華南師範大學

本科学生实验（实践）报告

院　　系：人工智能学院

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　　学期

专　　业：　人工智能

班　　级：　1 班

学生姓名：　张斯博

华南师范大学教务处

华南师范大学实验报告

学生姓名 张斯博　　学　　号 20224001099

专　　业 人工智能　　年级、班级 22 级 1 班

课程名称 编译原理　　实验项目 TINY 扩充语言的语法树生成

实验类型　□验证　√设计　□综合　　实验时间 2024 年 10 月 27 日

实验指导老师 黄煜廉　　实验评分

一、实验内容

1.增加 while 循环；

2.增加 for 循环；

3.扩充算术表达式的运算符号：-=减法赋值运算符号（类似于 C 语言的-=）、+= 加法赋值运算符号（类似于 C 语言的+=）、求余%、乘方^，

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

（二）对应的语法规则分别为：

1.while 循环语句的语法规则：while-stmt-->while exp do stmt-sequence enddo

2.for 语句的语法规则：

（ 1) for-stmt-->for identifier:=simple-exp to simple-exp do stmt-sequence enddo　　步长递增 1

(2) for-stmt-->for identifier:=simple-exp downto simple-exp do stmt-sequence enddo　　步长递减 1

3.-=减法赋值运算符号（类似于 C 语言的-=）、+= 加法赋值运算符号、求余%、乘方^等运算符号的文法规则请自行组织。

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

5.TINY 语言的 BNF 语法规则如下：

programstmt-sequence

stmt-sequence stmt-sequence ； statement |statement

statement if-stmt |repeat-stmt |assign-stmt | read-stmt|write-stmt

if-stmtif exp then stmt-sequence end

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

repeat-stmtrepeat stmt-sequence until exp

assign-stmtidentifier := exp

read-stmtread identifier

write-stmtwrite exp

expsimple-exp comparison-op simple-exp |simple-exp

comparison-exp < | =

simple-expsimple-exp addop term | term

addop + |-

termterm mulop factor | factor

mulop * | /

factor(exp) | number | identifier

输入：一个扩充语法的 TINY 语言源程序

输出：输出生成的语法树。

二、实验目的

1.掌握 while 循环的语法树结构，学会仿照标准代码写出生成 while 循环语法树的代码，理解其基本思路和构造方法。

2. 掌握 for 循环的语法树结构，学会仿照标准代码写出生成 for 循环语法树的代码，理解其基本思路和构造方法。

3.学会扩充算数表达式的运算符号，理解根据递归代码的调用来实现算术优先级的概念

三、实验文档：

根据文档和百度网盘下载的代码，进行修改。在百度网盘中给出的代码使用了 C 语言进行编写，同时使用了文件读写和终端输入执行程序。为了理解代码内容，写出我能理解的代码，我把部分对语法树生成有用的代码改写为了 C++语言，同时，把文件读写和终端输入的用户输入模式改成了普通的运行程序后直接输入。

1. 增加 while 循环

while 循环的语法结构是：while 表达式 do 循环体 enddo。在原代码中，使用了 TreeNode* exp() 来表示表达式，使用了 TreeNode* stmt_sequence() 来表示循环体，所以只需要照葫芦画瓢，写出 while 循环的语法树生成结构即可，代码如下：

```
TreeNode* while_stmt() {
    //while 语句写法：while 表达式 do 循环体 enddo
    TreeNode* t = newStmtNode(WhileK);
    match(WHILE);
    if (t != nullptr) {
        t->child[0] = exp();
    }
    match(DO);
    if (t != nullptr) {
        t->child[1] = stmt_sequence();
    }
    match(ENDDO);
    return t;
}
```

