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：2024 ～ 2025 年度第 1 学期

专　　业：网络工程

班　　级：网工 2 班

学　　生：肖翔

学　　号：20222132002

一、各功能的测试结论【本次测试只测试正确的源程序】

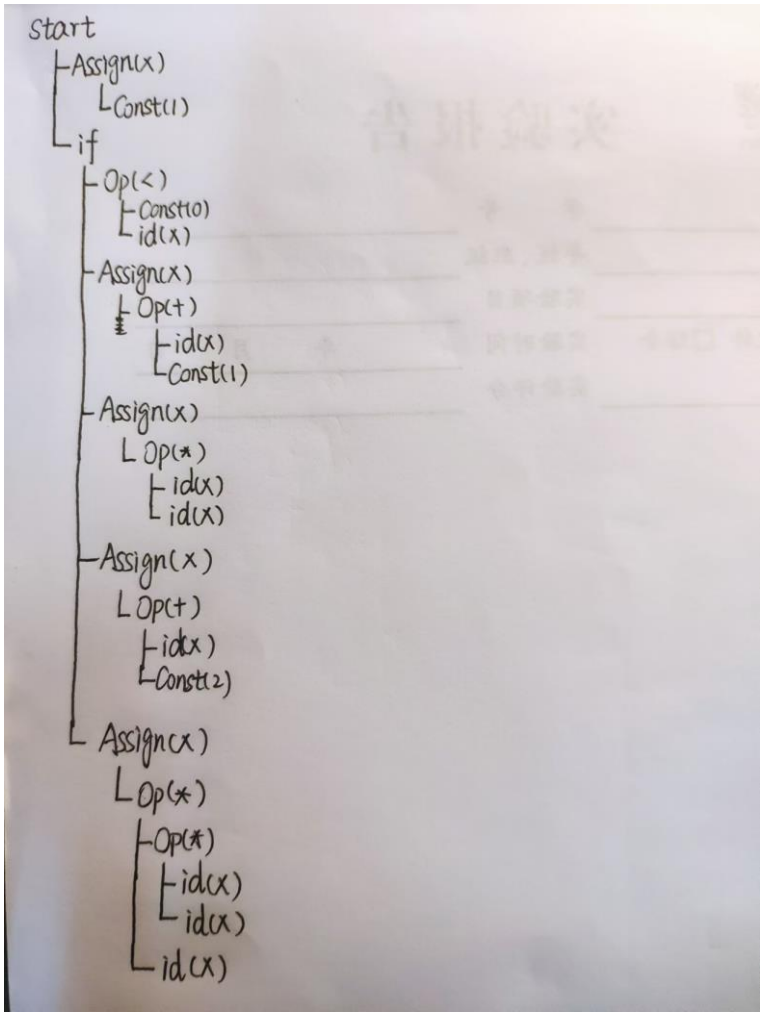
功能 1：改写书写格式后的新 if 语句

说明：使用中括号分割语句块，划定 if 的范围和 else 的范围。因此将原测试程序改为以下书写格式。

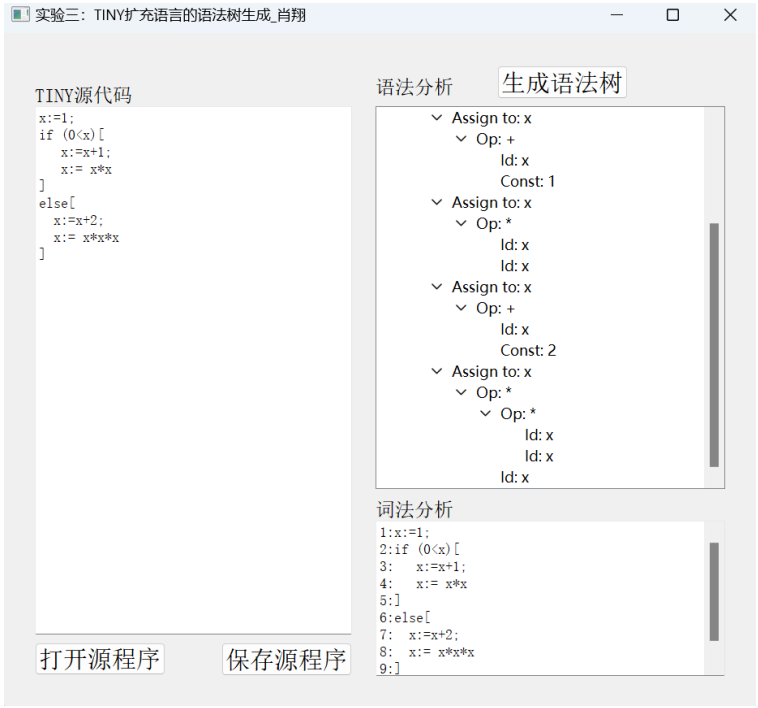
{ 下面程序段是测试修改书写格式后的 if 语句 }

```
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，程序执行结果正确。

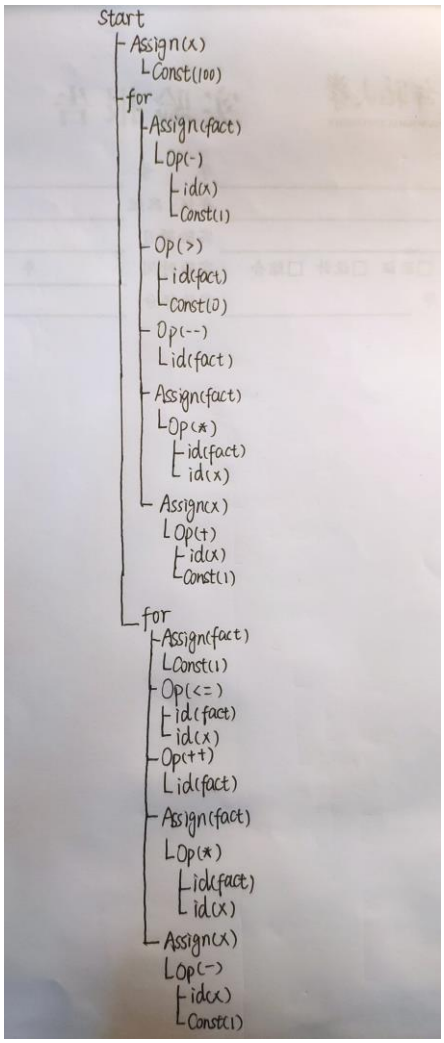
功能 2：for 循环

说明：实验三要求系统实现前置自增和前置自减即可。在本系统中，分号必须出现在语句之间。使用中括号分割语句块，划定 for 循环体的范围。因此将原测试程序改为以下书写格式。

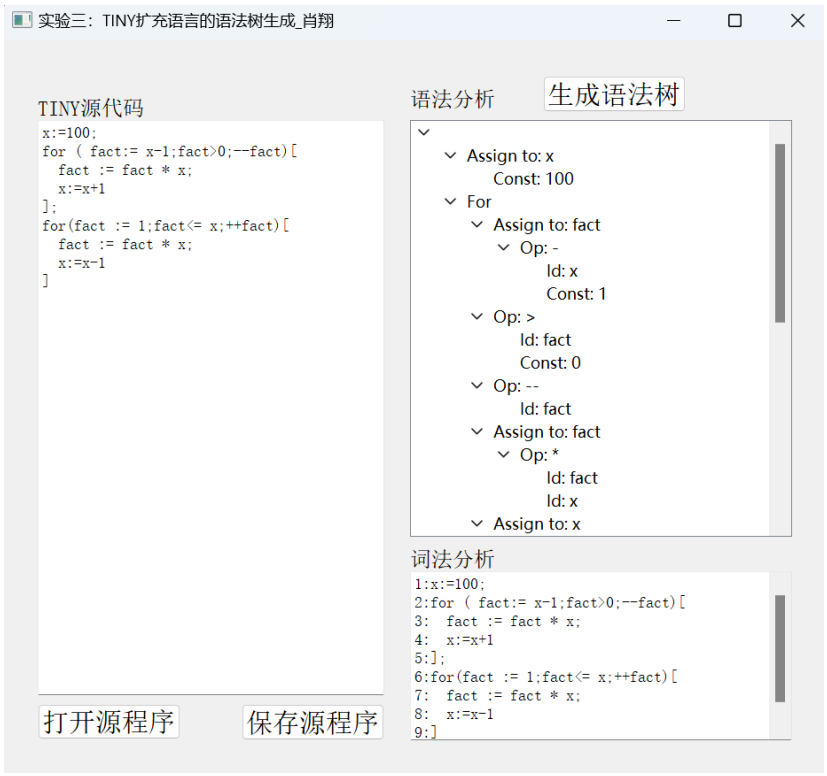
```
{ 下面程序段是测试 for 循环 }
x:=100;
