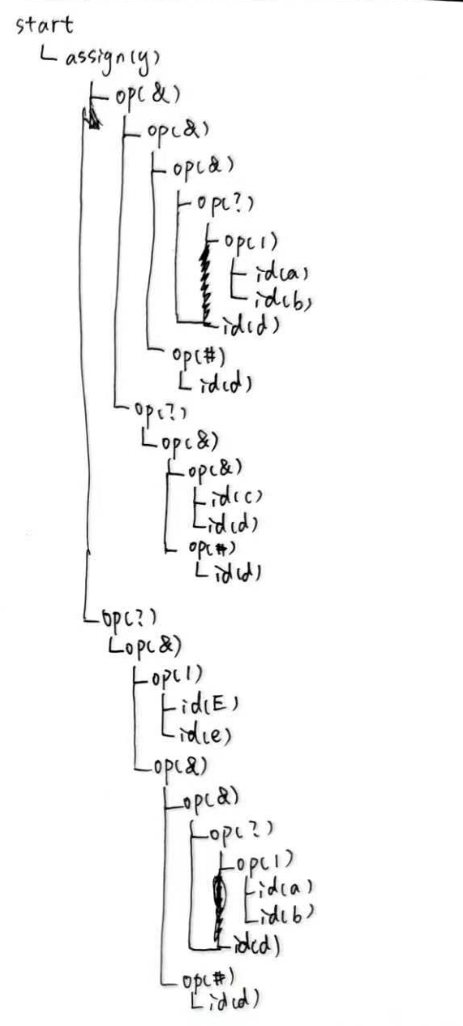
reg\_factor不含特殊符号，同时在tiny解释器中，获取token的时候没有特殊符号的enum，所以样例程序的+-.均无法识别，我们替换成abc（不影响语法树的生成和生成后的结构）。

