1 测试一

测试以下 **Tiny** 程序：

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }
if  (0<x)  { don't compute if x <= 0 } [
  for  fact := x downto 1 do
    fact := fact * x;
  enddo
  write fact;  { output factorial of x }
]
```

测试结果如下图：

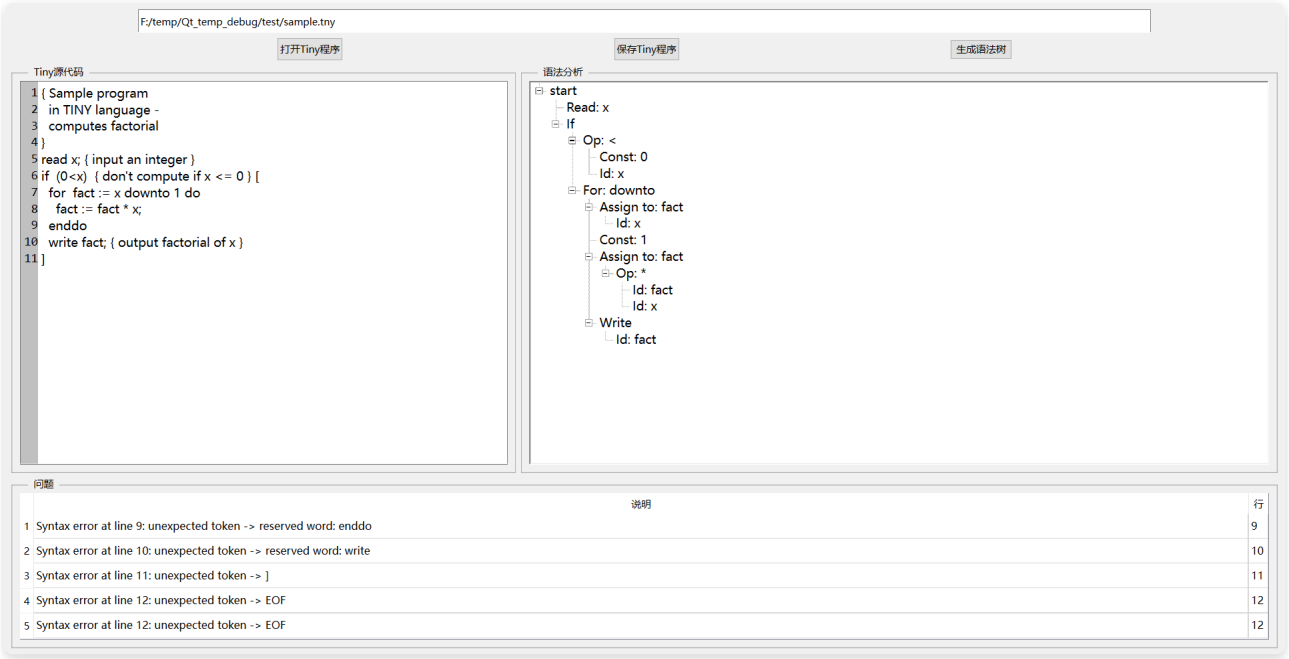

图1-1. 测试1-1

可见该程序存在语法错误，第 **8**、**10** 行多了分号，而第 **9** 行缺少分号，因此生成了错误的语法树。

