# 编译第十四次作业

> 22373407 王飞阳

## 1.类 C 语言 `return` 语句的属性翻译文法及其动作处理程序

### 属性翻译文法

#### 文法（参考 SysY 的 `return` 语句文法）

```
<stmt> ::= 'return' [Exp] ';'
```

#### 属性与动作处理程序

为了支持语义翻译，需要将 `return` 语句的结果传递给调用者。下面给出属性文法的语义动作：

- **属性**：
  - `return_stmt.value`：保存 `return` 语句中返回的表达式的值。
  - `expression.value`：计算并返回表达式的值。
- **语义动作**：
  - `return_stmt.value` 的值为 `expression.value`。
  - `expression.value` 的值根据表达式计算得出。

#### 动作处理程序（伪代码）

```
return_stmt → 'return' expression ';' {
    return_stmt.value = expression.value;
    generate_code("return " + return_stmt.value); // 生成汇编或中间代码
}

expression → ... { /* 根据表达式的类型进行计算 */ }
```

- 在此，`expression.value` 可以通过计算表达式的具体值（如 `+`, `-`, `*`, `/` 等运算）得到。
- `generate_code` 用于将 `return` 语句转译成汇编或中间代码，通常将返回值存储到寄存器或栈中，以供调用方使用。

## 2. 类 C 语言 `for` 语句的属性翻译文法及其动作处理程序

### 文法（参考 SysY 的 `for` 语句文法）

```
<Stmt> ::= 'for' '(' [ForStmt] ';' [Cond] ';' [ForStmt] ')' Stmt
<Stmt> ::= 'break' ';' | 'continue' ';'
```

### 属性与动作处理程序

在 `for` 语句中，控制条件和循环跳转需要相应的标签标记，以实现正确的控制流。通过设置属性和生成中间代码，可以将其实现为伪汇编形式的控制流。

- **属性**：
  - `for_stmt.start_label`：循环的开始标签。

    - `for_stmt.end_label`：循环的结束标签，用于 `break` 语句。
    - `for_stmt.continue_label`：循环更新表达式标签，用于 `continue` 语句。
    - `cond_expr.value`：循环条件的值，用于决定循环是否继续。
- **语义动作**：
    - 生成 `start_label`、`end_label` 和 `continue_label` 标签。
    - `break` 跳转到 `end_label`，`continue` 跳转到 `continue_label`。

## 动作处理程序（伪代码）

```
for_stmt → 'for' '(' init_expr ';' cond_expr ';' update_expr ')' statement {
    for_stmt.start_label = generate_label();
    for_stmt.end_label = generate_label();
    for_stmt.continue_label = generate_label();

    generate_code(init_expr.code); // 初始化表达式代码
    generate_code(for_stmt.start_label + ":"); // 开始标签

    // 条件判断
    generate_code("if " + cond_expr.value + " == 0 goto " + for_stmt.end_label);
    generate_code(statement.code); // 循环体

    generate_code(for_stmt.continue_label + ":"); // continue标签
    generate_code(update_expr.code); // 更新表达式
    generate_code("goto " + for_stmt.start_label); // 回到开始标签

    generate_code(for_stmt.end_label + ":"); // 结束标签
}

break_stmt → 'break' ';' {
    generate_code("goto " + for_stmt.end_label);
}

continue_stmt → 'continue' ';' {
    generate_code("goto " + for_stmt.continue_label);
}
```

- `generate_label()` 生成一个新的标签，用于控制循环跳转。
- `init_expr.code`、`cond_expr.value`、`update_expr.code` 分别对应循环的初始化、条件判断和更新表达式。
- `statement.code` 包括循环体内部的代码，可包含 `break_stmt` 或 `continue_stmt`。
- `break` 跳转至 `end_label` 实现循环的结束，`continue` 跳转至 `continue_label` 进入下次循环的条件判断。