同时，文法规则定义的正则表达式是::=开头，因此，我们样例程序改编如下：

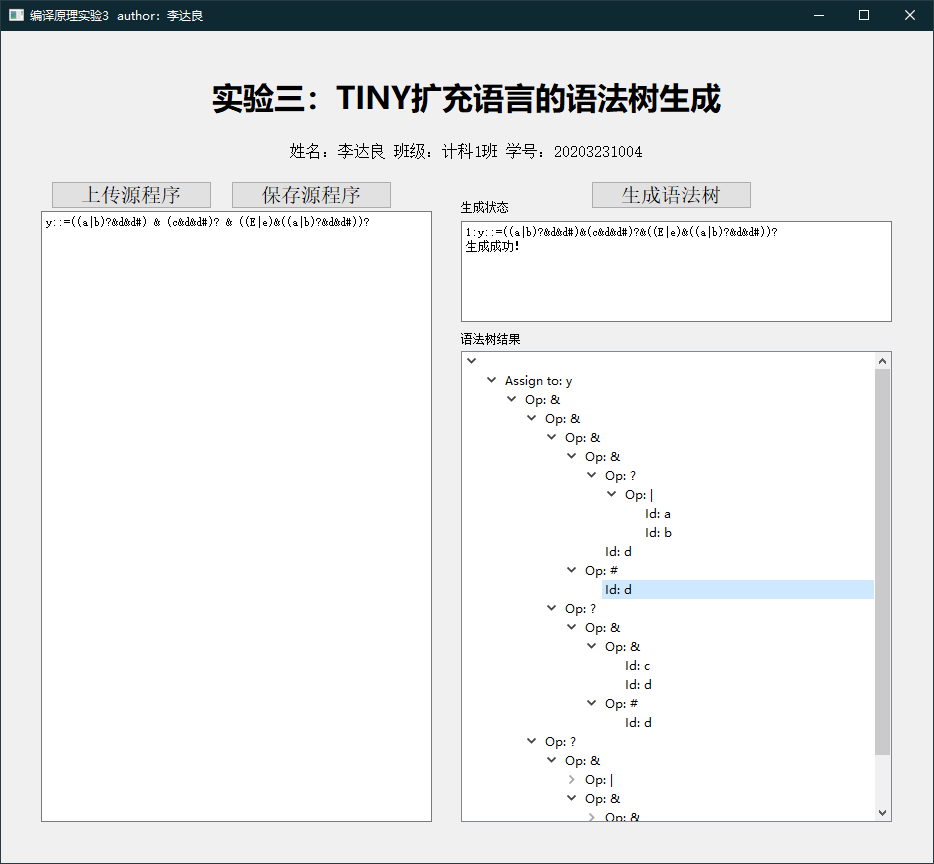
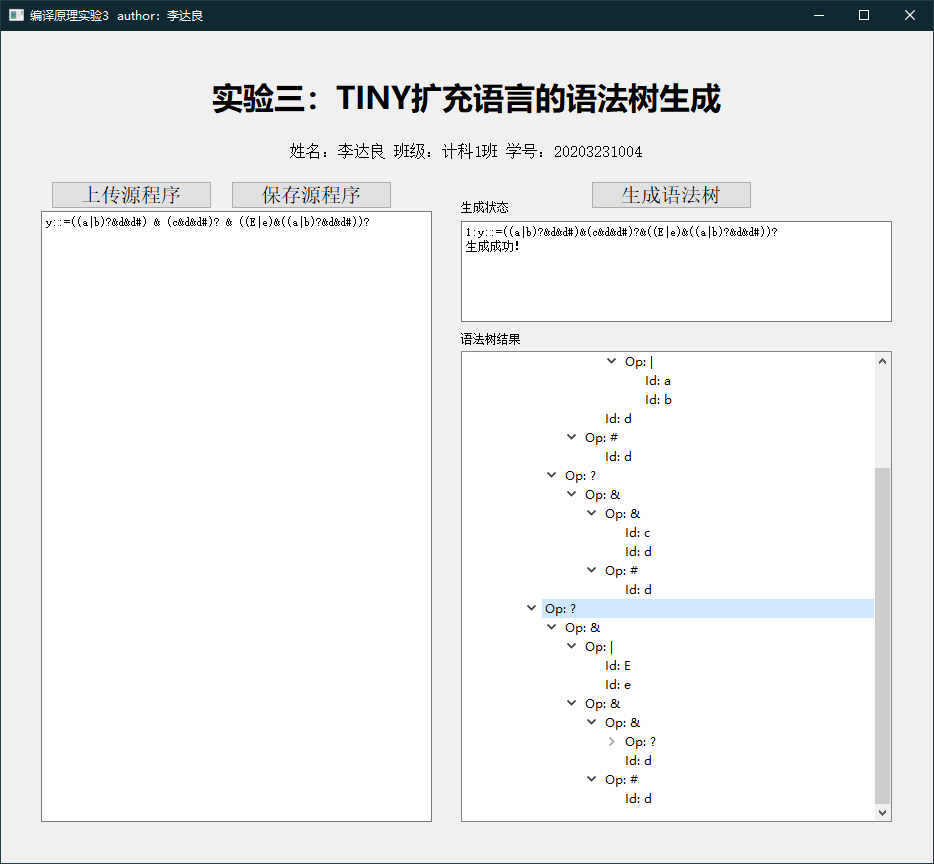
{ 下面程序段是测试正则表达式，其运算符号有： 或(|) 、连接(&)、闭包(#)、括号( ) 、可选运算符号（?）和基本正则表达式。 }

