Algorithm 1: CFG building algorithm (the first version)

```
Result: Control-flow graph of a contract
function_list = all functions in a contract;
while function_list do
    remove element e from funclist;
    process-func(fun.body)
end
def process-func(body):
    node = new_node (ENTRYPOINT);
    block = get_first_block(body);
    node = process-block(block);
    return node;
def REC: process-block(block, node):
    foreach statement in block["statements"] do
        the_last_node = process-statement(statement, node)
    end
    return the_last_node;
def process-statement(statement, node):
    /* Statement = IfStatement | WhileStatement |
       ForStatement | Block | InlineAssemblyStatement | ...
       | SimpleStatement                                    */
    /* SimpleStatement = VariableDefinition |
       ExpressionStatement                                  */
    switch statement.getName() do
        case If-Statement do
            node = process-if(statement, node);
        end
        case Block do
            node = process-block(statement, node);
        end
        case DoWhile-Statement do
            node = process-dowhile(statement, node);
        end
        case Expression-Statement (normal) do
            new_node = new_node(TYPE.EXPR, statement.getExpr());
            link-node(node, new_node);
            node = new_node;
        end
        ...
        case DEFAULT do
            print ("Statement not parsed!!!");
        end
    end
    return node;
```

Algorithm 2: Each statment handler

```
def link-node(n1, n2):
    n1.add_son(n2);
    n2.add_father(n1);
def process-if(statement, node):
    condition = new_node(TYPE.IF, statement.getCondition());
    link-node(node, condition);
    truePart = process-statement(statement.getTrue(),
      condition);
    ... (handling FALSE part);
    return (truePart OR falsePart);
def process-dowhile(statement, node):
    whileStart = new_node (TYPE.STARTLOOP, statment);
    condition = new_node(TYPE.IFLOOP, statement.getCondition());
    link-node(node, condition);
    whileBody = process-statement(statement.getBody(),
      condition);
    whileEnd = new_node(TYPE.ENDLOOP, statement);
    link-node(node, whileStart);
    link-node(whileBody, condition);
    link-node(condition, whileEnd);
    return whileEnd;
```