2. 增加 for 循环

for 循环的写法和 while 循环基本相似，语法是：for ID := 表达式 to 表达

式 do 循环体 enddo，在写的时候，让当前语法节点的第一个孩子存储起始值，第二个孩子存储结束值，第三个节点存储循环体。

代码如下：

```
TreeNode* for_stmt() {
    //for 语句写法：for ID := 表达式 to 表达式 do 循环体 enddo
    TreeNode* t = nullptr;
    match(FOR);
    std::string id_name = tokenString;
    match(ID);
    match(ASSIGN);
    TreeNode* start_exp = simple_exp();

    if (token == TO) {
        t = newStmtNode(ForToK);
        match(TO);
    }
    if (t != nullptr) {
        t->attr.name = new std::string(id_name);
        //起始值
        t->child[0] = start_exp;
        //终止值
        t->child[1] = simple_exp();
        match(DO);
        //循环体
        t->child[2] = stmt_sequence();
        match(ENDDO);
    }
    return t;
}
```

3. 扩充算术表达式的运算符号

1) .添加-=和+=。这两个符号基本上可以看为等价于:=，语法是类似的，在相应的地方添加相应的代码即可，但是需要添加第一个孩子，用于存储表达式。

2) 添加%。这个符号基本上可以看为和原代码中的*，/是类似的，因为他们的算数优先级是一样的，在原代码中出现了*和/的地方都相应的添加上%即可

3) 添加^。这个符号需要新建一个新的算术优先级，优先级位于*,/,%和 factor 之间。需要特别提一嘴的是，在最后存储第二个孩子的时候，需要

再次调用表示^的函数，因为乘方是一个右结合的运算，如果不这么做会导致乘方变成左结合的运算。代码如下;

```
TreeNode* power() {
    //第四级
    TreeNode* t = factor();
    while (token == POWER) {
        TreeNode* p = newExpNode(OpK);
        if (p != nullptr) {
            p->child[0] = t;
            p->attr.op = token;
            t = p;
            match(token);
            p->child[1] = power();//因为指数运算是右递归，所以需要改
        }
    }
    return t;
}
```

4. 扩充比较运算符号

扩充 >,<=,>=,<>四个符号，只需要在原代码中，

1) .新增 token 判断是否为”>”，并判断下一个字符是否为”=”

2) .判断”<”的时候，再额外判断下一个字符是否为”=”，判断下一个字符是否为”>”

5. 额外改动

1) .在 globals.h 文件中，新增所有的新增 token，新增所有的句子类型。

2) 为了防止爆栈，让每个语法树节点都有 7 个孩子

3) 添加了跳过注释和空白的处理

4) 在可能出错的地方都做了异常处理，如果出错了，大概率是输入的文法不对，请严格按照可执行程序文件夹中的编译方法介绍编译 tiny 语言源程序

四、实验总结（心得体会）

在本次实验中，我学会了完整实现 TINY 语言中 while 循环和 for 循环

的语法树生成过程。这一过程让我深入理解了语法树的构造方法，包括如何解析循环条件、初始化循环变量、设置循环体语句序列，以及如何通过递归生成树形结构。同时，我成功扩展了 TINY 语言的功能，新增了 *、/、% 等算术运算符，以及 <、<=、>、>=、!=、== 等比较运算符，完善了条件判断和复杂表达式的支持。

在实验过程中，我克服了扩展运算符优先级处理和节点子树分配的技术难点，并通过多次调试验证了语法树的正确性。这不仅帮助我掌握了语言扩展的实现方法，也提升了对编译原理中语法分析和树形结构设计的理解。此外，实验让我积累了丰富的代码调试经验，进一步增强了逻辑思维能力和程序设计能力，为今后更复杂的语言设计与实现奠定了扎实基础。

五、参考文献：

1.实验三 Tiny 扩充语言的语法树生成_tiny 语言-CSDN 博客

2.编译原理实验 3 - 爱弹琴的小黑 - 博客园

3.guanjunyou/TINYSyntaxAnalyse: 编译原理实验：扩充 TINY 语言生成语法树

4.byyl_experiment3: 编译原理实验三

5.【编译原理大作业】Tiny+的语法树 | Four's