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 扩充语言的中间代码生成

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 扩充语言的中间代码生成

实验类型　□验证　√设计　□综合　　实验时间 2024 年 12 月 10 日

实验指导老师 黄煜廉　　实验评分

一、实验内容

（一）为 Tiny 语言扩充的语法有：

1.增加 while 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.-=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。

3.>(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。

4.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：输出所生成的中间代码（四元组）。

二、实验目的

1. 掌握 while 循环的语法树结构

学会仿照标准代码编写生成 while 循环四元组的代码，理解其基本思路和构造方法。

2. 掌握扩展算术运算符的语法树结构

学会仿照标准代码编写支持 -=、+=、% 和 ^ 的语法树生成代码，理解其基本思路和实现方式。

3. 掌握扩展比较运算符的语法树结构

学会仿照标准代码编写支持 >、<=、>= 和 <> 的语法树生成代码，理解其基本思路和递归构造方法。

4. 实现中间代码生成

掌握从语法树生成四元组中间代码的基本方法，理解代码生成的递归调用逻辑。

三、实验文档：

在本次实验中一共有三个实验要求，接下来将一一讲解实现的办法：

1. 增加 while 循环；

思路：因为本实验要求文档中没有 and 和 or 的判断需求，所以对于每个 while 循环，真出口指向的必定是当前四元组编号的下两个四元组。所以这里我采用了一种偏取巧的方式，不使用 ppt 或者上课讲到的使用链表把真出口和假出口分别连起来，并使用回填函数回填四元组编号的方式，而是简单的认为真出口指向的必定是当前四元组编号的下两个四元组。最终结果运行出来也是差不多的效果。

```
while m>n do k:=1; x:=3+4*5; enddo;
(1) (J> , m , n , 3 )
(2) (J , , , 7 )
(3) (* , 4 , 5 , t1 )
(4) (+ , 3 , t1 , t2 )
(5) (:= , t2 , , x )
(6) (J , , , 1 )
```

在实现过程中，为了不对原代码进行过多的更改，我新建了一个 void output_later(int start_index, int end_index)函数，用于输出因为 while 的假出口出口编号不能确定而导致的延后输出的四元组，使用一个全局变量 in_while 来判断当前是否应该调用 output_later 函数。初始化 in_while 函数为 false，在 while_stmt 中，让 in_while 函数为 true。当 in_while 函数为真时，调用 output_later 函数，输出完毕后，让 in_while 重新置 false。

为了配合 output_later 函数，我使用了一个全局变量 gen_map 来存储所有的四元组，记录下来方便输出，这个 gen_map 的结构体中存储的是 map<int, entry>，第一个变量表示编号，第二个变量是自定义的数据结构，表示四元组。

在输出的过程中，因为 while 语句的语法是 while exp do stmt_sequence enddo，其中 exp 函数在原代码中已经给出，但是只处理了加和减。为了引入条件判断，我使用了一个新的函数 OperandsNode rop_exp()，详情介绍请看 3.扩充比较运算符号。

2. 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^

新建关键字 SUB_ASSIGN(-=)，ADD_ASSIGN(+=)，MOD(%)，POW(^)。

更改了 void OutputOP(TokenStru token)函数，使其能正确输出 -= 和 += 。

更改了 void GetToken()函数，使其能正确识别新增的四种算术表达式。

在 OperandsNode term()函数中，新增求余符号的处理。因为求余符号的优先级和乘除是相同的。

新建了 OperandsNode power()函数，用于处理乘方，乘方的优先级高于乘除求余，但是低于数，和实验三一样，为了让乘方实现右结合的运算顺序，在函数调用的时候需要让 y = power()，power()函数的内容如下：

```
OperandsNode power()
{
    //和实验三一样的，因为乘方是右结合的，所以需要进行递归
    TokenStru op;

    char str[5];
    OperandsNode x, y, z;

    x = factor();// 获取一个操作数

    while (token.ID == POW)
    {
        op = token;
        match(token.ID);
        y = power();// 获取一个操作数

        z.flag = 1; strcpy(z.NewVar, "t");    // 产生一个新的临时变量
        _itoa(varOrder, str, 10); strcat(z.NewVar, str);
```

```
        varOrder++;  // 产生了一个新临时变量后，序号+1
        Gen(op, x, y, z);  //生成一个四元组
        x = z;
    }
    return x;
}
```

3. 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

新建关键字 GRE(大于),GEQ（大于等于），LEQ（小于等于），UEQ（不等于）。新建一个函数 OperandsNode rop_exp()用于处理比较运算符号。内容基本与 exp 函数一致。因为比较运算符号的优先级是最低的，应该排在加和减的前面，还需要更改原来代码中，所有调用 exp 函数的地方都改成调用 rop_exp。rop_exp 函数的内容如下：

```
OperandsNode rop_exp()
{
    //扩充比较表达式,比较表达式的运算优先级最低的,所以要新开一个函数在 exp 的前面
    TokenStru op;
    char str[5];
    OperandsNode x, y, z;

    x = exp();   // 获取一个操作数

    //只有遇到了比较符号才需要进入这个循环，也就是说进入这个循环的时候会出现要处理“真出口”和“假出口”的情况，这里处理生成真出口的情况
    while (token.ID == GRE || token.ID == GEQ || token.ID == LEQ || token.ID == UEQ)
    {
        op = token;
        match(token.ID);
        y = exp();    // 获取一个操作数

        //更改这里，让 z 成为指示编号的位置，初始时置空
        z.flag = 0;
        z.val = NextGen + 2;//下下个编号才是真出口的内容

        Gen(op, x, y, z);    //生成一个四元组
        x = z;
```

```
    }
    return x;
}
```

在这个函数里，我还处理了 while 循环的真出口的内容。因为在实验文档中，既没有 and 和 or，也没有要求 if 语句或者 for 语句的扩充，所以条件判断总是只能出现在 while 语句里，所以我提前处理了 while 循环的真出口的内容。

更改 void GetToken()函数，让对应位置能正确识别比较运算符号。

更改 void OutputOP(TokenStru token)函数，使其能正确输出比较运算符号。

四、实验总结（心得体会）

实验心得

在本次实验中，我学会了完整实现 TINY 语言扩展语法的解析，并成功完成了中间代码（四元组）的生成。这一过程让我深入理解了语法扩展的设计与实现，包括如何解析 while 循环、扩展算术运算符（+=、-=、%、^）、扩展比较运算符（>、<=、>=、<>），并将其转化为可供后续处理的语法树结构。

在实验过程中，我克服了以下技术难点：

扩展运算符的文法定义与优先级解析：通过调整递归调用顺序和优先级规则，确保扩展的算术和比较运算符能够正确解析。

复杂语句与表达式的语法树构造：如 while 循环中条件表达式与语句序列的嵌套处理，以及赋值语句中复合运算符的节点分配。

这次实验让我进一步掌握了编译原理中递归下降解析的核心方法，也积累了丰富的代码调试经验，提升了逻辑思维能力和问题解决能力。通过扩展 TINY 语言的功能，我对语言设计与实现的灵活性有了更深的理解，为后续深入学习和实践奠定了坚实的基础。

五、参考文献：

1. TatttLeung/SLR1Analyse: 华南师范大学 2021 级（2023 年） 编译原理 实验四 SLR(1)分析生成器

2. WA-automat/Compiler_Design_SCNU: 2022 级华南师范大学编译原理实

验

3. TatttLeung/Compiler-Principles-Labs: 华南师范大学 2021 级（2023 年） 编译原理 实验汇总