華南師範大學

本科学生实验（实践）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：网工二班

学　　生：肖翔

学　　号：20222132002

华南师范大学教务处

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 **学　号** 20222132002

专　　业 网络工程 **年级、班级** 2022 级网工二班

课程名称 编译原理 **实验项目** TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 **实验评分**

一、实验内容

设计一个应用软件，生成 TINY 扩充语言的语法树。

（一）为 Tiny 语言扩充的语法有

1.实现改写书写格式的新 if 语句；

2.扩充算术表达式的运算符号：++（前置自增 1）、 --（前置自减 1）运算符号（类似于 C 语言的++和--运算符号，但不需要完成后置的自增 1 和自减 1）、求余%、乘方^；

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

4 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。

5.for 语句的语法规则（类似于 C 语言的 for 语言格式）：

书写格式：for(循环变量赋初值;条件;循环变量自增或自减 1) 语句序列

6.while 语句的语法规则（类似于 C 语言的 while 语言格式）：

书写格式：while(条件) 语句序列 endwhile

（二）对应的语法规则分别为：

1. 把 TINY 语言原有的 if 语句书写格式 *

if_stmt-->if exp then stmt-sequence end | | if exp then stmt-sequence else stmt-sequence end

改写为：

if_stmt-->if(exp) stmt-sequence else stmt-sequence | if(exp) stmt-sequence

2. ++（前置自增 1）、 --（前置自减 1）运算符号、求余%、乘方^等运算符号的文法规则请自行组织。

3.<=(小于等于)、>(大于)、>=(大于等于)、<>(不等于)等运算符号的文法规则请自行组织。

4.为 tiny 语言增加一种新的表达式——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，对应的文法规则请自行组织。

5.为 tiny 语言增加一种新的语句，ID::=正则表达式 （同时增加正则表达式的赋值运算符号::=）

6.为 tiny 语言增加一个符合上述 for 循环的书写格式的文法规则，

7.为 tiny 语言增加一个符合上述 while 循环的书写格式的文法规则，

8.为了实现以上的扩充或改写功能，还需要注意对原 tiny 语言的文法规则做一些相应的改造处理。

二、实验目的

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 **学　　号** 20222132002

专　　业 网络工程 **年级、班级** 2022 级网工二班

课程名称 编译原理 **实验项目** TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 **实验评分**

（1）要提供一个源程序编辑的界面，以让用户输入源程序（可输入，可保存、可打开源程序）

（2）可由用户选择是否生成语法树，并可查看所生成的语法树。

（3）实验 3 的实现只能选用的程序设计语言为：C++

（4）要求应用程序的操作界面应为 Windows 界面。

（5）应该书写完善的软件文档

三、实验文档：

（一）系统概述

1. 系统结构

系统分为 3 个模块：语句扫描模块、语句分析（生成语法树）模块、输出语法树模块。

2. 数据结构的选择

```
typedef enum { StmtK, ExpK } NodeKind;
typedef enum { IfK, RepeatK, AssignK, ReadK, WriteK, ForK, WhileK, RegexK} StmtKind;
typedef enum { OpK, ConstK, IdK } ExpKind;


#define MAXCHILDREN 4

typedef struct treeNode
{
    struct treeNode* child[MAXCHILDREN];
    struct treeNode* sibling;
    int lineno;
    NodeKind nodekind;
    // union 的特点是其所有成员共享同一块内存空间，这意味着在任何时候只能使用其中一个成员，因为它们在内存中重叠
    union { StmtKind stmt; ExpKind exp; } kind; // 取决于 TreeNode 表示的是语句节点还是表达式节点
    union {
        TokenType op;    // 操作符
        int val;         // 常量值
        char* name;      // 变量名
    } attr;      // 取决于节点的具体属性
} TreeNode;
```