for ( fact:= x-1;fact>0;--fact)[
        fact := fact * x;
```

```
        x:=x+1
];
for(fact := 1;fact <=x;++fact)[
        fact := fact * x;
        x:=x-1
]
```

预期结果：



程序执行结果：



结论：符合预期结果，程序执行结果正确。

功能 3：while 循环

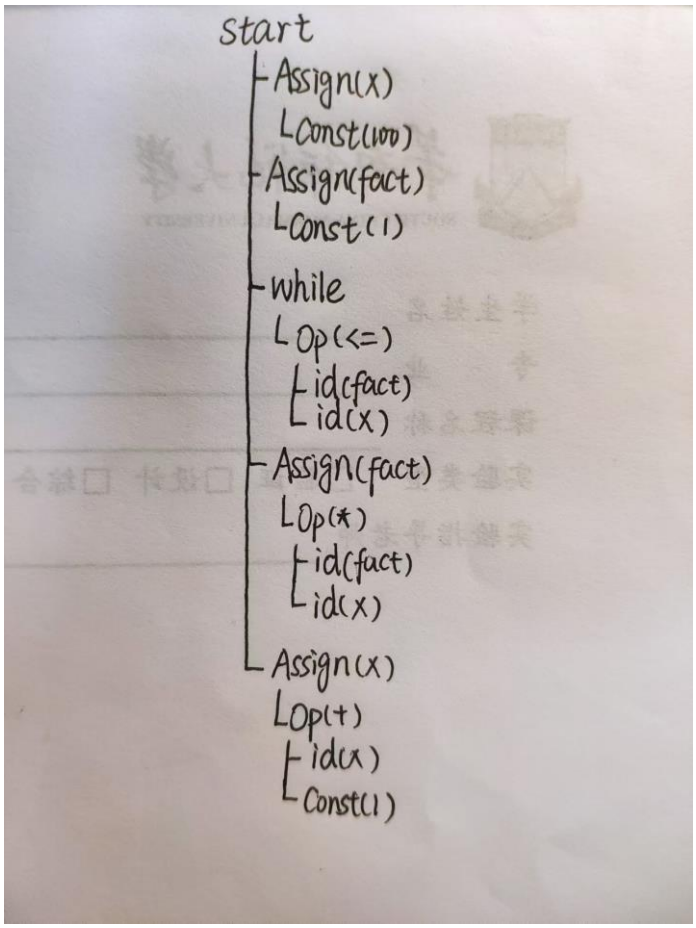
{ 下面程序段是测试 while 循环 }

x:=100;

fact:=1;

```
while(fact<=x)
        fact := fact * x;
        x:=x+1
endwhile
```

预期结果：



程序执行结果：

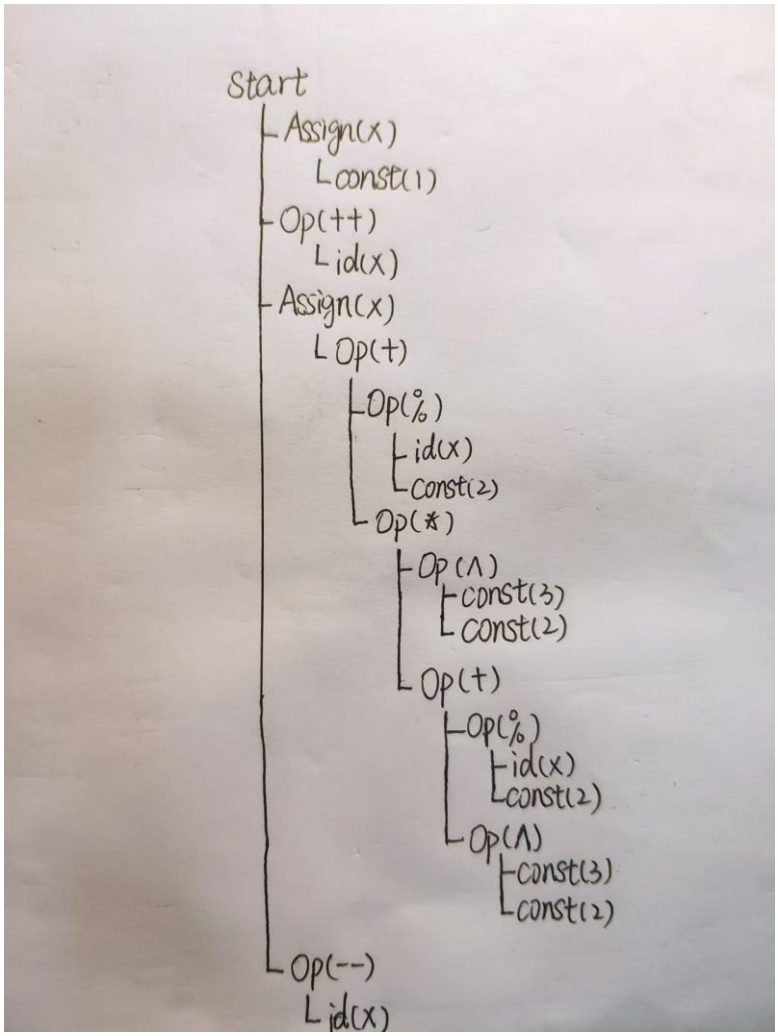
结论：符合预期结果，程序执行结果正确。

功能 4：扩充算术表达式的运算符号：++ 、 -- 、求余%、乘方^

说明：实验三要求系统实现前置自增和前置自减即可。在本系统中，分号只允许出现在语句之间，无需在最后一个语句末尾添加分号。原测试程序语法错误，赋值语句多写了一个等号。因此将原测试程序改为以下书写格式。

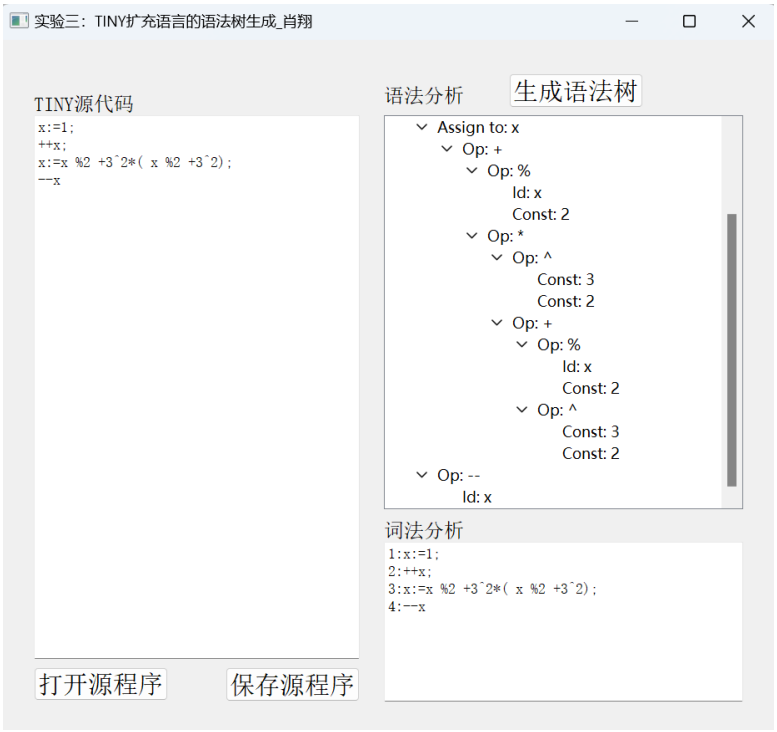
