 

**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **编译技术** |
| **学生姓名：** | **熊腾浩** |
| **学生学号：** | **201930383212** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2021-2022第一学期** |

**软件学院**

**2021年11月**

# Decaf实验3

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **地 点：** | | | B7 楼 | | 233 房； | | **实验台号：** | |  | |
| **实验日期与时间：** | | | 2021.12.3 | | | | **评 分：** | |  | |
| **预习检查纪录：** | | |  | | | | **实验教师：** | |  | |
| **电子文档存放位置：** | | |  | | | | | | | |
| **电子文档文件名：** | | |  | | | | | | | |
| **批改意见：** | | |  | | | | | | | |
| 出勤情况（20%） | 课堂态度（20%） | | 实验报告质量（60%） | | | | | |
| 任务完成情况 | | 源代码注释 | | 报告内容 | |
| □ 出勤  □ 缺勤  □ 早退  □ 请假 | □ 认真  □ 不认真 | | □ 完成  □ 未完成 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | |

# 报告内容

1、实验目标

Implementing a Scanner for decaf

You are to write a lexical analyzer/scanner for the language decaf

Goals

1. The input of the scanner is a source code file and the output of the

scanner is a stream of tokens.

2. Your scanner should go for longest possible match i.e. a string ‘:=’is to be identified as ‘ass-symbol’and not as ‘:’and ‘=’.

3. Token is represented as (Kind, Value). We use the following symbols to denote different kinds of tokens KEY denotes reserved words SYM denotes special symbols ID denotes identifiers NUM denotes numeric constants STR denotes string constants

4. Check lexical errors: giving meaning error messages and the lines where errors occur. The kinds of lexical errors are:

- Illegal character, that is, scanner may recognize a character that is not in the alphabet of Decaf, such as @ is an illegal character

- The right bracket of a STRING is lost, such as "scanner

实验二：

1. 参考第4讲Top-down Parsing中Recursive-Descent parsing知识理解读懂提供的源码，在其基础上添加注释。

2. 提供的源码中没有实现对for语句进行语法分析的功能，首先在lexer中加入对for关键字的词法识别，然后尝试参考源码中parser包与inter包中的while、do语句、else语句的语法分析程序的实现，扩展parser让其也能够识别for语句语法，并能检测其语法错误。

实验三：

1.参考第6讲中Semantic Analysis of a Program知识理解读懂提供的Decaf语言语义分析源码，重点理解对变量声明与使用、表达式和语句进行语义分析的代码，在其基础上添加相关的注释。 ¡变量声明与使用时的符号表处理 ¡表达式中的类型检查：算术运算表达式操作数需要类型兼容，注意理解inter包中arith节点的构造函数；逻辑运算表达式操作数需要为bool类型，注意理解inter包中logical节点的check函数与构造函数；关系运算表达式左部和右部应该类型相同，注意理解inter包中rel节点的check函数； ¡语句中的类型检查：赋值语句左部和右部需要类型兼容，注意理解inter包中set节点的check函数与构造函数；控制语句(if, else, do, while语句等)中的条件表达式需要是bool类型，注意理解inter包中if, else, do, while等节点的构造函数；

¡2.提供的源码中没有对for语句的条件表达式进行类型检查，参考源码中inter包中的while语句的类型检查代码，实现对for语句的类型检查，并能提示类型检查的相关语义错误。

¡3. 在第二次语法分析实验报告的基础上继续添加撰写完成语义分析的实验报告，每位同学在2021年12月23日前在教学在线提交完整的实验报告与源代码。

2、实验环境

（实验使用的软件/硬件环境）

Ubuntu 18.04 LTS，openjdk 11.0.11 2021-04-20，javac 11.0.11

3、关键数据结构和核心算法

3.1 数据结构

无

3. 2 核心算法

实验一（词法分析器）：

 // comment

        if (peek == '/') {

            readch();

            if (peek == '/') {

                while (peek != '\n' && peek != '\r' && peek != '\0') {

                    readch();

                }

            } else if (peek == '\*') {

                while (true) {

                    readch();

                    if (peek == '\0') {

                        throw new Error(

                            String.format("ERROR %d:%d: comment not terminated",

                                          line, column));

                    }

                    if (peek == '\*') {

                        if (readch('/')) {

                            return Word.Comment;

                        }

                    }

                    if (peek == '/') {

                        if (readch('\*')) {

                            throw new Error(String.format(

                                "ERROR %d:%d: comment not terminated", line,

                                column));

                        }

                    }

                    if (peek == '\n') {

                        line++;

                        column = 0;