图 1 语法树的数据结构

如图 1 所示，本系统主要使用了结构体 struct、联合体 union、枚举 enum、数组等数据结构。以下是对本系统部分数据结构的详细介绍。

Struct treeNode 表示语法树的结点。数组 child 表示语法树结点的孩子结点，在本系统中，for 语句具有最多的子结点（图 2），一共 4 个子结点，因此一个语法树结点至多有 4 个子结点。Kind 表示当前结点的类型（语句类型和表达式类型）。sibling 指针指向同一层级下语相邻的句结点，也就是说同一层级的语句结点会形成一个链表。lineno 表示当前结点表示的 token 所在的行数。TokenType 用于标识不同的运算符（图 3）。

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

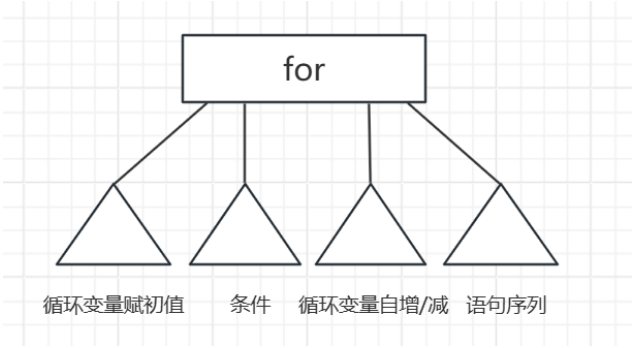
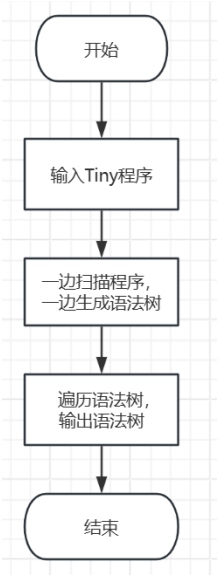


图 2 for 语法树

```
typedef enum
/* book-keeping tokens */
{
    ENDFILE, ERROR,
    /* reserved words */
    IF, THEN, ELSE, END, REPEAT, UNTIL, READ, WRITE, FOR, WHILE, ENDWHILE,
    /* multicharacter tokens */
    ID, NUM,
    /* special symbols */
    ASSIGN, EQ, PLUS, MINUS, TIMES, OVER, LPAREN, RPAREN, SEMI,
    /* 乘方，模 */
    POWER, MOD,
    /* 小于，大于，小于等于，大于等于，不等于 */
    LT, RT, LTEQ, RTEQ, NOTEQ,
    /* 前置自增，前置自减 */
    AUTOINC, AUTODEC,
    REGEX,OR,AND,CLOSE,CHOOSE,
    /* 左中括号，右中括号 */
    LPM,RPM
} TokenType;
```

图 3 TokenType

3. 程序整体大致流程的流程图（图 4）如下。



华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

图 4 程序执行大致流程

（二）实验过程

1. 写出文法规则

根据实验要求，写出扩充后的 TINY 语言的文法规则。

（注：由于本系统不支持缩进表示代码块的层级结构，而实验规定的文法规则中并未提供保留字以表示 for 语句或 if 语句的范围，因此不妨轻微改动文法规则，使用中括号（小括号和大括号已经被使用）表示 for 语句或 if 语句代码块的层次结构。

文法中的红色中括号并非 ENBF 语法，而是 if、else 或 for 的范围，是终结符号；绿色中括号表示 ENBF 语法。）

```
program → stmt-sequence
stmt-sequence → stmt-sequence ; statement | statement
statement → if-stmt | repeat-stmt | assign-stmt | read-stmt | write-stmt |
for-stmt | while-stmt | regex-stmt | autofactor
if-stmt → if ( exp ) [ stmt-sequence ] else [ stmt-sequence ] | if ( exp )
[ stmt-sequence ]
repeat-stmt → repeat stmt-sequence until exp
（注：用:=和::=来区分普通的表达式和正则表达式）
assign-stmt → identifier := exp | identifier ::= regex-exp
read-stmt → read identifier
write-stmt → write exp
for-stmt → for ( assign_stmt ; exp ; autofactor ) [ stmt-sequence ]
while-stmt → while ( exp ) stmt-sequence endwhile
（注：比较运算符的优先级比算术运算符的优先级低，因此它的文法规则层级比算术运算符的浅，因此将它的文法规则写在前面。而且比较运算符在一个表达式中只使用一次，因此无需考虑递归。）
exp → simple-exp comparison-op simple-exp | simple-exp
comparison-op → < | > | <= | >= | <>
simple-exp → simple-exp addop term | term
addop → + | -
（注：模运算符和乘、除运算符优先级一样，因此考虑将模运算和乘、除放在同一层级。）
term → term mulop power | power
mulop → * | / | %
（注：幂运算符是左结合的运算符，且优先级高于乘除，因此它的层次更加深且具有左递归。）
power → power powop autofactor | autofactor
```

