

# 1 测试一

测试以下 Tiny 程序：

```
1 { Sample program
2   in TINY language -
3   computes factorial
4 }
5 read x; { input an integer }
6 if (0<x) { don't compute if x <= 0 } [
7   for fact := x downto 1 do
8     fact := fact * x;
9   enddo
10  write fact; { output factorial of x }
11 ]
```

测试结果如下图：

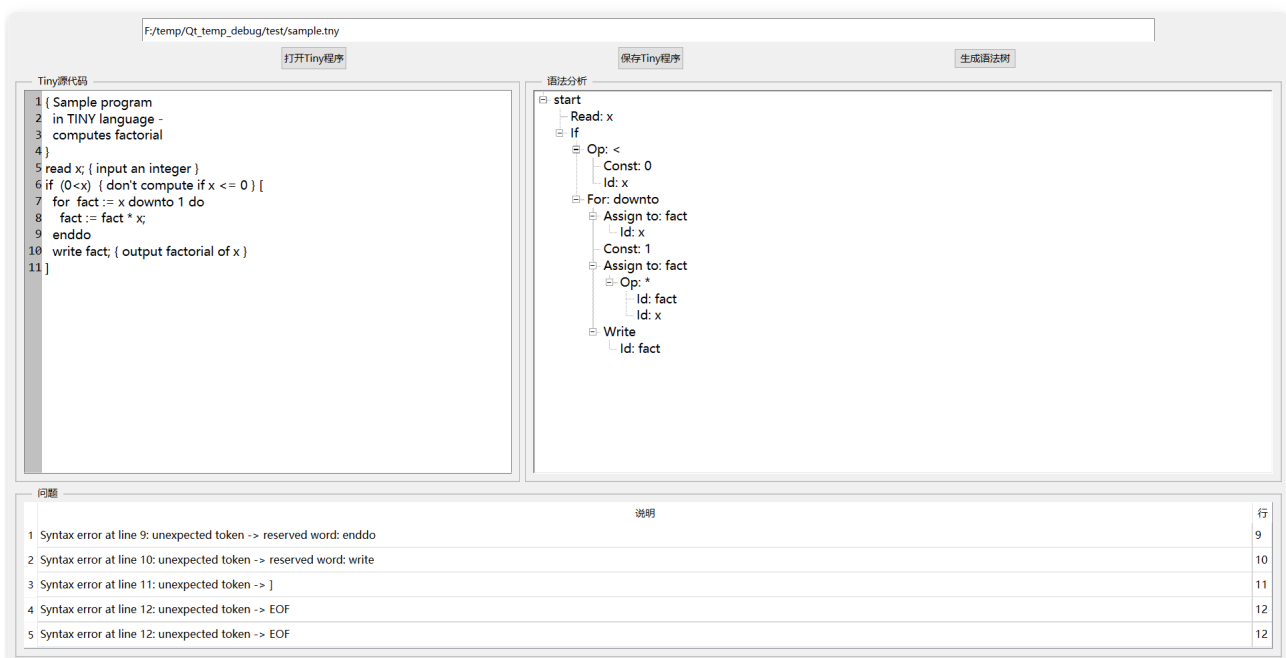


图1-1. 测试1-1

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：

```

1 { Sample program
2   in TINY language -
3   computes factorial
4 }
5 read x; { input an integer }
6
7 if (x>0) { don't compute if x <= 0 } [
8   fact := 1;
9   repeat
10    fact := fact * x;
11    x := x - 1
12  until x = 0;
13  write fact { output factorial of x }
14 ]