```
{ 下面程序段是测试 ++    --    %   ^   }
x:=1;
++x;
x:=x %2 +3^2*( x %2 +3^2);
x--
```

预期结果：


Start
Assign(x)
const(1)
Op(++)
id(x)
Assign(x)
Op(+)
Op(%)
id(x)
const(2)
Op(*)
Op(^)
const(3)
const(2)
Op(+)
Op(%)
id(x)
const(2)
Op(^)
const(3)
const(2)
Op(--)
id(x)

程序执行结果：


实验三：TINY扩充语言的语法树生成_肖翔
TINY源代码
x:=1;
++x;
x:=x %2 +3^2*(x %2 +3^2);
--x
打开源程序
保存源程序
语法分析
生成语法树
Assign to: x
Const: 1
Op: ++
Id: x
Assign to: x
Op: +
Op: %
Id: x
Const: 2
Op: *
Op: ^
Const: 3
Const: 2
Op: +
Op: %
Id: x
Const: 2
词法分析
1:x:=1;
2:++x;
3:x:=x %2 +3^2*(x %2 +3^2);
4:--x



结论：符合预期结果，程序执行结果正确。

功能 5：比较运算符号的扩充：>(大于)、<=(小于等于)、>=(大于等于)、<>(不等于)等运算符号