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学 号 20222132002

专 业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

powop → ^

autofactor → autoOp factor | factor

autoOp → ++|--

factor → (exp) | number | identifier

（注：考虑正则表达式运算符的优先级 priority(#)=priority(?)>priority(&)>priority(|)，仿照算术表达式的文法规则，我们可以写出正则表达式文法。）

regex-exp → regex-exp orop andreg | andreg

orop → |

andreg → andreg andop topreg | topreg

randop → &

topreg → topreg topop | reg_factor

topop → # | ?

reg-factor → (regex-exp) | identifier | number

2. 改造文法

为了能够使用递归向下的分析方法写出分析程序，必须消除文法中的左递归以及回溯性，因此我们还需要使用 ENBF 语法改造文法。

program → stmt-sequence

stmt-sequence → statement {; statement}

statement → if-stmt | repeat-stmt | assign-stmt | read-stmt | write-stmt | for-stmt | while-stmt | regex-stmt | autofactor

if-stmt → if (exp) [stmt-sequence] [else [stmt-sequence]]

repeat-stmt → repeat stmt-sequence until exp

assign-stmt → identifier (:= exp | identifier ::= regex-exp)

read-stmt → read identifier

write-stmt → write exp

for-stmt → for (assign_stmt ; exp ; autofactor) [stmt-sequence]

while-stmt → while (exp) stmt-sequence endwhile

exp → simple-exp [(< | > | <= | >= | <>) simple-exp]