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

预期结果：



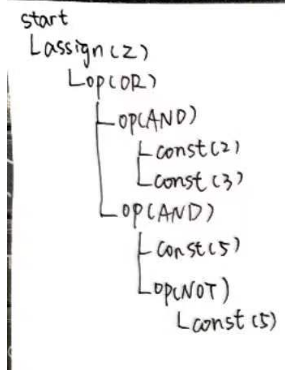
程序执行结果：

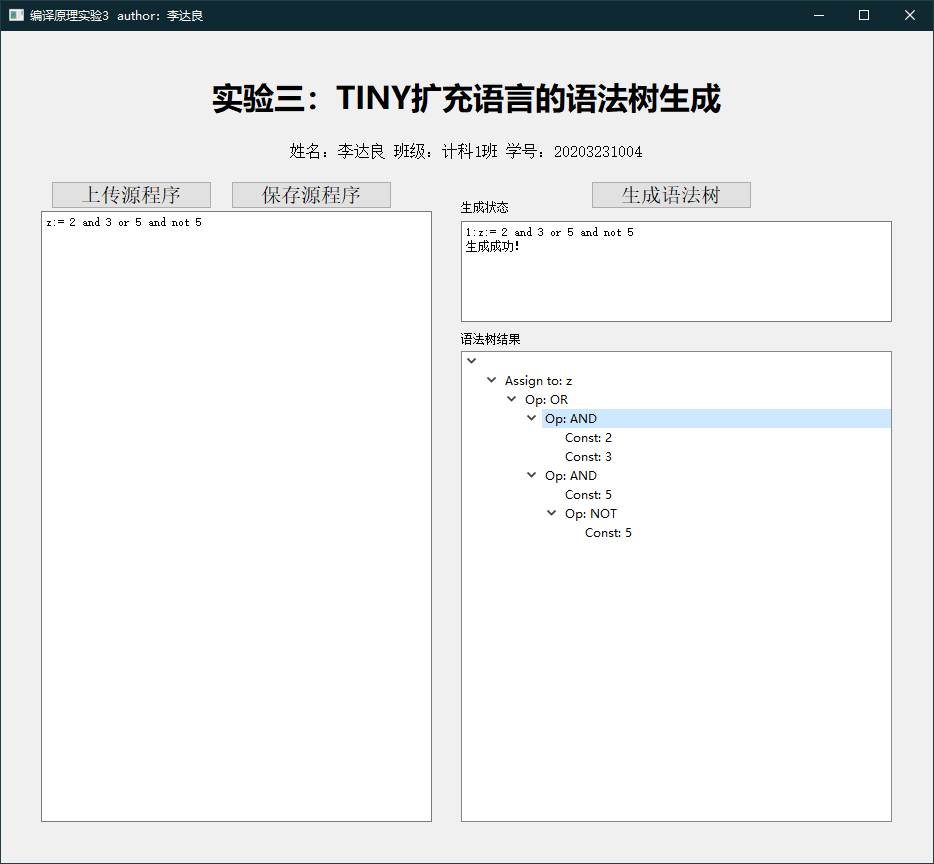
结论：符合预期结果，生成正确

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：符合预期结果，生成正确