说明：在本系统中，分号只允许出现在语句之间，无需在最后一个语句末尾添加分号。因此将原测试程序改为以下书写格式。

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

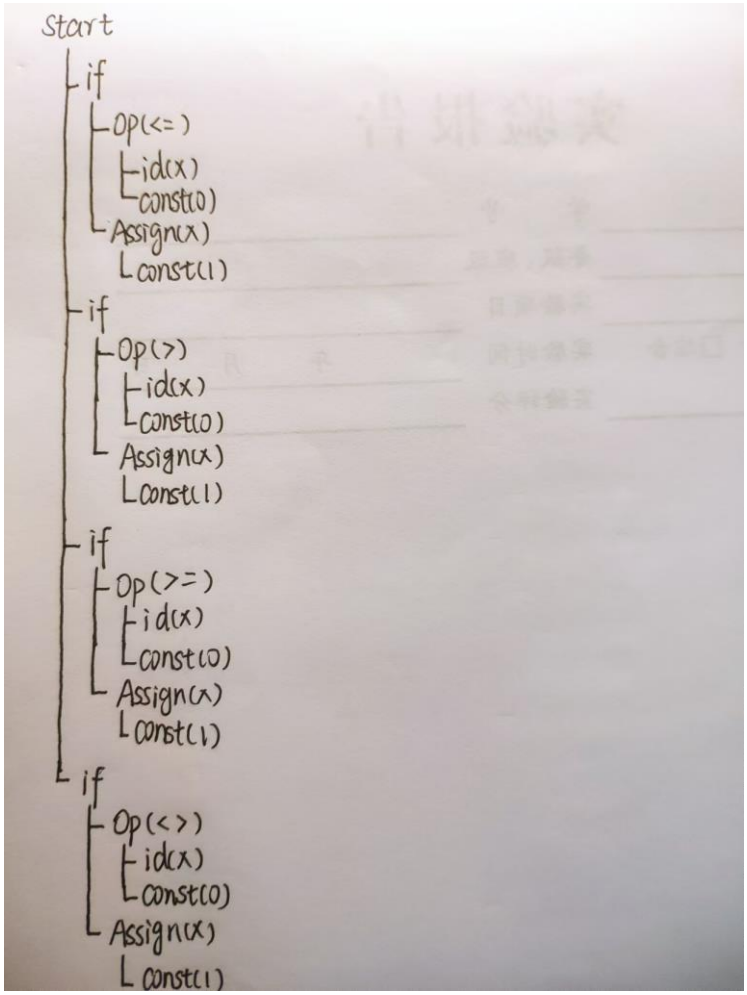
if (x<=0) x:=1;

if (x>0) x:=1;

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

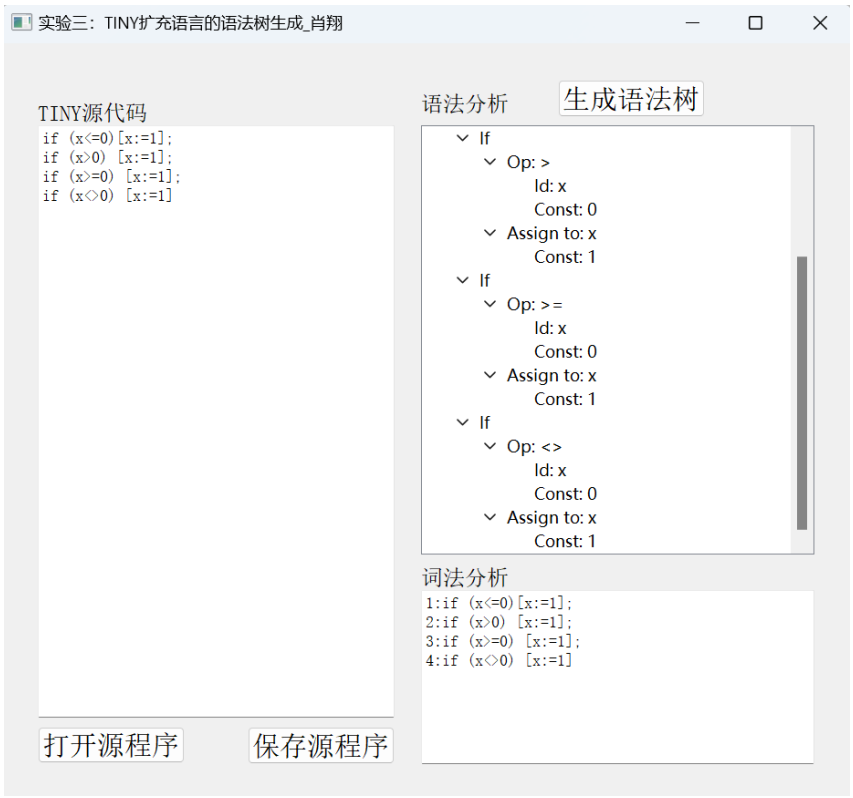
if (x<>0) x:=1

预期结果：


Start
if
Op(<=)
id(x)
const(0)
Assign(x)
const(1)
if
Op(>)
id(x)
const(0)
Assign(x)
const(1)
if
Op(>=)
id(x)
const(0)
Assign(x)
const(1)
if
Op(<>)
id(x)
const(0)
Assign(x)
const(1)

程序执行结果：


实验三：TINY扩充语言的语法树生成_肖翔
TINY源代码
if (x<=0)[x:=1];
if (x>0) [x:=1];
if (x>=0) [x:=1];
if (x◇0) [x:=1]