simple-exp → term{(+ | -) term}

term →power {(* | / | %) power}

power →autofactor {^ autofactor}

autofactor → [++|--] factor

factor → (exp) | number | identifier

regex-exp → andreg {| andreg}

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

andreg → topreg {& topreg}

topreg → reg_factor {# | ?}

reg-factor → (regex-exp) | identifier | number

3. 编写递归子程序

3.1 if 语句

下图（图 5）是 if 语句的语法树。图中虚线表示 else 部分可选择。

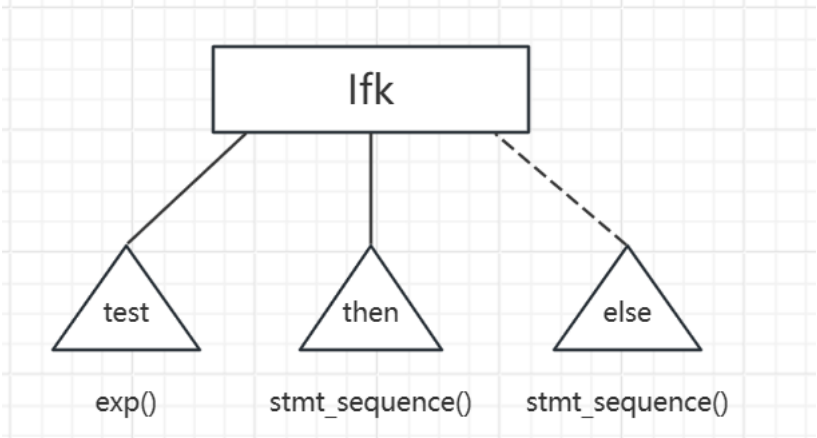


图 5 if 语法树

根据 if 的语法树和文法规则，很容易写出它的递归子程序（图 6）。

注意，对于可选的 else 部分，我们应该判断 else 保留字是否存在。若存在，则匹配保留字，并建立相应的结点；若不存在，则不进行任何处理。

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学 号 20222132002

专 业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

```
TreeNode* if_stmt(void)
{
    TreeNode* t = newStmtNode(IfK);
    match(IF);
    match(LPAREN);
    if (t != NULL) t->child[0] = exp();
    match(RPAREN);
    match(LPM);
    if (t != NULL) t->child[1] = stmt_sequence();
    match(RPM);
    if (token == ELSE) {
        match(ELSE);
        match(LPM);
        if (t != NULL) t->child[2] = stmt_sequence();
        match(RPM);
    }
    return t;
}
```

图 6 if 递归子程序

3.2 for 语句

for 语句的语法树如图 2 所示，它的递归子程序如下图（图 7）所示。

```
TreeNode* for_stmt(void)
{
    match(FOR);
    match(LPAREN);
    TreeNode* p = assign_stmt();
    TreeNode* t = NULL;
    t = newStmtNode(ForK);
    t->child[0] = p;
    match(SEMI);
    t->child[1] = exp();
    match(SEMI);
    t->child[2] = autofactor();
    match(RPAREN);
    match(LPM);
    t->child[3] = stmt_sequence();
    match(RPM);
    return t;
}
```

图 7 for 递归子程序

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

3.3 while 语句

下图（图 8）是 while 语句的语法树。

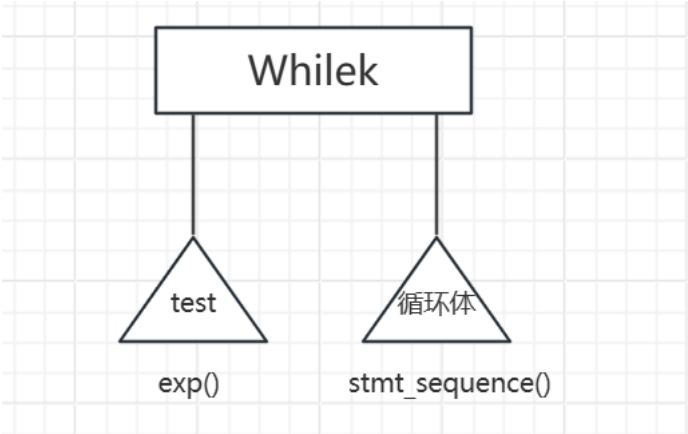


图 8 while 语法树

根据 while 的语法树和文法规则，很容易写出它的递归子程序（图 9）。

```
TreeNode* while_stmt(void){
    TreeNode* t = NULL;
    t = newStmtNode(WhileK);
    match(WHILE);
    match(LPAREN);
    t->child[0] = exp();
    match(RPAREN);
    t->child[1] = stmt_sequence();
    match(ENDWHILE);
    return t;
}
```