```

测试结果如下图：

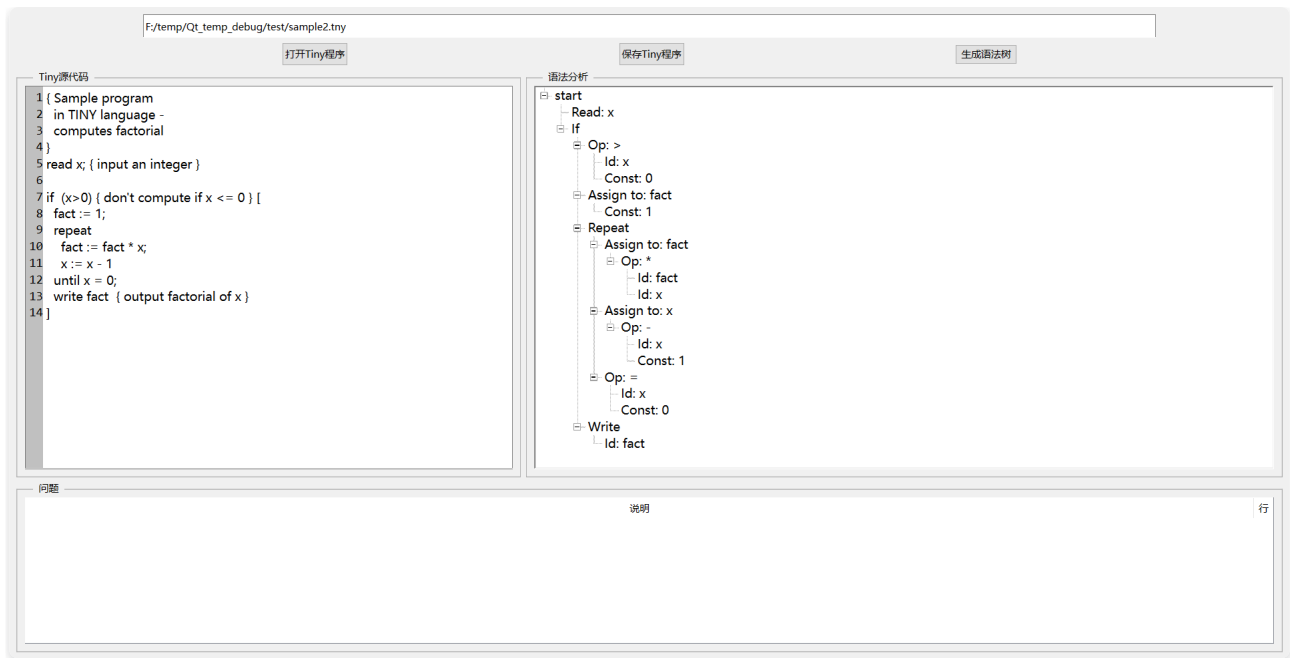


图2. 测试2

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：

```

1 { 计算a对于p的逆元 }
2 read a;
3 read p;
4 exponent := p - 2;
5 result := 1;
6
7 repeat
8     if (exponent % 2 = 1) [result := (result * a) % p];
9     a := (a * a) % p;
10    exponent := exponent / 2
11 until exponent <= 0;
12
13 write result;
14
15 { 计算1~n的素数 }
16 read n;
17
18 for i := 2 to n do
19     isPrime := 1;
20     for j := i-1 downto 2 do
21         if (i % j = 0) [
22             isPrime := 0
23         ]
24     enddo;
25     if (isPrime = 1) [
26         write i
27     ] else [

```

```

28         write 0 - i
29     ]
30 enddo;
31
32 { 测试嵌套if }
33 if (a = b) [
34     if (a = c) [
35         write 1
36     ] else [
37         if (a = d) [
38             write 2
39         ];
40         write 3
41     ]
42 ] else [
43     write 4
44 ];
45
46 { 浮点数正则表达式 }
47 regtwo ::= (a|b)?&d&d# & (p&d&d#)? & ((e|E)&(a|b)?&d&d#)?;
48
49 { 测试复杂表达式 }
50 a += 1 + (2 and 5) * 3 - 4 / not 5 % 6 or 7^(8-1)^(9+10)

```

测试结果如下图：

The screenshot displays the Tiny compiler's user interface. At the top, there's a file path 'F:/temp/Qt\_temp\_debug/test/sample2.tny' and buttons for '打开Tiny程序' (Open Tiny program), '保存Tiny程序' (Save Tiny program), and '生成语法树' (Generate syntax tree). The main area is split into two panes. The left pane, titled 'Tiny源代码' (Tiny source code), shows a list of 23 lines of code, including comments in Chinese and operations like reading variables, calculating exponents, and checking for primes. The right pane, titled '语法分析' (Syntax analysis), displays a hierarchical syntax tree for the first line of code. The tree starts with 'start', leading to 'Read: a', 'Read: p', and 'Assign to: exponent'. It then branches into 'Op: -', 'Id: p', and 'Const: 2', followed by 'Assign to: result' and 'Const: 1'. A 'Repeat' block follows, containing an 'If' statement with 'Op: =' and 'Op: %', leading to 'Id: exponent', 'Const: 2', and 'Const: 1'. This is followed by 'Assign to: result', 'Op: %', 'Op: \*', 'Id: result', 'Id: a', 'Id: p', and finally 'Assign to: a'. At the bottom, there are sections for '问题' (Questions) and '说明' (Explanations), with a '行' (Line) column on the far right.

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：

```
1  if (1 = 1) [  
2      x := 1;  
3  ];  
4  
5  for x += 1 to 3 do  
6      write 1  
7  enddo  
8  
9  x := 1
```

测试结果如下图：

The screenshot shows a web-based interface for testing Tiny programs. It has three main sections: a source code editor on the left, a syntax tree viewer on the right, and a table of errors at the bottom.

**Tiny源代码**

```
1 if (1 = 1) [  
2     x := 1;  
3 ];  
4  
5 for x += 1 to 3 do  
6     write 1  
7 enddo  
8  
9 x := 1
```

**语法分析**

```
start  
├── If  
│   ├── Op: =  
│   │   ├── Const: 1  
│   │   └── Const: 1  
│   └── Assign to: x  
├── For:  
│   ├── Assign to: x  
│   ├── Write  
│   │   └── Const: 1  
│   └── Assign to: x
```

**问题**

问题	说明	行
1	Syntax error at line 2: unexpected token -> =	2
2	Syntax error at line 2: unexpected token -> NUM, val= 1	2
3	Syntax error at line 3: unexpected token -> ]	3
4	Syntax error at line 5: unexpected token -> +=	5
5	Syntax error at line 5: unexpected token -> NUM, val= 1	5

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。