

实验三 测试报告

20203231004 计科1班 李达良

本实验tiny语法规则

program->stmt-sequence

stmt-sequence->stmt-sequence;statement | statement

statement->if-stmt | repeat-stmt | assign-stmt | read-stmt | write-stmt | plusassign-stmt | for-stmt | regex-stmt

if-stmt->if(exp) [] stmt-sequence [] else stmt-sequence [] [] (黄色的[]代表不是EBNF语法的[])

repeat-stmt->repeat stmt-sequence until exp

assign-stmt->identifier := exp

plusassign-stmt->identifier += exp

read-stmt->read identifier

write-stmt->write exp

for-stmt->for identifier:=simple-exp to simple-exp do stmt-sequence enddo | for identifier:=simple-exp downto simple-exp do stmt-sequence enddo

exp->exp orop orexp | orexp

orop->|

orexp->orexp andop andexp | andexp

andop->&

andexp->simple-exp comparison-op simple-exp | simple-exp

comparison-op->< | > | <= | >= | <>

simple-exp->simple-exp addop term | term

addop->+ | -

term->term mulop notexp | notexp

mulop->* | / | %

notexp->notop (power | notexp) | power

notop->~

power->power powop factor | factor

powop->^

factor->(exp) | number | identifier

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。

该报告最终结果：完美解析tiny程序的语法树

if-stmt

测试程序：

```
if(1)[write fact];  
if(1)[write fact]  
else[x:=1]
```

结果：

编译原理实验3 author: 李达良

— □ ×

实验三：TINY扩充语言的语法树生成

姓名：李达良 班级：计科1班 学号：20203231004

上传源程序

保存源程序

if(1)[write fact];
if(1)[write fact]
else[x:=1]

生成状态

1:if(1)[write fact];
2:if(1)[write fact]
3:else[x:=1]
生成成功!

语法树结果

▼

▼ If

Const: 1

▼ Write

Id: fact

▼ If

Const: 1

▼ Write

Id: fact

▼ Assign to: x

Const: 1

for

测试程序：

```

for i:=0 to x+5
do
  for j:=0 downto x-5
  do
    y+=1
  enddo;
  haha:=ji{嘿嘿}
enddo

```

结果：

编译原理实验3 author: 李达良

— □ ×

实验三：TINY扩充语言的语法树生成

姓名：李达良 班级：计科1班 学号：20203231004

上传源程序

保存源程序

生成语法树

```

for i:=0 to x+5
do
  for j:=0 downto x-5
  do
    y+=1
  enddo;
  haha:=ji{嘿嘿}
enddo

```

生成状态

```

1: for i:=0 to x+5
2:   do
3:     for j:=0 downto x-5
4:       do
5:         y+=1
6:       enddo;
7:     haha:=ji{嘿嘿}

```

语法树结果

- ForTo
 - Assign to: i
 - Const: 0
 - Op: +
 - Id: x
 - Const: 5
 - ForDownto
 - Assign to: j
 - Const: 0
 - Op: -
 - Id: x
 - Const: 5
 - Assign to: y
 - Const: 1
 - Assign to: haha
 - Id: ji

+=

测试程序：

```

a += 2;
b += a;
c += a + b

```

结果：

编译原理实验3 author: 李达良

— □ ×

实验三：TINY扩充语言的语法树生成

姓名：李达良 班级：计科1班 学号：20203231004

上传源程序

保存源程序

生成语法树

```
a := 2;
b := a;
c := a + b
```

生成状态

1: a := 2;
2: b := a;
3: c := a + b
生成成功!

语法树结果

▼

▼ Assign to: a

▼ Op: +

Id: a

Const: 2

▼ Assign to: b

▼ Op: +

Id: b

Id: a

▼ Assign to: c

▼ Op: +

Id: c

▼ Op: +

Id: a

Id: b

比较运算符

测试程序：

```
a := x<y;
b := x>y;
c := x=y;
d := x<=y;
e := x>=y;
f := x<>y
```

测试截图：

编译原理实验3 author: 李达良

— □ ×

实验三：TINY扩充语言的语法树生成

姓名：李达良 班级：计科1班 学号：20203231004

上传源程序

保存源程序

生成语法树

```
a := x<y;
b := x>y;
c := x=y;
d := x<=y;
e := x>=y;
f := x<>y
```

生成状态

1: a := x<y;
2: b := x>y;
3: c := x=y;
4: d := x<=y;
5: e := x>=y;
6: f := x<>y
生成成功!