功能7：混合功能测试

根据本实验文法规则，样例修改如下：

①if语句加括号

②+-.替换成abc

③正则表达式符号为::=

修改后样例：

{下面的程序段是混合式的功能测试 }

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

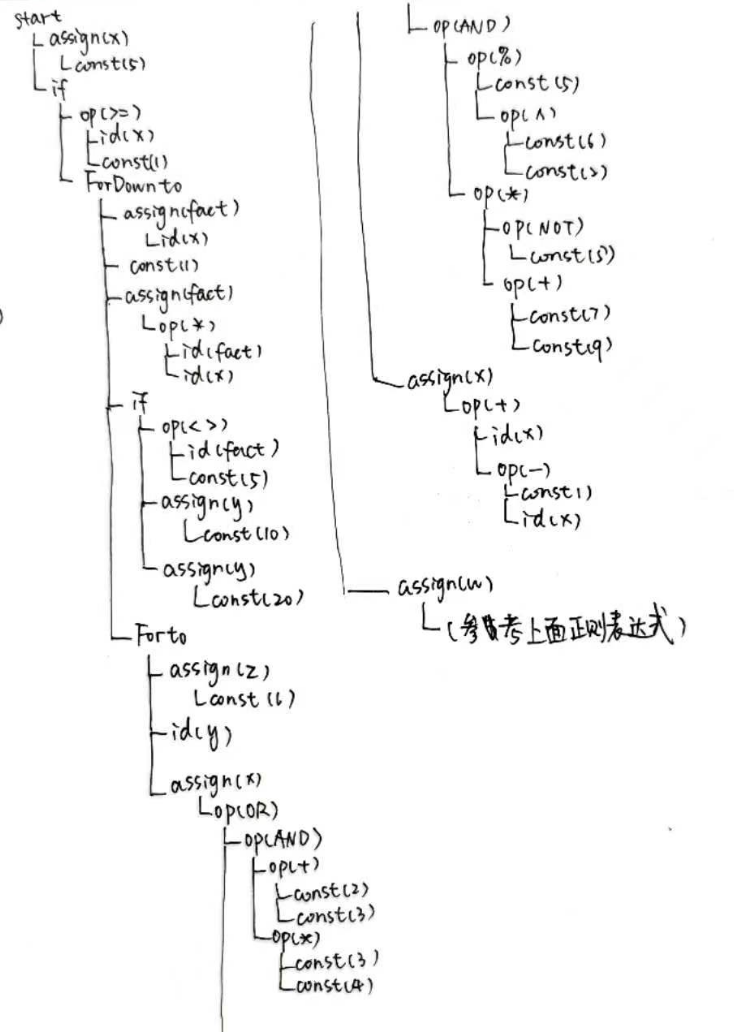
enddo

enddo]