将错误修改后再次测试，结果如下图：

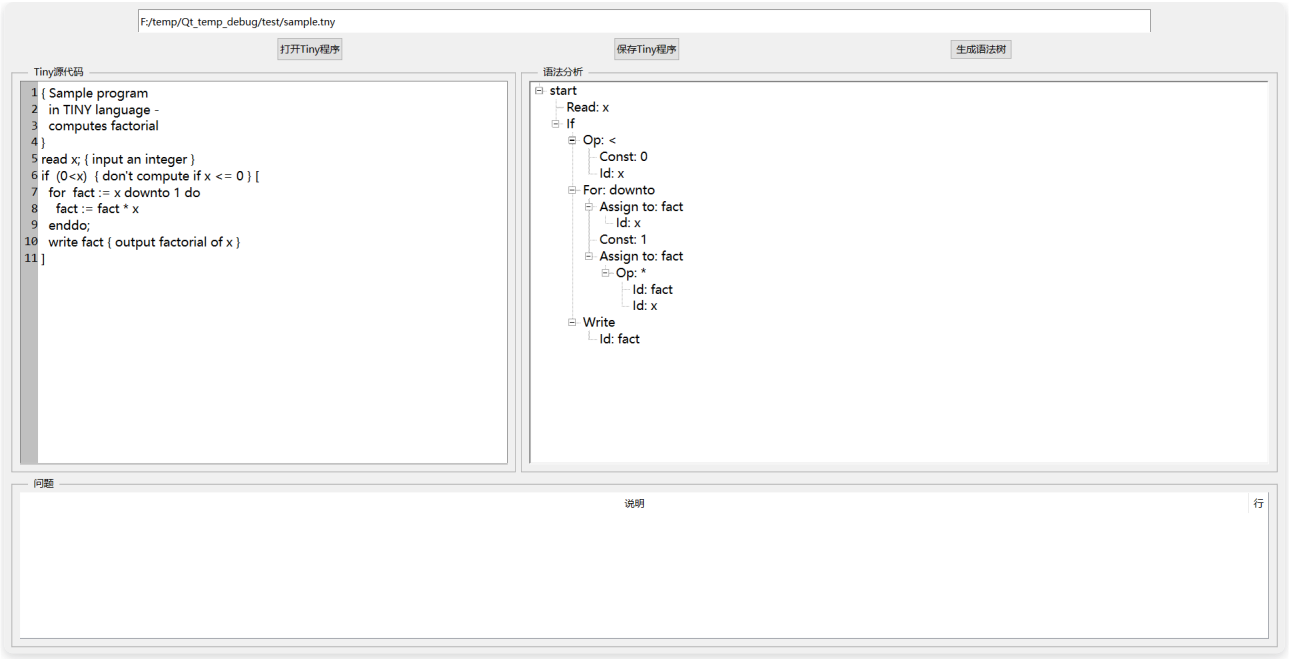

图1-2. 测试1-2

此时能生成正确的语法树。

2 测试二

测试以下 **Tiny** 文件：

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }

if  (x>0) { don't compute if x <= 0 } [
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact  { output factorial of x }
]
```