图 9 while 递归子程序

3.4 正则表达式语句

下图（图 10）是正则表达式的语法树，书写它必须依赖赋值语句。右结点虚线表示如果操作符是单目运算符（闭包和选择），那么构建语法树的过程中则不需要右正则表达式。

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

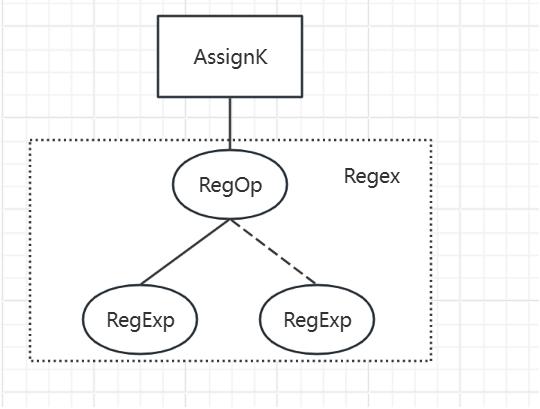


图 10 正则表达式语法树

根据正则表达式的文法规则和语法树，可以写出它的递归子程序（图 11、图 12、图 13）。

```
TreeNode* assign_stmt(void)
{
    TreeNode* t = newStmtNode(AssignK);
    if ((t != NULL) && (token == ID))
        t->attr.name = copyString(tokenString); // tokenString;
    match(ID);
    if (token == ASSIGN)
    {
        match(ASSIGN);
        if (t != NULL) t->child[0] = exp();
    }
    else if (token == REGEX)
    {
        qDebug() << "assign_stmt: " << token;
        match(REGEX);
        if (t != NULL) t->child[0] = regex_exp();
    }
    return t;
}
```

图 11 正则表达式递归子程序（1）

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

```c
TreeNode* regex_exp(void)
{
    TreeNode* t = andreg();
    while ((token == OR))
    {
        TreeNode* p = newExpNode(OpK);
        if (p != NULL) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
            match(token);
            t->child[1] = andreg();
        }
    }
    return t;
}

TreeNode* andreg(void)
{
    TreeNode* t = topreg();
    while ((token == AND))
    {
        TreeNode* p = newExpNode(OpK);
        if (p != NULL) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
            match(token);
            t->child[1] = topreg();
        }
    }
    return t;
}
```

图 12 正则表达式递归子程序（2）

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

```
TreeNode* topreg(void)
{
    TreeNode* t = reg_factor();
    while ((token == CLOSE) || (token == CHOOSE))
    {
        TreeNode* p = newExpNode(OpK);
        if (p != NULL) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
            match(token);
        }
    }
    return t;
}

TreeNode* reg_factor(void)
{
    TreeNode* t = NULL;
    switch (token) {
    case NUM:
        t = newExpNode(ConstK);
        if ((t != NULL) && (token == NUM))
            t->attr.val = atoi(tokenString);
        match(NUM);
        break;
    case ID:
        t = newExpNode(IdK);
        if ((t != NULL) && (token == ID))
            t->attr.name = copyString(tokenString);
        match(ID);
        break;
    case LPAREN:
        match(LPAREN);
        t = regex_exp();
        match(RPAREN);
        break;
    default:
        syntaxError("unexpected token -> " + getTokenString(token, tokenString).toStdString());
        printToken(token, tokenString);
        token = getToken();
        break;
    }
    return t;
}
```

图 13 正则表达式递归子程序（3）

3.5 新增算术运算符、比较运算符、前置自增和前置自减

```
TreeNode* term(void)
{
    TreeNode* t = power();
    while ((token == TIMES) || (token == OVER) || (token == MOD))
    {
        TreeNode* p = newExpNode(OpK);
        if (p != NULL) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
            match(token);
            p->child[1] = power();
        }
    }
    return t;
}
```

图 14 新增模运算

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

```
TreeNode* power(void)
{
    TreeNode* t = autofactor();
    while ((token == POWER))
    {
        TreeNode* p = newExpNode(OpK);
        if (p != NULL) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
            match(token);
            p->child[1] = autofactor();
        }
    }
    return t;
}
```

图 15 新增乘方运算

```
TreeNode* exp(void)
{
    TreeNode* t = simple_exp();
    if (token == LTEQ || token == RTEQ || token == LT || token == RT || token == NOTEQ || token == EQ) {
        TreeNode* p = newExpNode(OpK);
        if (p != NULL) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
        }
        match(token);
        if (t != NULL)
            t->child[1] = simple_exp();
    }
    return t;
}
```

图 16 比较运算递归子程序

```
TreeNode* autofactor(void){
//    qDebug() << "autofactor: " << token;
    TreeNode* t = nullptr;
    if(token == AUTOINC || token == AUTODEC){
        TreeNode* p = newExpNode(OpK);
        p->attr.op = token;
        match(token);
        t = factor();
        p->child[0] = t;
        t = p;
    }
    else{
        t = factor();
    }
    return t;
}
```