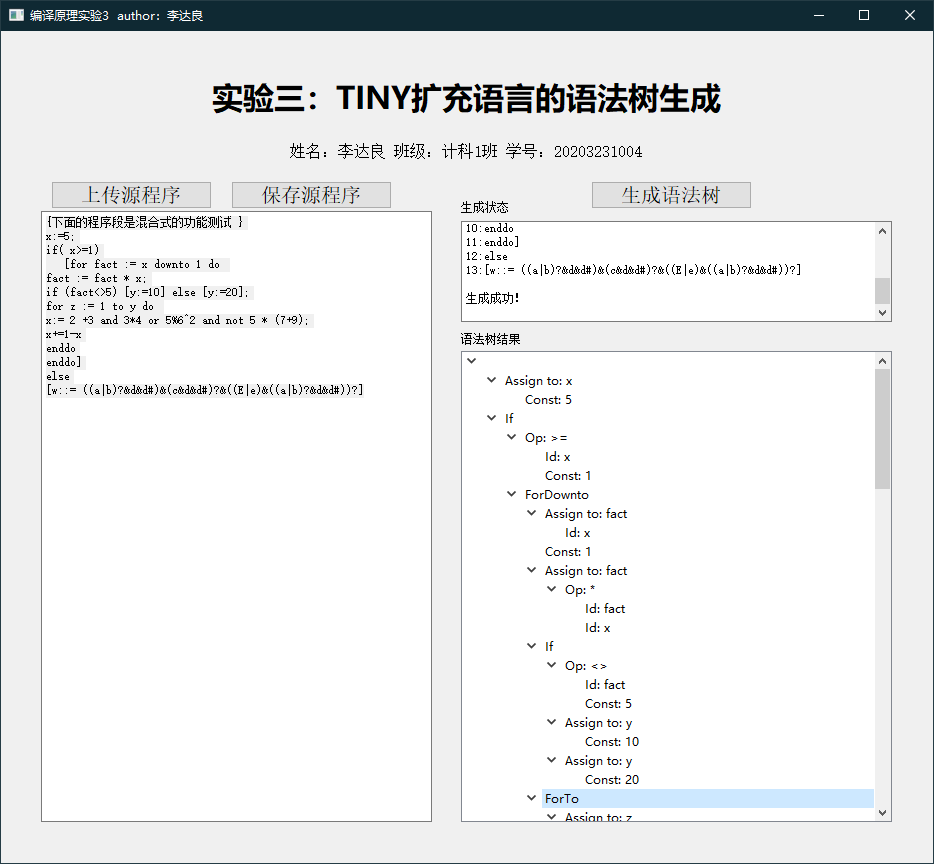
else

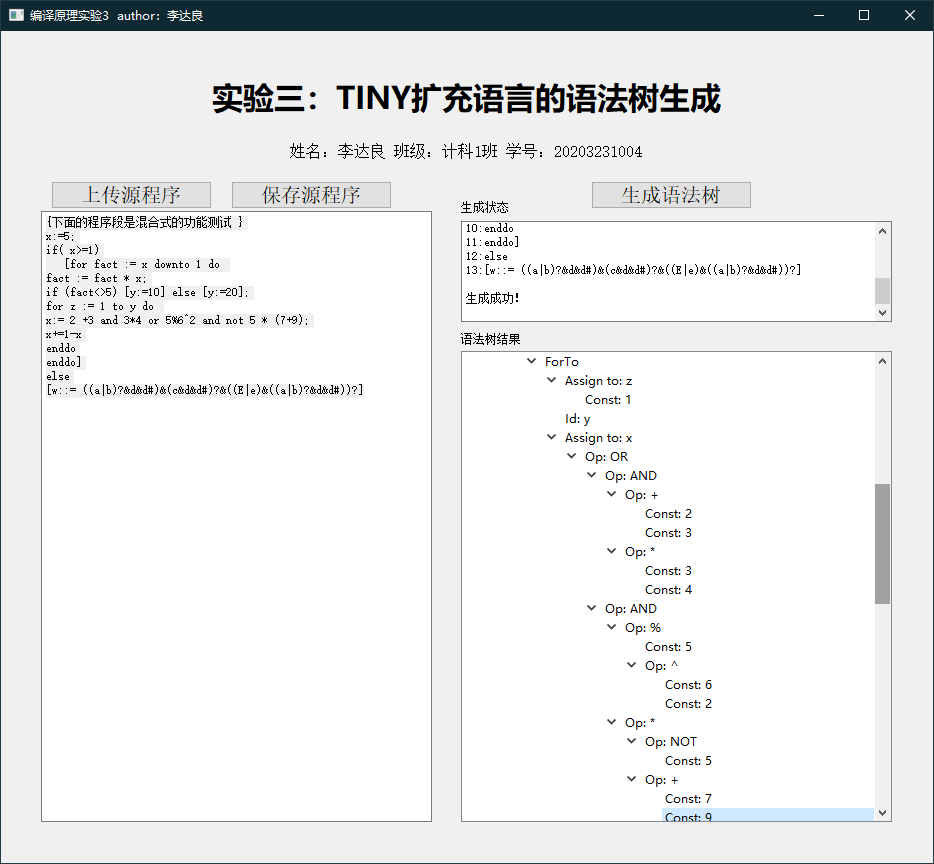
[w::= ((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|b)?&d&d#))?]

