编译原理 Lab7 实验报告

姓名：熊丘桓

学号：201250127

邮箱：eaglebear@smail.nju.edu.cn

1. 实现功能

本次实验完成了以下功能：

1. while 循环
2. 循环控制

2. 实验设计

本次实验新增了 visitWhileStmt 等方法，使用栈属性维护当前最内层的循环对应的 block 。核心代码如下：

```
@Override
public LLVMValueRef visitWhileStmt(SysYParser.WhileStmtContext ctx) {
    LLVMBasicBlockRef whileCondition = LLVMAppendBasicBlock(currentFunction,
"whileCondition");
    LLVMBasicBlockRef whileBody = LLVMAppendBasicBlock(currentFunction, "whileBody");
    LLVMBasicBlockRef afterWhile = LLVMAppendBasicBlock(currentFunction, "afterWhile");

    LLVMBuildBr(builder, whileCondition);

    LLVMPositionBuilderAtEnd(builder, whileCondition);
    LLVMValueRef condVal = this.visit(ctx.cond());
    LLVMValueRef cmpResult = LLVMBuildICmp(builder, LLVMIntNE, zero, condVal,
"cmp_result");
    LLVMBuildCondBr(builder, cmpResult, whileBody, afterWhile);

    LLVMPositionBuilderAtEnd(builder, whileBody);
    whileConditionStack.push(whileCondition);
    afterWhileStack.push(afterWhile);
    this.visit(ctx.stmt());
    LLVMBuildBr(builder, whileCondition);
    whileConditionStack.pop();
    afterWhileStack.pop();
    LLVMBuildBr(builder, afterWhile);

    LLVMPositionBuilderAtEnd(builder, afterWhile);
    return null;
}

@Override
public LLVMValueRef visitBreakStmt(SysYParser.BreakStmtContext ctx) {
    return LLVMBuildBr(builder, afterWhileStack.peek());
}

@Override
public LLVMValueRef visitContinueStmt(SysYParser.ContinueStmtContext ctx) {
    return LLVMBuildBr(builder, whileConditionStack.peek());
}
```

3. 实验困难

笔者实验过程中遇到了 OJ 的报错：

```
lli-13: lli: out.ir:67:3: error: instruction expected to be numbered '%3' %2 = call i32 @get_one(i32 0) ^
```

笔者分析发现 OJ 会检查以纯数字命名的变量。在代码中，只有函数调用的返回值用纯数字作为序号命名，且 OJ 还要求返回值为空类型的函数调用不应当有变量名。

笔者根据函数的返回值类型生成变量名：

```
if (retTypeMap.get(functionName).equals("void")) {
    functionName = "";
}
return LLVMBuildCall(builder, function, args, argsCount, functionName);
```