打开源程序
保存源程序
语法分析
生成语法树
If
Op: <=
Id: x
Const: 0
Assign to: x
Const: 1
If
Op: >
Id: x
Const: 0
Assign to: x
Const: 1
If
Op: >=
Id: x
Const: 0
Assign to: x
词法分析
1:if (x<=0)[x:=1];
2:if (x>0) [x:=1];
3:if (x>=0) [x:=1];
4:if (x◇0) [x:=1]



结论：符合预期结果，程序执行结果正确。

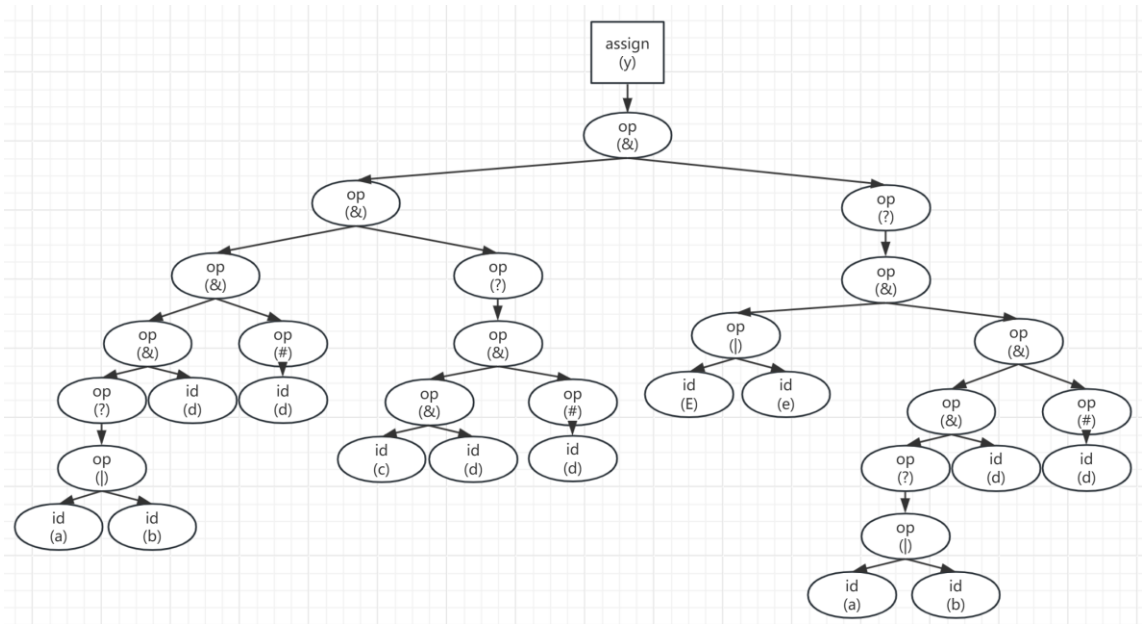
功能 6：正则表达式，其支持的运算符号有： 或(|) 、连接(&)、闭包(#)、括号( ) 、可选运算符号（?）和基本正则表达式。

说明：在设计正则文法时忽略了特殊符号，所以原测试程序中的“+”、“-”和“.”无法识别。因此分别用 a、b 和 c 代替“+”、“-”和“.”。这样做并不会改变语法树结构。在本系统中，分号只允许出现在语句之间，无需在最后一个语句末尾添加分号。因此将原测试程序改为以下书写格式。

{ 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。 }

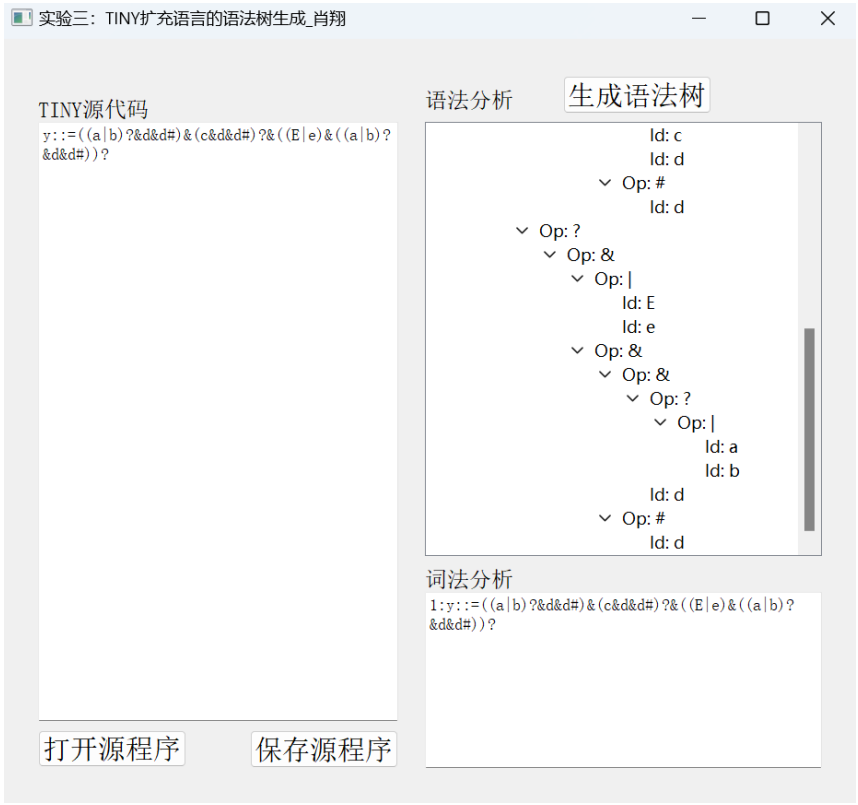
y::=((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|b)?&d&d#))?

预期结果：


assign
(y)
op
(&)
op
(&)
op
(?)
op
(&)
op
(&)
op
(?)
op
(&)
op
(&)
op
(#)
op
(&)
op
(|)
op
(&)
op
(?)
id
(d)
id
(d)
op
(&)
op
(#)
id
(E)
id
(e)
op
(&)
op
(#)
op
(|)
id
(c)
id
(d)
id
(d)
op
(?)
id
(d)
id
(d)
op
(|)
id
(a)
id
(b)
id
(a)
id
(b)

程序执行结果：


实验三：TINY扩充语言的语法树生成_肖翔
TINY源代码
y::=((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|b)?&d&d#))?
语法分析
生成语法树
Assign to: y
Op: &
Op: &
Op: &
Op: &
Op: ?
Op: |
Id: a
Id: b
Id: d
Op: #
Id: d
Op: ?
Op: &
Op: &
Id: c
Id: d
词法分析
1:y::=((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|b)?&d&d#))?
