**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **编译技术** |
| **学生姓名：** | **葛子午** |
| **学生学号：** | **202330550381** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2024-2025第二学期** |

**软件学院**

**2025年06月**

# 实验四 中间代码生成

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **地 点：** | | | B7楼 | | 331房； | | **实验台号：** | |  | |
| **实验日期与时间：** | | | 2025.5.17 | | | | **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**

实现Decaf编译器中对for语句的中间代码生成功能，通过扩展inter包中For类的gen函数，将for循环语句转换为三地址码（if x goto L1等形式），并验证生成的中间代码的正确性。

**2、实验环境**

操作系统: Windows11

开发语言: Java (JDK 17)

开发工具: IntelliJ IDEA

测试用例: 包含for语句的Decaf源程序（如文档1末尾的示例）

**3、关键数据结构和核心算法**

**3.1 数据结构**

**3. 2 核心算法**

根据属性文法规则实现For.gen()函数：



**3.3 错误处理**

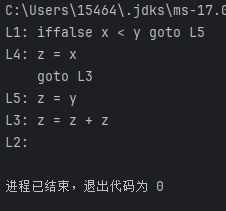
① 空循环体检测：若body为空，抛出NullPointerException。

② 无效条件表达式：若condition非布尔类型，报错"Condition must be boolean"。

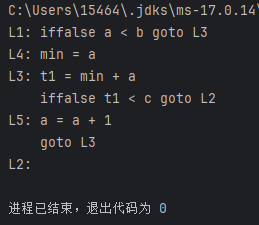
③ 标签冲突：通过newlabel()自动分配唯一标签避免冲突。

**3.4 运行结果**

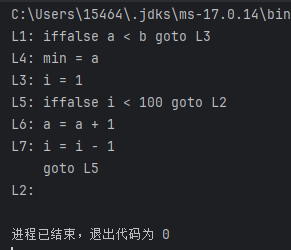
（1）测试案例1



（2）测试案例2



（3）测试案例3



**4、总结**

在实现for循环的中间代码生成时，重点关注了三个关键方面：首先通过标签管理机制，利用newlabel()函数动态生成唯一的跳转标签（如L1、L2等），确保控制流转移的准确性；其次采用循环结构分解策略，将for语句划分为初始化→条件判断→循环体→更新语句四个逻辑部分，每部分都通过独立标签进行管理；最后基于继承复用原则，充分利用现有框架中的Stmt.gen()和Expr.jumping()方法，保持与其它控制结构（如while、if）的代码生成一致性。

在实现过程中，解决了两个关键问题：针对更新语句未正确跳回条件判断的问题，通过显式添加goto指令确保循环逻辑的闭合性；针对循环体与更新语句标签冲突的问题，采用独立标签分配策略为每个逻辑块创建唯一标签（如L3和L5）。为提升实现质量，建议增加嵌套循环（如包含if的复杂循环）的测试用例以验证鲁棒性，同时优化标签命名方案（如使用FOR\_COND\_1等语义化标签）增强中间代码的可读性和可维护性。