测试结果如下图：

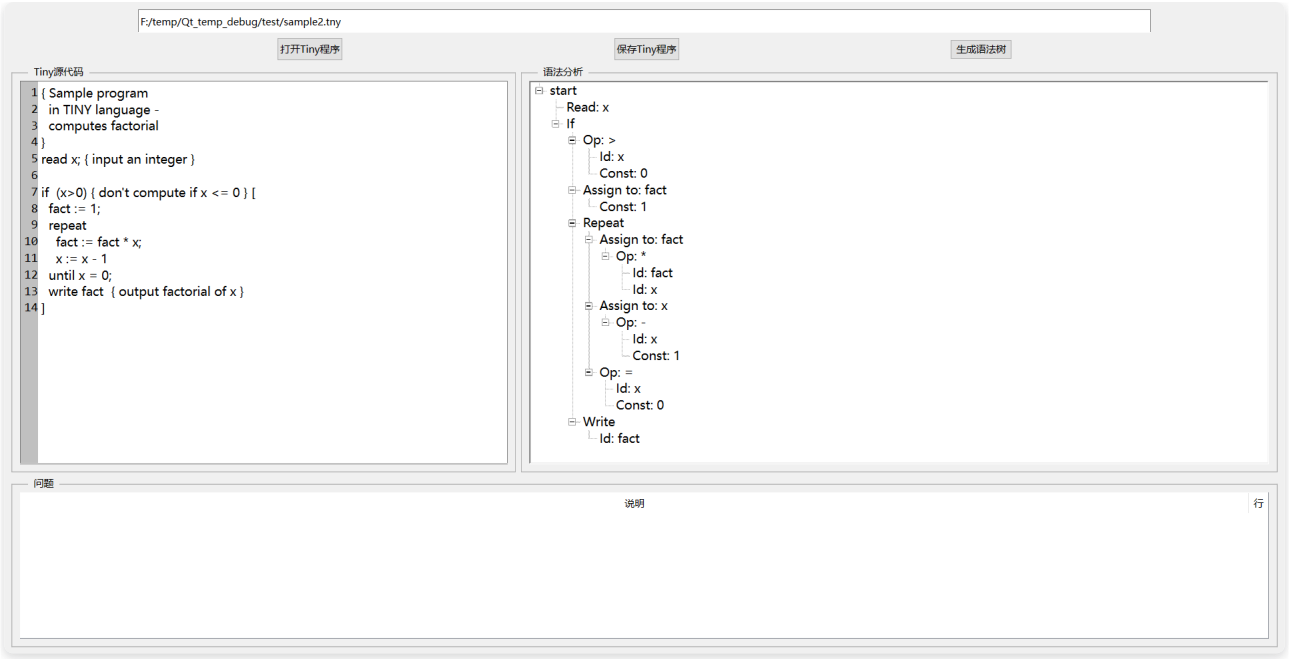

图2. 测试2

该程序语法正确，可以生成正确的语法树。

3 测试三

测试以下 **Tiny** 文件：

```
{ 计算a对于p的逆元 }
read a;
read p;
exponent := p - 2;
result := 1;

repeat
    if (exponent % 2 = 1) [result := (result * a) % p];
    a := (a * a) % p;
    exponent := exponent / 2
until exponent <= 0;

write result;

{ 计算1~n的素数 }
read n;

for i := 2 to n do
    isPrime := 1;
    for j := i-1 downto 2 do
        if (i % j = 0) [
            isPrime := 0
        ]
    enddo;
    if (isPrime = 1) [
        write i
    ] else [
```

```
        write 0 - i
    ]
enddo;

{ 测试嵌套if }
if (a = b) [
    if (a = c) [
        write 1
    ] else [
        if (a = d) [
            write 2
        ];
        write 3
    ]
] else [
    write 4
];

{ 浮点数正则表达式 }
regtwo ::= (a|b)?&d&d# & (p&d&d#)? & ((e|E)&(a|b)?&d&d#)?;

{ 测试复杂表达式 }
a += 1 + (2 and 5) * 3 - 4 / not 5 % 6 or 7^(8-1)^(9+10)
```