打开源程序
保存源程序



结论：系统虽然未考虑正则表达式中的特殊符号，但是识别正则表达式的功能整体正确。

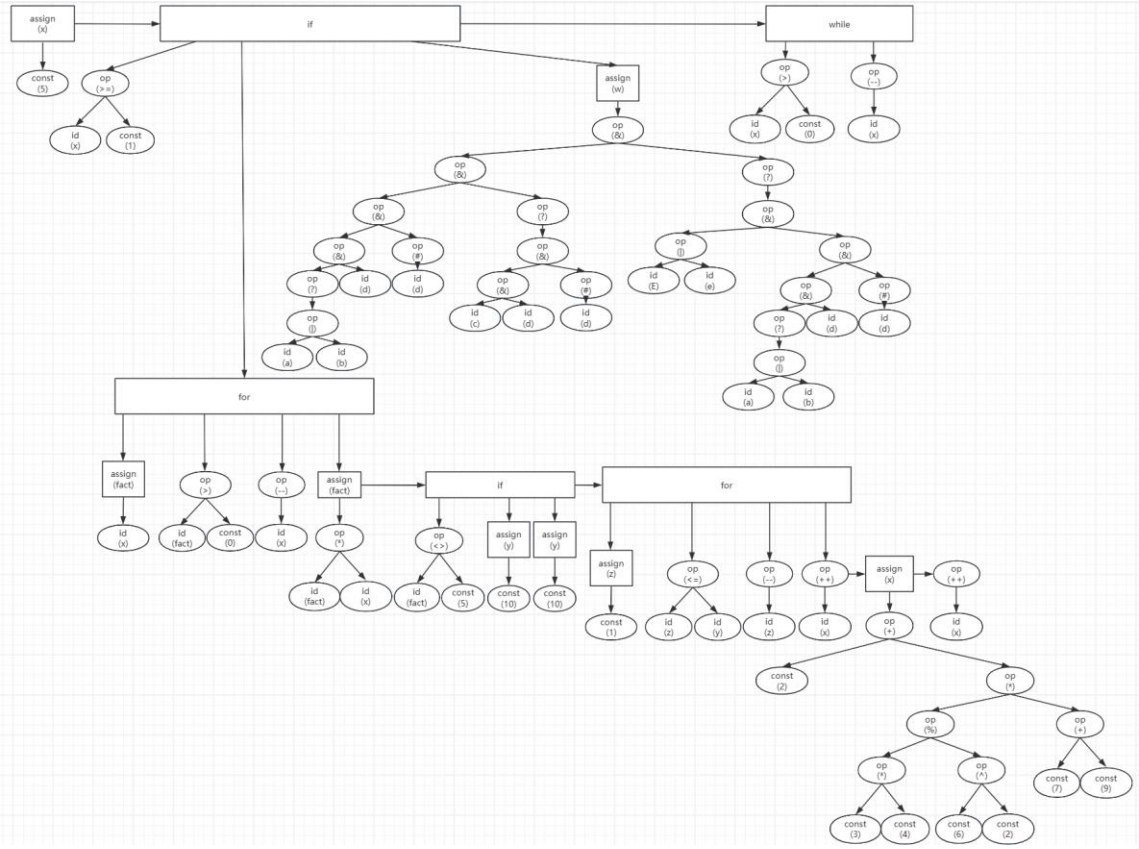
功能 7：混合功能测试

说明：实验三要求系统实现前置自增和前置自减即可。在本系统中，分号只允许出现在语句之间，无需在最后一个语句末尾添加分号，且分号必须出现在语句之间。原测试程序语法错误，赋值语句多写了一个等号。使用中括号分割语句块，划定 for 循环体的范围、if 范围和 else 范围。在设计正则文法时忽略了特殊符号，所以原测试程序中的“+”、“-”和“.”无法识别。因此分别用 a、b 和 c 代替“+”、“-”和“.”。这样做并不会改变语法树结构。因此将原测试程序改为以下书写格式。

```
{下面的程序段是混合式的功能测试 }
x:=5;
if( x>=1)[
    for( fact := x;fact>0;--fact)[
      fact := fact * x;
      if (fact<>5) [y:=10] else [y:=20]
      for (z := 1;z<=y;--z)[
          ++x;
          x:= 2 +3*4 %6^2*(7+9);
          x--
      ]
   ]
]
else
```

```
w::= ((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|b)?&d&d#))?;
while (x>0)
    --x;
endwhile
```

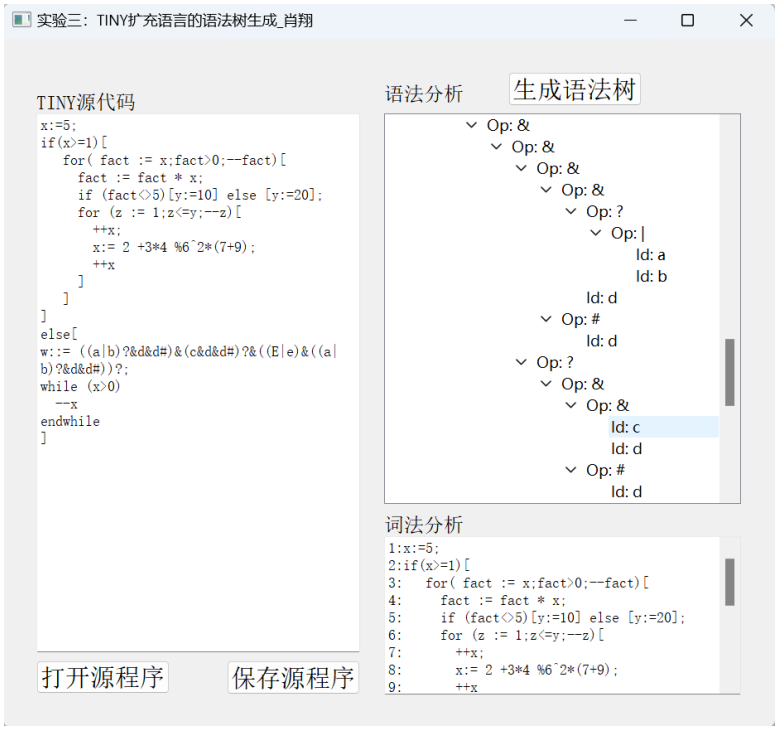
预期结果：

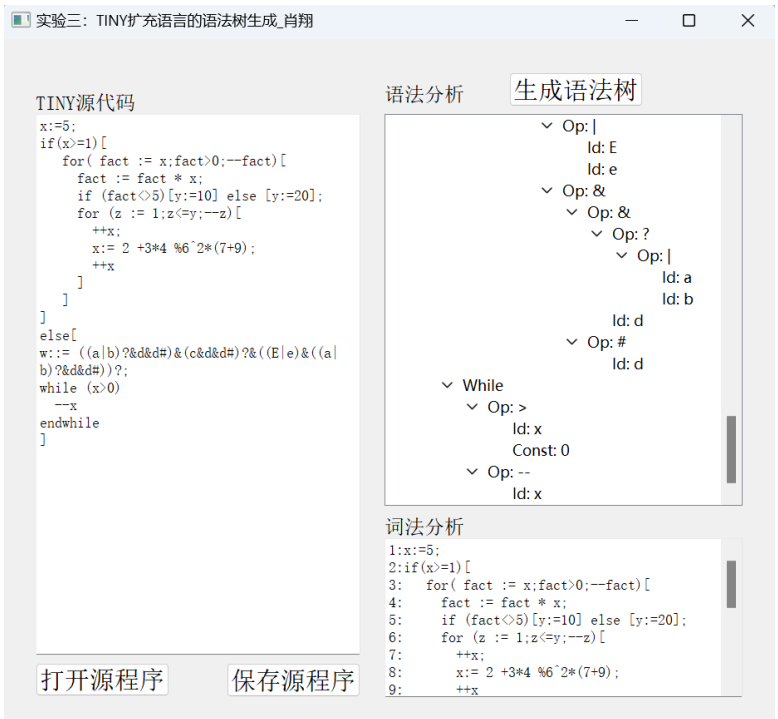


程序执行结果：


实验三：TINY扩充语言的语法树生成_肖翔
TINY源代码
x:=5;
if(x>=1)[
for(fact := x;fact>0;--fact)[
fact := fact * x;
if (fact<>5)[y:=10] else [y:=20];
for (z := 1;z<=y;--z)[
++x;
x:= 2 +3*4 %6^2*(7+9);
++x
]
]
]
else[
w:= ((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|
b)?&d&d#))?;
while (x>0)
--x
endwhile
]
打开源程序
保存源程序
语法分析
生成语法树
Id: x
If
Op: <>
Id: fact
Const: 5
