# 单元测试代码生成模块设计文档

## 功能描述：

根据代码分析后识别到的函数信息为基础，对相关接口进行单元测试所需的通用代码进行自动化生成，包括参数、返回值、桩函数和期望结果测试。



## 设计方案

整体设计方案分为两个部分：

1. 自动化测试代码必要信息的输入流程实现。
2. 自动化测试代码生成功能实现。

### 自动化测试代码信息流输入

根据目标函数信息，可以获取到以下信息：

1. 函数名称。
2. 参数信息。
3. 判断返回值信息。
4. 调用的其他方法。

因此生成针对目标函数的自动化测试代码应该包括以上几个方面信息，当前测试框架选择为 Catch2+FakeIt 的组合，因此生成的测试代码主要也是测试框架支持的接口为主。

第一步，依据输入测试名称来生成对应的测试方法名成：

例如：

TEST\_CASE( test name [, tags ] )

SECTION( section name )

test name 与 section name 字符串格式并且要唯一，tags 同样时字符串格式它标识一个或多个用方括号标记的标签。

通过输入的 test name 与 tags 来实现对 TEST\_CASE 和 SECTION 的生成。

第二步，依据想要测试的接口参数信息生成对应的参数测试代码：

例如：

TEST\_CASE("Generators") {

auto i = GENERATE(1, 2);

auto j = GENERATE(3, 4, 5);

myFunction(i, j);

}

第三步，判断返回值数据，也就是对于我们传入的参数或者调用方法，应该正确返回的数据值也就是测试的断言部分：

例如：

REQUIRE(i.foo(1) == 1);

第四步，调用其他方法，这里主要也就是指对一些未实现的接口进行 Mock，在当前的测试框架设计中主要由两种情况：

1. 注入式：经典的 Mock 方式，使用FakeIt 可以很轻易的实现，通过生成一个 fake 接口注入到调用的目标函数中，供其调用链接，并且可以控制fake 接口的返回值以及判别调用次数等等。优势：整洁、对源码友善。缺点：必须在编写代码时针对其进行设计考量，对于老旧代码无能为力。
2. 针对老旧代码，注入式mock无法实现其功能，因此我们还需要生成对应的接口源码，类似直接通过元模板的方式来实现目标函数中调用的未实现接口，将其直接实现为简单代码，便于测试链接通过。优势：万金油型的测试代码生成，方便快捷。缺点：代码量飙赠，维护费力，编写测试需要注意的细节过多。

### 二，自动化代码生成