**实验任务书**

**(Decaf compiler的实现)**

课程名称：编译技术

实验学时：16学时

综合性、设计性实验：□是 🗹否

面向专业和班级： 软件工程 2016级5、6班、卓越班

学生人数：101（5、6班）+ 50（卓越班）

任务编制人：课程负责人

# 实验目的

本课程实验目标是参考和理解教材附录的源码，采用Java语言实现decaf语言的一个编译器，完成decaf语言的词法分析、语法分析、语义分析以及中间代码的生成。

* 1. 实验1: 实现decaf语言的词法分析程序
  2. 实验2: 实现decaf语言的语法分析程
  3. 实验3: 实现语义分析和中间代码生成
  4. 实验4: 基于前三次实验，实现完整的decaf语言编译器

# 实验环境

1. 硬件环境需求

Pentium4 2.8及以上，至少1G以上RAM

1. 软件环境需求

Linux或windowXP以后版本，JVM, Eclipse IDE, Word文档编辑器

# 实验内容

**【注】1）每次实验阅读源代码，均要求对源代码进行注释，注释粒度应到每个类及其核心属性、方法（核心算法）；**

**2）要求对每次实验运行结果截屏，粘贴到实验报告中。**

1. 实验1：实现decaf语言的词法分析程序（扫描程序）
   * + - * 扫描程序的输入是源代码文件，输出是token串。
         * 需要遵循“最长子串匹配原则”，比如“:=”识别为assign-symbol，而不是“:”和“=”。
         * Token 以键值对 (Kind, Value)的形式表示其中：

**KEY**  reserved words

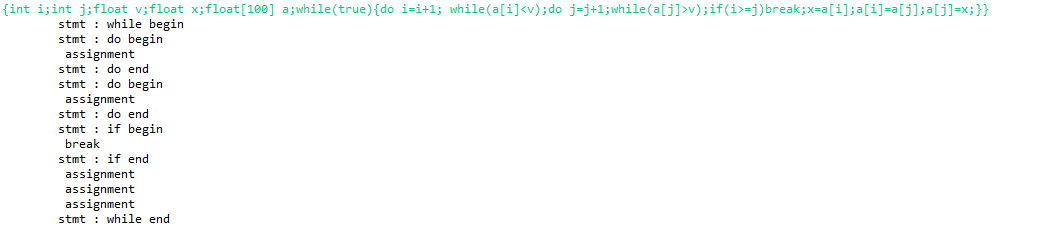
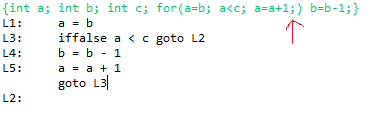
**SYM**  special symbols

**ID** identifiers

**NUM**  numeric constants

**STR**  string constants

* + - * + 检测词法错误：给出有意义的错误信息和错误发生的行号。例如字符@并非decaf程序中的合法符号，若这个字符在注释以外出现，则需要提示一个词法错误。

1. 实验2：实现decaf语言的语法分析程序
   * + - * 扫描程序的输出token串作为输入，输出语法树。
         * 递归下降分析法。
         * 实现for语言结构。
         * 检测语法错误
         * 输出结果为语法树，输出形式如下：
         * 
2. 实验3: decaf语义分析和中间代码生成
   * + - * 采用教材的语法制导的代码生成方法
         * 添加for语句的语义分析和中间代码生成
         * 语义分析主要完成类型检查，包括
       - 变量在使用前需要声明
       - 类型检查
       - 条件表达式需要是bool类型
       - 操作数需要类型兼容
       - 赋值左部和右部类型相同
         * 实验结果采用三地址代码表达，如
         * 
3. 实验4: decaf语言完整的编译器
   * + - * 集成前三次实验的词法、语法、语义分析和中间代码生成
         * 调试、测试decaf编译器使其正常执行

# 实验课程安排

* 实验1: 2017-11- 1 （5-8节卓越班）（9-12节5、6班）
* 实验2: 2017-12-6（5-8节卓越班）、2017-12-13（9-12节56班）
* 实验3: 2017-12-20（5-8节卓越班）、2017-12-21（5-8节5、6班）
* 实验4: 2017-12-27（5-8节卓越班）、2017-12-25（1-4节5、6班）

**【注】 每次实验完成后要求提交纸质版实验报告；同时，实验报告及源代码电子档要求提交到教学在线（具体提交时间根据实际情况另行通知）；**

# 备注

1. Decaf语言介绍

Decaf is a strongly typed, object­oriented language with support for inheritance and encapsulation.By design, it has many similarities with C/C++/Java, so you should find it fairly easy to pick up.  Keep in mind it is not an exact match to any of those languages.  The feature set has been trimmed down and simplified to keep the programming projects manageable.  Even so, you'll still find the language expressive enough to build all sorts of nifty object­oriented programs.

1. Decaf基本文法规则
   * + - * The keywords of the language are the following:

|  |
| --- |
| **void int double bool string class true false null this extends for while if else return new NewArray Print ReadInteger ReaderLine** |

All keywords are reserved.

* + - * + Operators and punctuation characters used by the language includes:

|  |
| --- |
| **+ - \* / % < <= > >= = == != && || ! ; , . [] [ ] ( ) { }** |

* + - * + Whitespace(i.e. spaces, tabs, and newlines)serves to separate tokens, but is otherwise ignored. Keywords and identifiers must be separated by whitespace or a token that is neither a keyword nor an identifier. **Ifintthis** is a single identifier, not three keywords. **If(23this** scans as four tokens.
        + An integer constant can either be specified in decimal (base 10) or hexadecimal (base 16). A decimal integer is a sequence of decimal digits. A hexadecimal integer must begin with 0X or 0x(that is a zero, not the letter oh) and is followed by a sequence of hexadecimal digits.
        + Hexadecimal digits include the decimal digits and the letters a through f (either upper or lowercase). For example, the following are valid integers:8, 012, 0x0, 0X12aE.
        + A double constant is a sequence of decimal digits, a period, followed by any sequence of digits (maybe none). Thus, .12 is not valid, but both 0.12and 12. are. A double can also have an optional exponent, e.g., 12.2E+2. For a double in this sort of scientific notation, the decimal point is required, the sign of the exponent is optional (if not specified, + is assumed), and the E can be lower or upper case. As above, .12E+12 is invalid, but 12.E+2 is valid. Leading zeros on the mantissa and exponent are allowed.
        + A string constant is a sequence of characters enclosed in double quotes. Strings can contain any character except a newline or double quote. A string must start and end on a single line; it cannot be split over multiple lines:
        + An identifier is a sequence of letters, digits, and underscores, starting with a letter. Decaf is casesensitive,e.g., if is a keyword, but IF is an identifier; binky and Binky are two distinct identifiers.
        + A single-line comment is started by // and extends to the end of the line. C-style comments start with /\* and end with the first subsequent \*/. Any symbol is allowed in a comment except the sequence \*/ which ends the current comment. C-style comments do not nest.

1. Decaf程序样例programs in decaf

*class Main {*

*static void main(){*

*class Fibonacci f = New Fibonacci();//new a Fibonacci*

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

*}*

*}*

*/\*\**

*\*Fibonacci*

*\*/*

*class Fibonacci {*

*int get(int i){*

*if(i<2){*

*return 1;*

*}*

*return get(i-1) + get(i-2);*

*}*

*}*

*/\*end.\*/*