

# C++测试驱动开发与持续集成

使用Google Test与Travis Ci进行现代C++项目开发

信息科学技术学院

1200012778

赵睿哲

# Google Test简介

- 类似JUnit的xUnit测试框架，核心功能：

- 相等与断言
- 检测异常
- 使用宏生成完整的测试程序

```
Expected: equation_solver->solve() throws an exception.
Actual: it doesn't.
[ FAILED ] FloatEquationSolverTest.NullSolPtrTest (0 ns)
[-----] 2 tests from FloatEquationSolverTest (0 ns total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test case ran. (1 ns total)
[ PASSED ] 1 test.
[ FAILED ] 1 test, listed below:
[ FAILED ] FloatEquationSolverTest.NullSolPtrTest
```

测试报告

初始化命令行参数

```
TEST(DoubleSqrtTest, Positive) {
    EXPECT_EQ(1.0, d_sqrt(1.0));
    EXPECT_EQ(0.5, d_sqrt(0.25));
    EXPECT_EQ(50.332, d_sqrt(2533.310224));
}

TEST(DoubleSqrtTest, ZeroAndNegative) {
    EXPECT_EQ(0.0, d_sqrt(0.0));
}

int main(int argc, char *argv[]) {
    ::testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

测试声明

运行测试

# Google Test与TDD

- TDD即测试驱动开发，常见于Web应用和大型项目
- 目标应用：一元二次方程求解程序\*
  - 基本算法与项目框架确定
  - 使用Makefile搭建Google Test环境
  - 编写测试用例 - “从红变绿”

目标类EquationSolver和唯一的成员函数solve

```
template <typename T>
class EquationSolver {
/**
 * This class will gracefully solve a equation as:
 * a * x^2 + b * x + c = 0 (1)
 */
public:
/**
 * solve: Given all parameters, will create a equation and solve it
 * @ param a: a in the equation (1), default to be 0
 * @ param b: b in the equation (1), default to be 0
 * @ param c: c in the equation (1), default to be 0
 * @ param sol: Pointer to equation solutions.
 * @ return: Number of solutions, can only be 0, 1, 2.
 */
int solve(T a = 0, T b = 0, T c = 0, T **sol = NULL) {
```

kumasento add GNU compiler	
3rdparty/gtest	initialized project
bin/test	refactorized
include	refactorized
src	refactorized
.gitignore	Merge branch 'master' of github.com:kumasento/EquationSolver
.travis.yml	add GNU compiler
Makefile	fixed pthread
README.md	Update README.md

\*项目代码参见：<https://github.com/kumasento/EquationSolver>

# TDD示例

```
int solve(T a = 0, T b = 0, T c = 0, T **sol = NULL) {
    // pointer to solution storage should not be null
    // or it will throw an exception
    if (sol == NULL)
        throw "Pointer to solutions should not be null";
    return 0;
}
```

原始代码

增加了包含两个解的一元二次用例，未通过（红）

```
[ FAILED ] FloatEquationSolverTest.TwoSolutionsTest
[ RUN ] FloatEquationSolverTest.NullTest
[ OK ] FloatEquationSolverTest.NullTest
[-----] 2 tests from FloatEquationSolverTest
[-----] Global test environment tear down
[=====] 2 tests from 1 test case ran
[ PASSED ] 1 test.
[ FAILED ] 1 test, listed below:
[ FAILED ] FloatEquationSolverTest.TwoSolutionsTest
```

1 FAILED TEST

```
TEST(FloatEquationSolverTest, TwoSolutionTest) {
```

```
    float *sol;
    int num_sol;
    /**
     *  $x^2 - 1 = 0$ 
     * solutions: +1, -1
     */
    num_sol = equation_solver->solve(1, 0, -1.0, &sol);
    EXPECT_EQ(num_sol, 2);
    EXPECT_FLOAT_EQ(sol[0], 1.0);
    EXPECT_FLOAT_EQ(sol[1], -1.0);
}
```

```
/**
 *  $x^2 + 3x + 1 = 0$ 
 * solutions:
 */
num_sol = equation_solver->solve(1.0, 3.0, 1.0, &sol);
EXPECT_EQ(num_sol, 2);
EXPECT_FLOAT_EQ(sol[0], -0.381966);
EXPECT_FLOAT_EQ(sol[1], -2.618033);
}
```

```
int solve(T a = 0, T b = 0, T c = 0, T **sol = NULL) {
    // pointer to solution storage should not be null
    // or it will throw an exception
    if (sol == NULL)
```

```
        throw "Pointer to solutions should not be NULL";
```

```
    sol = 0;
```

```
    if (b * b - 4 * a * c > 0) {
        // all have two results
        sol = 2;
        T * sol_ptr = (T *) malloc(sizeof(T) * num_sol);
        if (sol_ptr == NULL)
            throw "Cannot allocate more memory";
        // no solutions
        sol_ptr[0] = (-b + sqrt(delta)) / (2 * a);
        sol_ptr[1] = (-b - sqrt(delta)) / (2 * a);
    }
}
```

```
    return num_sol;
```

修改过的代码，可以通过测试（绿）

# Google Test策略

- 例1：如何测试一个优先队列的实现？多个测试、但共享几个测试对象（队列）
- 例2：如何测试空指针异常？
- 例3：如何给测试增加参数？（value-parameterized）

```
// To use a test fixture, derive a class from testing::Test.  
class QueueTest : public testing::Test {  
protected: // You should make the members protected s.t. they can be  
            // accessed from sub-classes.
```

```
// virtual void SetUp() will be called before each test is run. You  
// should define it if you need to initialize the variables.  
// Otherwise, this can be skipped.  
virtual void SetUp() {  
    q1_.Enqueue(1);  
    q2_.Enqueue(2);  
    q2_.Enqueue(3);  
}
```

例1：使用Test Fixture创建测试上下文

```
ASSERT_THROW(Foo(5), bar_exception);
```

例2：使用ASSERT\_THROW测试异常产生


例3：对测试用例使用参数进行初始化

```
INSTANTIATE_TEST_CASE_P(  
    OnTheFlyAndPreCalculated,  
    PrimeTableTest,  
    Values(&CreateOnTheFlyPrimeTable, &CreatePreCalculatedPrimeTable<1000>));
```

# 持续集成

- 短时间内将更新代码加入主干代码并通过测试
- **Travis CI**（持续集成服务） + **GitHub**（代码托管服务）

Branch: master ▾ EquationSolver / .travis.yml

 kumasento add GNU compiler

1 contributor



12 lines (11 sloc) | 124 Bytes

```
1 language: cpp
2 compiler:
3   - clang
4   - g++
5 os:
6   - linux
7   - osx
8 script:
9   - make
10  - ./bin/test/foo_test.bin
11  - ./bin/test/bar_test.bin
```

Travis Ci配置文件，指定运行测试程序（10-11行）

git push

Build Jobs

oo #8.1	 </> Compiler: clang	no environment variables set	17 sec
oo #8.2	 </> Compiler: clang	no environment variables set	-
oo #8.3	 </> Compiler: g++	no environment variables set	18 sec
oo #8.4	 </> Compiler: g++	no environment variables set	-

测试过程中.....

结果

**EquationSolver** build passing

Google Test + Travis CI example

# 总结

- Google Test是最流行的C++测试框架之一，特点：
  - 简单地使用宏进行配置（CppUnit完全基于类，更像JUnit）
  - 好用美观的界面（命令行 or Google Test UI）
  - 流行
  - .....
- 使用TDD开发C++程序，效率高，正确率高
- 使用持续交付进行C++项目管理