语法树结果

Assign to: a

Op: <

Id: x

Id: y

Assign to: b

Op: >

Id: x

Id: y

Assign to: c

Op: =

Id: x

Id: y

Assign to: d

Op: <=

Id: x

Id: y

Assign to: e

Op: >=

Id: x

Id: y

Assign to: f

Op: <>

Id: x

Id: y

位运算

测试样例

```
x := not(123)and456 or not(not789);
y := a and (b or c) or (d and e) and not not f or g
```

结果：

编译原理实验3 author: 李达良

— □ ×

实验三：TINY扩充语言的语法树生成

姓名：李达良 班级：计科1班 学号：20203231004

上传源程序

保存源程序

```
x := not(123)and456 or not(not789);
y := a and (b or c) or (d and e) and not not f or g
```

生成状态

生成语法树

1:x := not(123)and456 or not(not789);
2:y := a and (b or c) or (d and e) and not not f or g
生成成功!

语法树结果

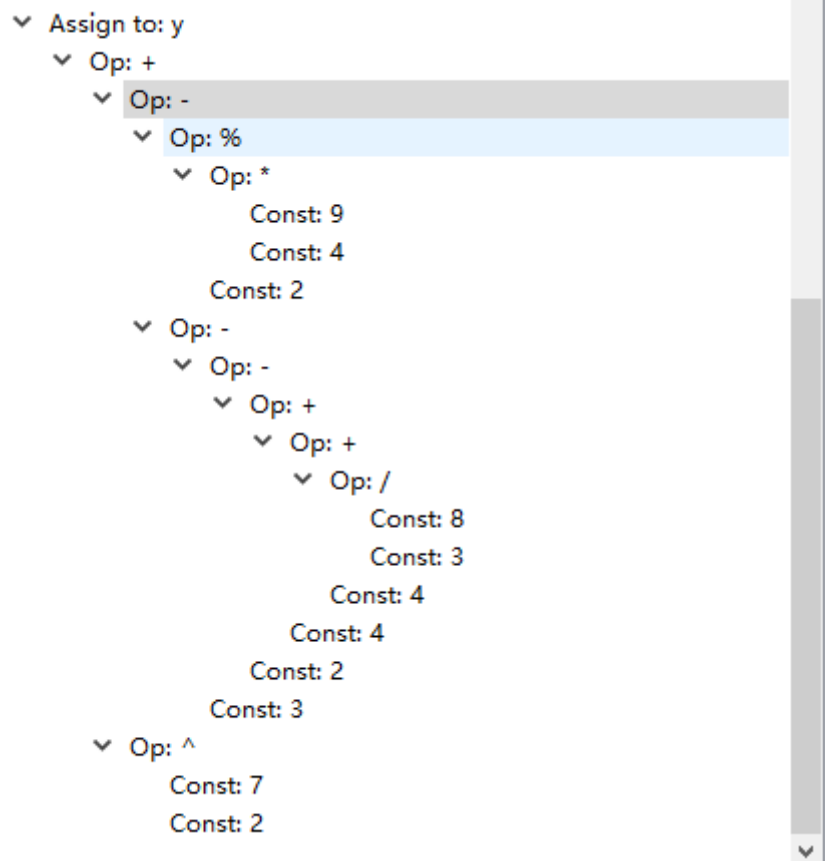
Assign to: x
 Op: OR
 Op: AND
 Op: NOT
 Const: 123
 Const: 456
 Op: NOT
 Op: NOT
 Const: 789
 Assign to: y
 Op: OR
 Op: OR
 Op: AND
 Id: a
 Op: OR
 Id: b
 Id: c
 Op: AND
 Op: AND
 Id: d
 Id: e
 Op: NOT
 Op: NOT
 Id: f
 Id: g
 Id: g
 Id: a

运算符

```
x:=(5+(((1*2%3^4)))*6^1%7);
y:= 9*4%2-(8/3+4+4-2-3)+7^2
```

结果：

```
graph TD
    Root[ ] --- Assign[Assign to: x]
    Assign --- Op1[Op: +]
    Op1 --- Const1[Const: 5]
    Op1 --- Op2[Op: %]
    Op2 --- Op3[Op: *]
    Op3 --- Op4[Op: %]
    Op4 --- Op5[Op: *]
    Op5 --- Const2[Const: 1]
    Op5 --- Const3[Const: 2]
    Op5 --- Op6[Op: ^]
    Op6 --- Const4[Const: 3]
    Op6 --- Const5[Const: 4]
    Op3 --- Op7[Op: ^]
    Op7 --- Const6[Const: 6]
    Op7 --- Const7[Const: 1]
    Op2 --- Const8[Const: 7]
```

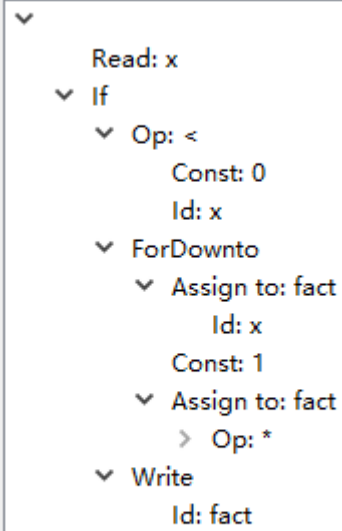


正则表达式

```
x: :=(a&b|c)?&(a&b|c)#;  
y: :=((a&b)#&c)?|d&e#
```

结果:

语法树结果



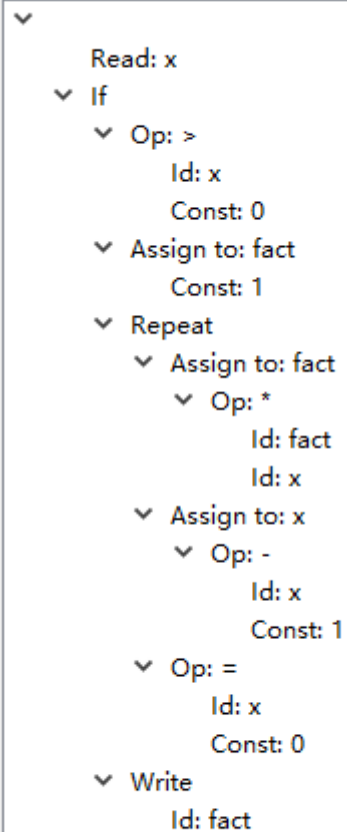
题目样例2

```
{ 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 }]
```

结果：

语法树结果



综合测试1

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

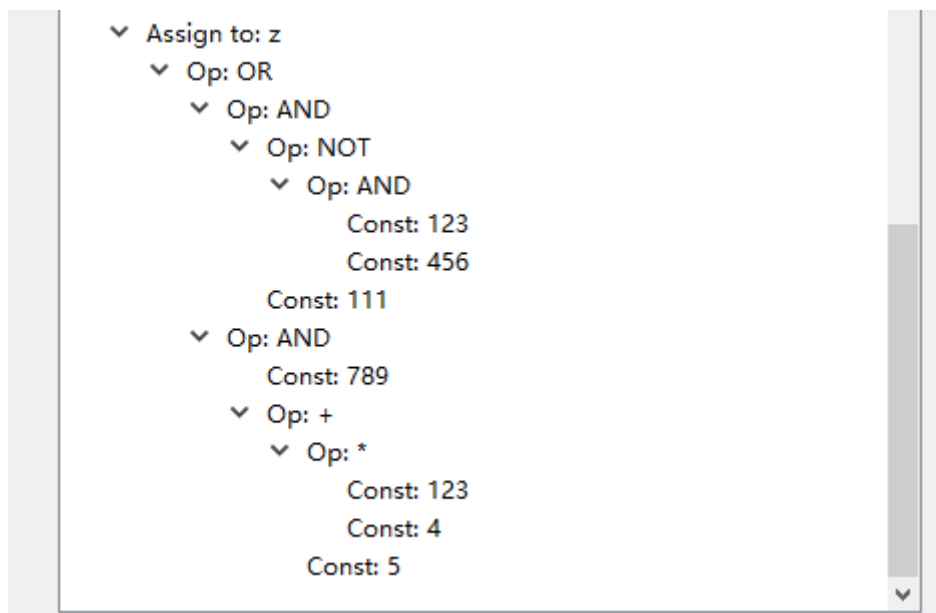
if (x>0) { don't compute if x <= 0 }
[fact := 1;
repeat
  fact := fact * x;
  x := x - 1
until x = 0;
write fact { output factorial of x }
];
x+=(1+2)*3%4^(fact);
y:=(a&b&c)#&d?;
z:= not (123 and 456 ) and 111 or 789 and 123 * 4 +5
```

结果:

语法树结果

```

  Read: x
  If
    Op: >
      Id: x
      Const: 0
    Assign to: fact
      Const: 1
    Repeat
      Assign to: fact
        Op: *
          Id: fact
          Id: x
      Assign to: x
        Op: -
          Id: x
          Const: 1
      Op: =
        Id: x
        Const: 0
    Write
      Id: fact
  Assign to: x
    Op: +
      Id: x
    Op: %
      Op: *
        Op: +
          Const: 1
          Const: 2
          Const: 3
        Op: ^
          Const: 4
          Id: fact
  Assign to: y
    Op: &
      Op: #
        Op: &
          Op: &
            Id: a
            Id: b
            Id: c
      Op: ?
        Id: d
```



综合测试2

```
{ 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 }
else
  [y:=a#&b?|c&(d)];{test}
for i := 0 to x*5
  do
for j :=3 downto 0
do
  x+=1{这里是两层的for循环}
enddo
enddo
```

结果:

语法树结果