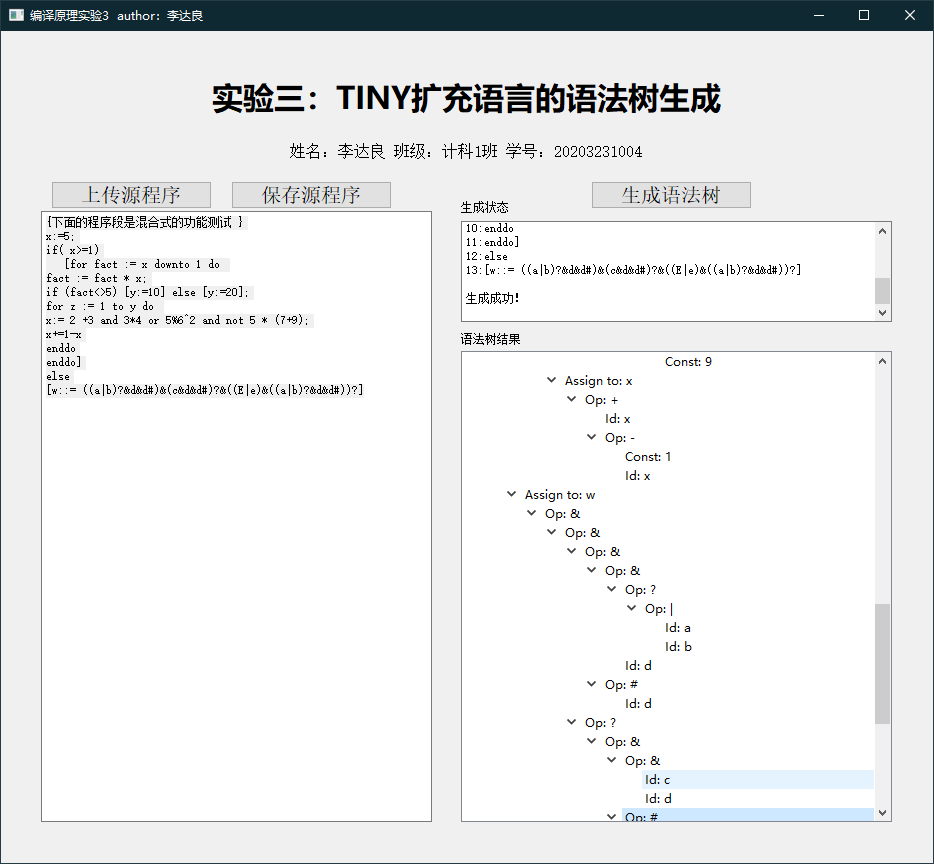
预期结果：

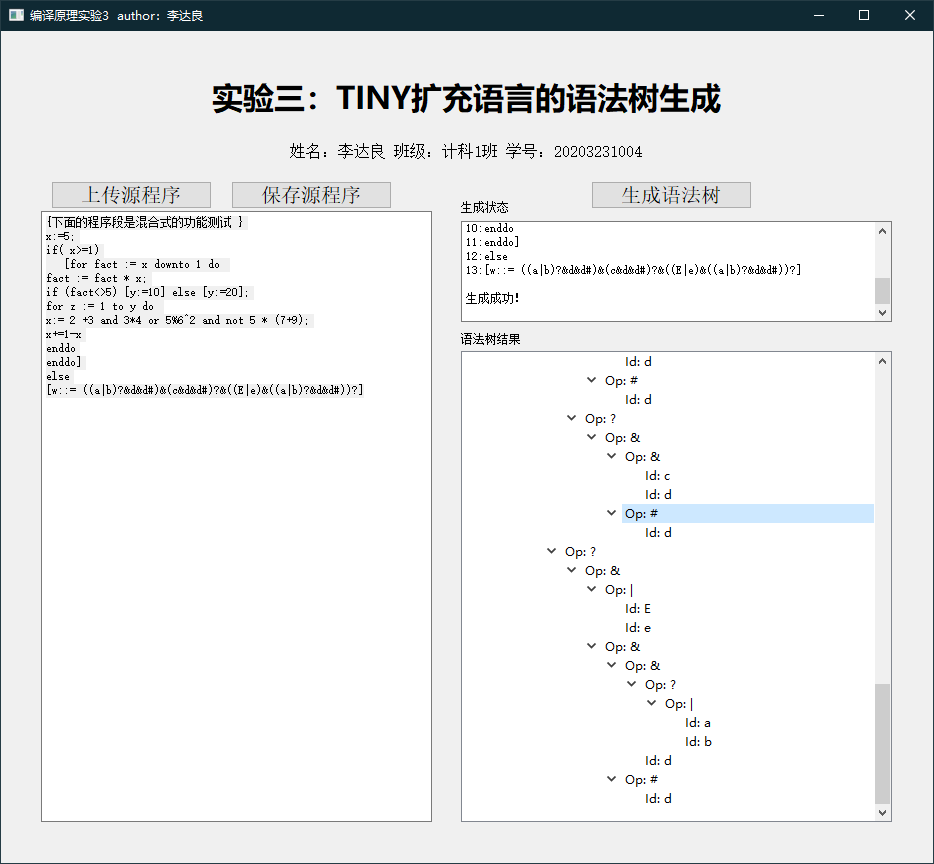


程序执行结果：









结论：符合预期结果，生成正确

最终结论：语法树生成正确，所有样例符合预期结果

三、通过测试结论对实验3的自评

自评分数：95

理由：所有样例均生成正确的语法树，但存在一些中文乱码和正则表达式中无法识别特殊符号的问题，但就本实验而言，语法树生成结构和逻辑均是正确的，因此，总体实验是成功的，5分是扣在一些细节上。

【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）】