                    }

                }

            }

            if (peek == '\n' || peek == '\r') {

                return Word.Comment;

            }

        }

// string

        if (peek == '"') {

            StringBuffer b = new StringBuffer();

            for (;;) {

                if (readch('"')) {

                    String s = b.toString();

                    Word w = (Word) words.get(s);

                    if (w != null) return w;

                    w = new Word(s, Tag.STRING);

                    words.put(s, w);

                    return w;

                } else if (peek == '\n') {

                    throw new Error(String.format("Error: %d:%d: Missing \"",

                                                  line, column));

                } else {

                    b.append(peek);

                }

            }

        }

// number

        if (Character.isDigit(peek)) {

            int v = 0;

            if (peek == 0) {

                // hex number

                readch();

                if (peek == 'x' || peek == 'X') {

                    for (;;) {

                        readch();

                        if (peek >= '0' && peek <= '9')

                            v = v \* 16 + peek - '0';

                        else if (peek >= 'a' && peek <= 'f')

                            v = v \* 16 + peek - 'a' + 10;

                        else if (peek >= 'A' && peek <= 'F')

                            v = v \* 16 + peek - 'A' + 10;

                        else

                            break;

                    }

                    return new HexNum(v);

                } else if (peek == '.') {

                    double x = v;

                    double d = 10;

                    for (;;) {

                        readch();

                        if (peek == 'e' || peek == 'E') {

                            // double number

                            readch();

                            int sym = 1;

                            if (peek == '+' || peek == '-') {

                                if (peek == '-') sym = -1;

                                readch();

                            } else {

                                throw new Error(String.format(

                                    "ERROR: %d:%d: Invalid double number", line,

                                    column));

                            }

                            int exp = 0;

                            while (Character.isDigit(peek)) {

                                exp = 10 \* exp + Character.digit(peek, 10);

                                readch();

                            }

                            return new Real(x \* Math.pow(10, exp \* sym));

                        } else {

                            if (!Character.isDigit(peek)) break;

                            x = x + Character.digit(peek, 10) / d;

                            d = d \* 10;

                        }

                    }

                    return new Real(x);

                }

            } else {

                do {

                    v = 10 \* v + Character.digit(peek, 10);

                    readch();

                } while (Character.isDigit(peek));

                if (peek != '.') return new Num(v);

                double x = v;

                double d = 10;

                for (;;) {

                    readch();

                    if (peek == 'e' || peek == 'E') {

                        // double number

                        readch();

                        int sym = 1;

                        if (peek == '+' || peek == '-') {

                            if (peek == '-') sym = -1;

                            readch();

                        } else {

                            throw new Error(String.format(

                                "ERROR: %d:%d: Invalid double number", line,

                                column));

                        }

                        int exp = 0;

                        while (Character.isDigit(peek)) {

                            exp = 10 \* exp + Character.digit(peek, 10);

                            readch();

                        }

                        return new Real(x \* Math.pow(10, exp \* sym));

                    } else {

                        if (!Character.isDigit(peek)) break;

                        x = x + Character.digit(peek, 10) / d;

                        d = d \* 10;

                    }

                }

                return new Real(x);

            }

        }

// identifier or keyword

        if (Character.isLetter(peek)) {

            StringBuffer b = new StringBuffer();

            do {

                b.append(peek);

                readch();

            } while (Character.isLetterOrDigit(peek) ||

                     peek == '\_'); // '\_' is allowed

            String s = b.toString();

            Word w = (Word) words.get(s);

            if (w != null) return w;

            w = new Word(s, Tag.ID);

            words.put(s, w);

            return w;

        }

实验二（语法分析器）：

case Tag.FOR:

            For fornode = new For();

            savedStmt = Stmt.Enclosing;

            Stmt.Enclosing = fornode;

            match(Tag.FOR);

            match('(');

            s1 = assignment();

            match(';');

            x = bool();

            match(';');

            s2 = assignment();

            match(')'); // for (s1;x;s2)

            s = stmt();

            fornode.init(s1, x, s2, s);

            Stmt.Enclosing = savedStmt; // reset Stmt.Enclosing

            return fornode;

        case '{':

            return block();

        default:

            return assign();

        }

}

Stmt assignment() throws IOException {

        Stmt stmt;

        Token t = look;

        match(Tag.ID);

        Id id = top.get(t);

        if (id == null) error(t.toString() + " undeclared");

        if (look.tag == '=') { // S -> id = E ;

            move();

            stmt = new Set(id, bool());

        } else { // S -> L = E ;

            Access x = offset(id);

            match('=');

            stmt = new SetElem(x, bool());

        }

        return stmt;

