# stage-3

## 一、实验内容

1. 作用域栈（`ScopeStack`）

   本阶段引入作用域与块语句。新增 `ScopeStack` 以实现对不同作用域的符号的区分。`ScopeStack` 类中以变量 stack（一个 Scope 的列表）来存储每个作用域的符号，当出现一个新的块语句时，就新建一个作用域并压入栈中，离开时将这个作用域弹出。仿照 `Scope` 类，在 `ScopeStack` 类中加入 `declare`，`lookup`，`lookup_current` 等函数完成符号声明、在全部作用域查找查找、在当前作用域查找等功能。

2. namer

   在 namer 中，将先前的 `Scope` 类改为 `ScopeStack` 类。在 `visitDeclaration` 函数中，将先前的在作用域查找的函数 `ctx.lookup`，改为在当前作用域查找的函数 `ctx.lookup_current`。在 `visitBlock` 和 `visitFunction` 函数中，在继续检查 `body` 之前，先新建一个作用域并把它压入作用域栈中，在检查完 `body` 之后，将这个作用域栈弹出。

3. 后端

   更改 `dataflow/cfg.py`，在 `CFG` 类的构造函数中，从 BasicBlock 0 开始做 DFS，找出所有可达的块 (reachable_blocks)，在 iterator 中，只返回可达的块。

## 二、思考题

1. 请画出下面 MiniDecaf 代码的控制流图。

```
int main(){
 int a = 2;
 if (a < 3) {
     {
         int a = 3;
         return a;
     }
     return a;
 }
}
```

   以上代码生成的 TAC 代码如下：

```
# block 1
FUNCTION<main>:
    _T1 = 2
    _T0 = _T1
    _T2 = 3
    _T3 = (_T0 < _T2)
    if (_T3 == 0) branch _L1
    # block 2
    _T5 = 3
    _T4 = _T5
```

```
    return _T4
    # block 3
    return _T0
# block 4
_L1:
    return
```

其控制流图为：