图 17 前置自增和前置自减递归子程序

华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 学　号 20222132002

专　业 网络工程 年级、班级 2022 级网工二班

课程名称 编译原理 实验项目 TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 实验评分

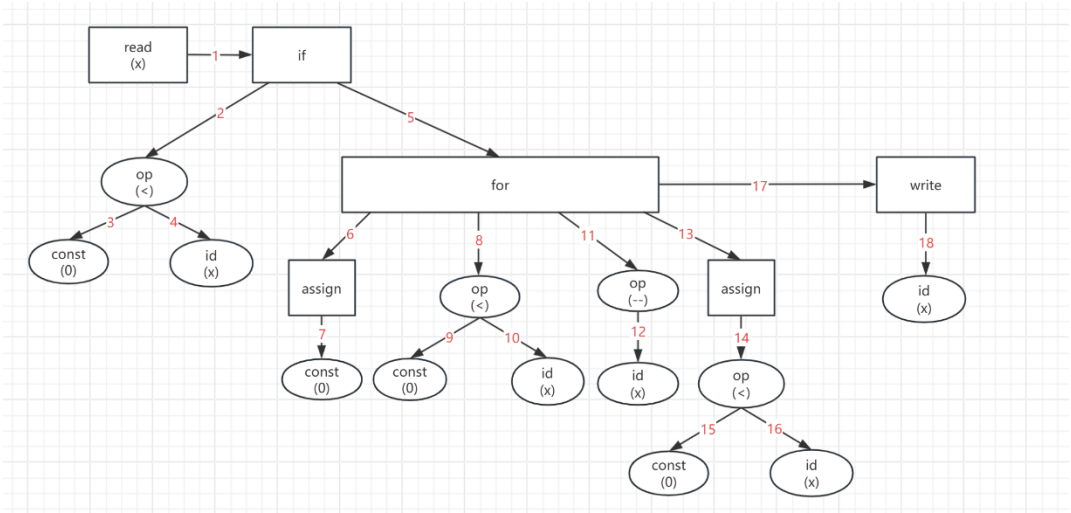
4. 遍历语法树

我使用的是深度遍历来遍历语法树。以下是深度优先遍历语法树的伪代码。

```
printTree(tree, parentNode){
    while(tree ≠ null){
        currentNode ← null;
        currentNode ← newTreeItem(parentNode);
        for i ← 0 to MAXCHILDREN{
            printTree(tree.child[i], currentNode);
        }
        tree ← tree.sibling;
    }
    return tree;
}
```

假设有一个如下的扩展后的 Tiny 源程序，那么它对应的语法树如图 18 所示，图中数字代表遍历顺序。

```
read x;
if(0<x)[
  for( fact := 1; x>0;--x)[
       fact := fact * x
  ];
  write fact
]
```



华 南 师 范 大 学 实 验 报 告

学生姓名 肖翔 **学　号** 20222132002

专　　业 网络工程 **年级、班级** 2022 级网工二班

课程名称 编译原理 **实验项目** TINY 扩充语言的语法树生成

实验时间 2024 年 11 月 12 日

实验指导老师 黄煜廉 **实验评分**

图 18 语法树遍历顺序

（三）测试

测试结果详见作业文件夹。

四、实验总结（心得体会）

通过为 Tiny 语言扩充语法并生成语法树，我对编译过程中的词法分析、语法分析、语法树构建等环节有了更直观和深入的理解。明白了如何将高级语言的语法规则转化为计算机能够处理的形式，以及如何根据这些规则进行语法分析和构建语法树。

在实验中，我自行组织各种扩充语法的文法规则，并对原 Tiny 语言的文法规则进行改造，使我学会了如何设计合理、无歧义且便于实现的文法。同时，也掌握了消除文法左递归和回溯性的方法，这对于编写正确的递归子程序至关重要。

本次实验让我深刻体会到编译原理理论知识在实际编程中的应用。只有将理论理解透彻，才能在实践中灵活运用，设计出合理的解决方案。

五、参考文献：

《编译原理复习提纲》

《编译原理及实践》