```

  Read: x
  If
    Op: >
      Id: x
      Const: 0
    Assign to: fact
      Const: 1
    Repeat
      Assign to: fact
        Op: *
          Id: fact
          Id: x
      Assign to: x
        Op: -
          Id: x
          Const: 1
        Op: =
          Id: x
          Const: 0
    Write
      Id: fact
    Assign to: y
      Op: |
        Op: &
          Op: #
            Id: a
          Op: ?
            Id: b
        Op: &
          Id: c
          Id: d
    ForTo
      Assign to: i
        Const: 0
      Op: *
        Id: x
        Const: 5
      ForDownto
        Assign to: j
          Const: 3
          Const: 0
        Assign to: x
          Op: +
            Id: x
            Const: 1
```

综合测试3

{预计这是一个很难的测试样例，你准备好了吗}
read xxx;{}

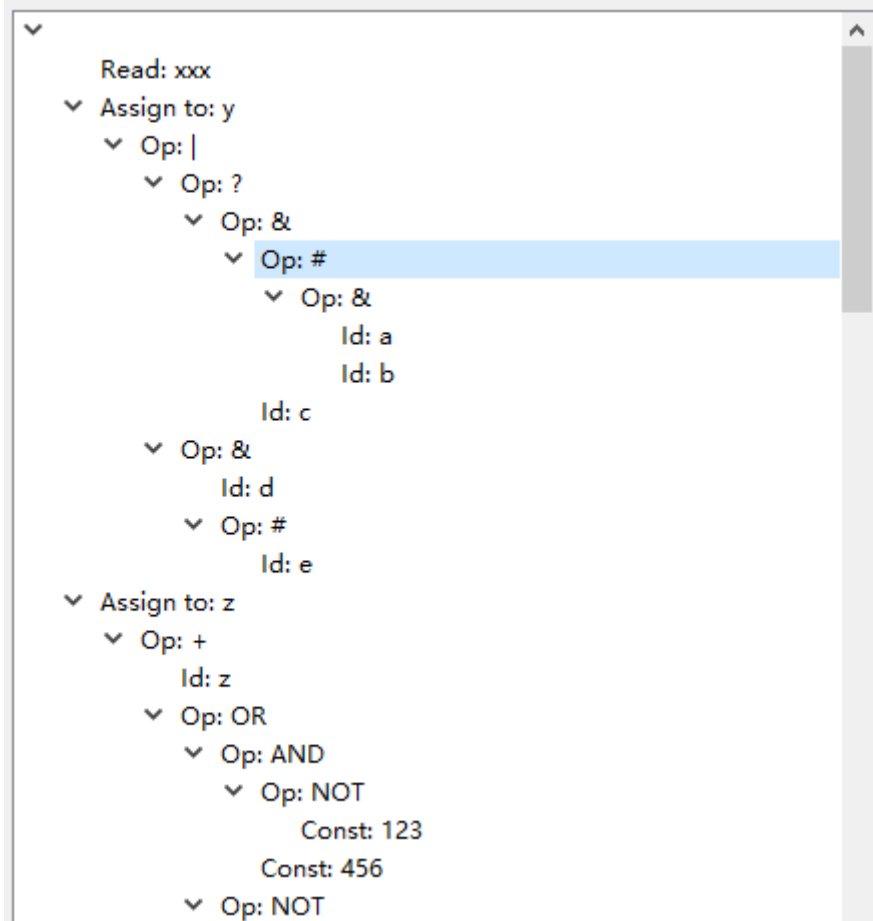
```

y:=((a&b)#&c)?|d&e#;
z+=not(123)and456 or not(not789);
x:=(5+(((1*2%3^4)))*6^1%7);
if (x<y)[
    if(1)
        [write fact]
    else
        [for i:=0 to x+5
            do
                for j:=0 downto x-5
                    do
                        y+=1
                    enddo;
                haha:=ji{嘿嘿}
            enddo;
        repeat
            true:=false;
            false+=heiheihei
        until x=0]
]

```

结果:

语法树结果



- Op: NOT
 - Const: 789
- Assign to: x
 - Op: +
 - Const: 5
 - Op: %
 - Op: *
 - Op: %
 - Op: *
 - Const: 1
 - Const: 2
 - Op: ^
 - Const: 3
 - Const: 4
 - Op: ^
 - Const: 6
 - Const: 1
 - Const: 7
- If
 - Op: <
 - Id: x
 - Id: y
 - If
 - Const: 1
- Write
 - Id: fact
- ForTo
 - Assign to: i
 - Const: 0
 - Op: +
 - Id: x
 - Const: 5
 - ForDownto
 - Assign to: j
 - Const: 0
 - Op: -
 - Id: x
 - Const: 5
 - Assign to: y
 - Op: +
 - Id: y
 - Const: 1
 - Assign to: haha
 - Id: ji
 - Repeat
 - Assign to: true
 - Id: false
 - Assign to: false

▼ Op: +
Id: false
Id: heiheihei

▼ Op: =
Id: x
Const: 0