    }

package inter;

import symbols.Type;

public class For extends Stmt {

    Expr expry;

    Stmt stmt;

    Stmt stmtx;

    Stmt stmtz;

    public For() {

        expry = null;

        stmt = null;

        stmtx = null;

        stmtz = null;

    }

    public void init(Stmt x, Expr y, Stmt z, Stmt s) {

        stmtx = x;

        expry = y;

        stmtz = z;

        stmt = s;

        if (expry.type != Type.Bool) expry.error("boolean required in For");

    }

    public void gen(int b, int a) {

    }

    public void display() {

        emit("stmt : for begin");

        stmt.display();

        emit("stmt : for end");

    }

}

实验三：

package inter;

import symbols.Type;

import inter.Set;

public class For extends Stmt {

    Expr expr;

    Stmt stmt1;

    Stmt stmt2;

    Stmt stmt3;

    public For() {

        expr = null;

        stmt1 = null;

        stmt2 = null;

        stmt3 = null;

    }

    public void init(Stmt s1, Expr x, Stmt s2, Stmt s3) {

        expr = x;

        stmt1 = s1;

        stmt2 = s2;

        stmt3 = s3;

        if (expr.type != Type.Bool)

            expr.error("boolean required in for"); // assign x = bool

        if (stmt1.getClass().getName() != Set.class.getName())

            expr.error("assignment required in for");

        if (stmt2.getClass().getName() != Set.class.getName())

            expr.error(

                "assignment required in for"); // assign s1, s2 = assignment

    }

    public void gen(int b, int a) {

    }

}

3. 3 错误处理

实验一：

字符串常量没有右双引号报错

多行注释嵌套报错

双精度数不符合格式要求报错

多行注释没有结束符报错

实验二：

For 循环语法错误

实验三：

For循环语义错误

3.4 运行结果

实验一：

运行输入：

class Main {

static void main() {

class Fibonacci f = new class Fibonacci();

Print.print(f.get(ReadInteger()));

}

}

/\*\*

\* Fibonacci

\*/

class Fibonacci {

public int get(int n) {

if (n == 0) {

return 0;

}

if (n == 1) {

return 1;

}

return get(n - 1) + get(n - 2);

}

}

运行输出（已将换行符改为空格）：

(KEY , class) (ID , Main) (SYM , {) (KEY , static) (KEY , void) (ID , main) (SYM , () (SYM , )) (SYM , {) (KEY , class) (ID , Fibonacci) (ID , f) (SYM , =) (KEY , new) (KEY , class) (ID , Fibonacci) (SYM , () (SYM , )) (SYM , ;) (ID , Print) (SYM , .) (KEY , print) (SYM , () (ID , f) (SYM , .) (ID , get) (SYM , () (ID , ReadInteger) (SYM , () (SYM , )) (SYM , )) (SYM , )) (SYM , ;) (SYM , }) (SYM , }) (KEY , class) (ID , Fibonacci) (SYM , {) (KEY , public) (KEY , int) (ID , get) (SYM , () (KEY , int) (ID , n) (SYM , )) (SYM , {) (KEY , if) (SYM , () (ID , n) (SYM , ==) (NUM , 0) (SYM , )) (SYM , {) (KEY , return) (NUM , 0) (SYM , ;) (SYM , }) (KEY , if) (SYM , () (ID , n) (SYM , ==) (NUM , 1) (SYM , )) (SYM , {) (KEY , return) (NUM , 1) (SYM , ;) (SYM , }) (KEY , return) (ID , get) (SYM , () (ID , n) (SYM , -) (NUM , 1) (SYM , )) (SYM , +) (ID , get) (SYM , () (ID , n) (SYM , -) (NUM , 2) (SYM , )) (SYM , ;) (SYM , }) (SYM , })

实验二；

运行输入：{int a;int b;int c;while(true){if(true) for(a=b;a<b;a=a+1) a=b;}}

运行输出：

stmt : while begin

stmt : if begin

stmt : for begin

assignment

stmt : for end

stmt : if end

stmt : while end

实验三：

运行输入：

{  
int a; int b; int c;int min;  
if(a<b&&c>b)  
{   
min=a;  
}  
while(min+a<c&&a<c||b<c)  
{  
a=a+1;  
}

}

运行输出：

L1: iffalse a < b goto L3

iffalse c > b goto L3

L4: min = a

L3: t1 = min + a

iffalse t1 < c goto L6

if a < c goto L5

L6: iffalse b < c goto L2

L5:L7: a = a + 1

goto L3

L2:

4、总结

体会：学习了词法分析器的逻辑，学习了语法分析器的逻辑，学习了语义分析的方法。