测试结果如下图：

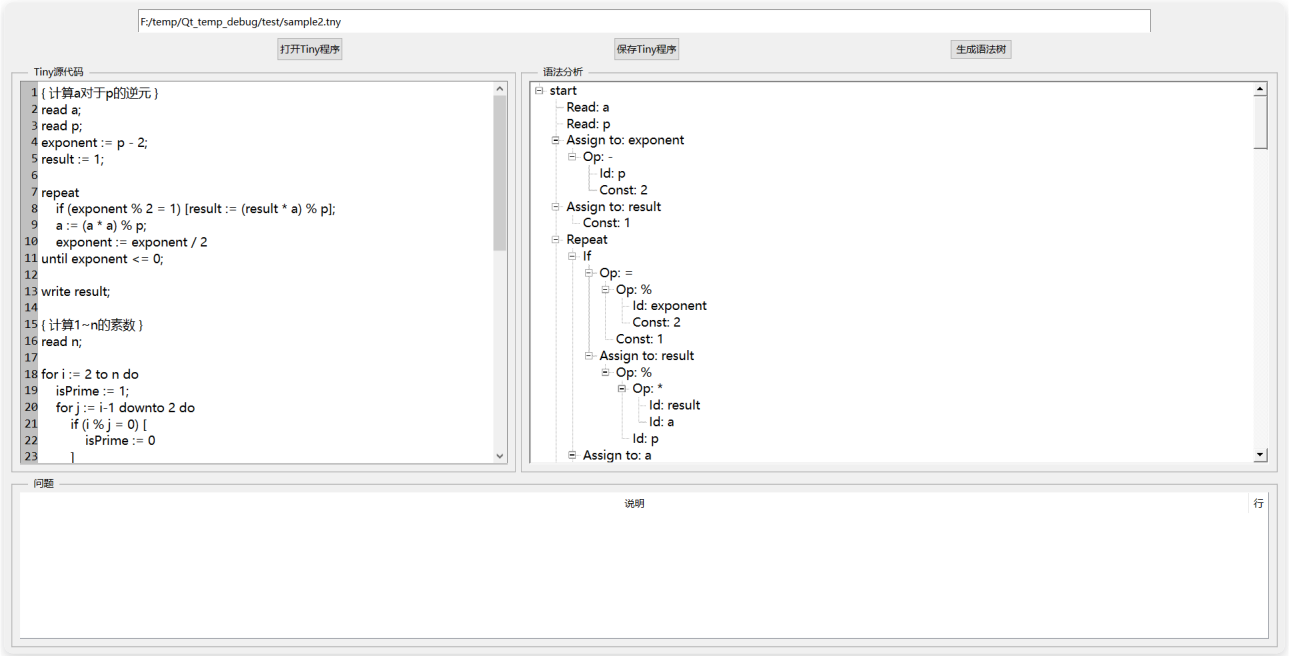

图3. 测试3

仔细检查后，确定程序可以生成正确的语法树，说明程序可以正确处理复杂的 **Tiny** 程序。

4 测试四

测试以下 **Tiny** 文件，观察程序是否能正确报错：

```
if (1 = 1) [
    x :== 1;
];

for x += 1 to 3 do
    write 1
enddo

x :== 1
```

测试结果如下图：

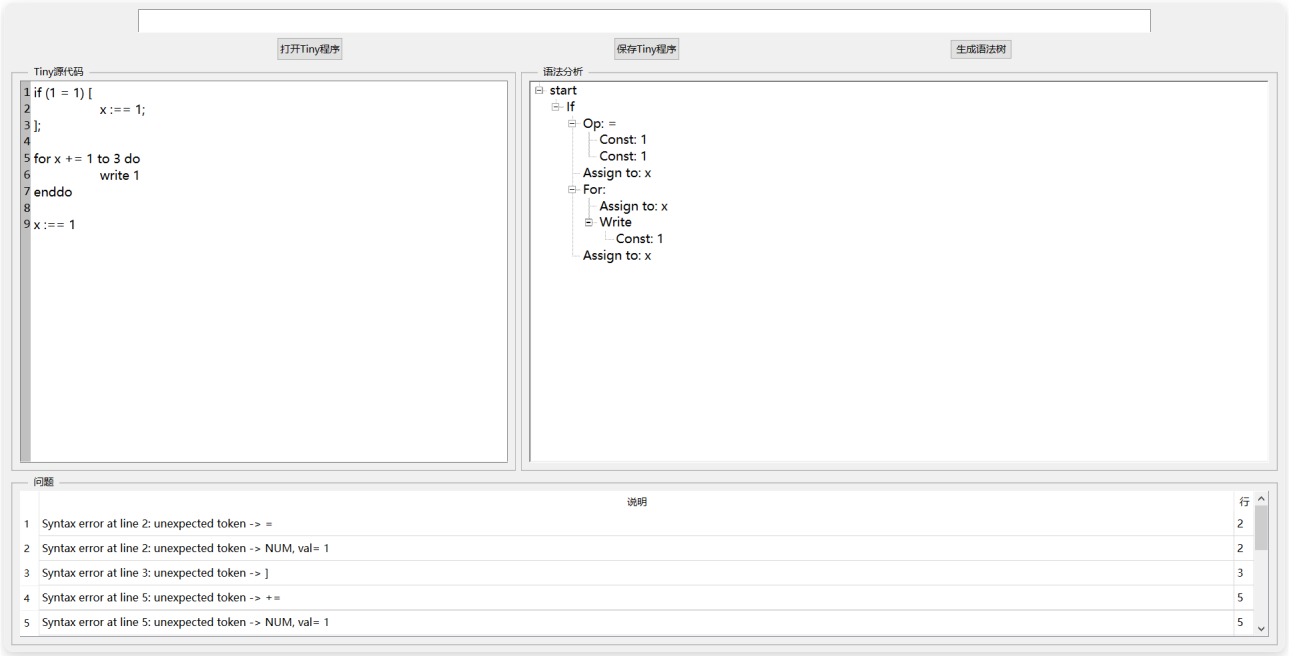

图4. 测试4

可以看到，程序能正确报错，说明程序能正确处理错误的 **Tiny** 程序。