Assign to: y
Const: 10
Assign to: y
Const: 20
For
Assign to: z
Const: 1
Op: <=
Id: z
Id: y
Op: --
Id: z
Op: ++
词法分析
1:x:=5;
2:if(x>=1)[
3: for(fact := x;fact>0;--fact)[
4: fact := fact * x;
5: if (fact<>5)[y:=10] else [y:=20];
6: for (z := 1;z<=y;--z)[
7: ++x;
8: x:= 2 +3*4 %6^2*(7+9);
9: ++x


实验三：TINY扩充语言的语法树生成_肖翔
TINY源代码
x:=5;
if(x>=1)[
for(fact := x;fact>0;--fact)[
fact := fact * x;
if (fact<>5)[y:=10] else [y:=20];
for (z := 1;z<=y;--z)[
++x;
x:= 2 +3*4 %6^2*(7+9);
++x
]
]
]
else[
w:= ((a|b)?&d&d#)&(c&d&d#)?&((E|e)&((a|
b)?&d&d#))?;
while (x>0)
--x
endwhile
]
打开源程序
保存源程序
语法分析
生成语法树
Id: x
Assign to: x
Op: +
Const: 2
Op: *
Op: %
Op: *
Const: 3
Const: 4
Op: ^
Const: 6
Const: 2
Op: +
Const: 7
Const: 9
Op: ++
Id: x
Assign to: w
词法分析
1:x:=5;
2:if(x>=1)[
3: for(fact := x;fact>0;--fact)[
4: fact := fact * x;
5: if (fact<>5)[y:=10] else [y:=20];
6: for (z := 1;z<=y;--z)[
7: ++x;
8: x:= 2 +3*4 %6^2*(7+9);
9: ++x





结论：系统虽然未考虑正则表达式中的特殊符号，但是识别正则表达式的功能整体正确。如不考虑正则表达式，程序执行结果与预期结果一致。

最终结论：系统功能预期运行，对于绝大部分程序都能够准确地生成语法树。

二、通过测试结论对实验 3 的自评

自评分数：94

原因：对于绝大部分测试用例都能准确地生成语法树。6 分扣在了设计正则表达式文法规则时未考虑特殊符号。

【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）】