Unit Testing

1. Что такое Unit Testing
2. Что тестировать?
3. Когда Тестировать?
4. Test Driven Development
5. Пример
6. Виды тестов
7. Макросы проверки
8. Вывод результатов
9. Преимущества и недостатки

Что такое Unit Testing

- Unit Testing - тестирование модулей приложения с помощью программных процедур.
- Обычно Unit Tests пишутся разработчиками, и являются первым уровнем тестирования приложения.
- Позволяют выявлять проблемы в незаконченных модулях на стадии разработки.
- Можно рассматривать как средство документирования кода.

Что тестировать?

1. Код должен быть относительно простым
2. Не иметь большого числа зависимостей
3. Обладать необходимым интерфейсом для тестирования
4. Входные и выходные данные должны быть стабильными и легко воспроизводимыми

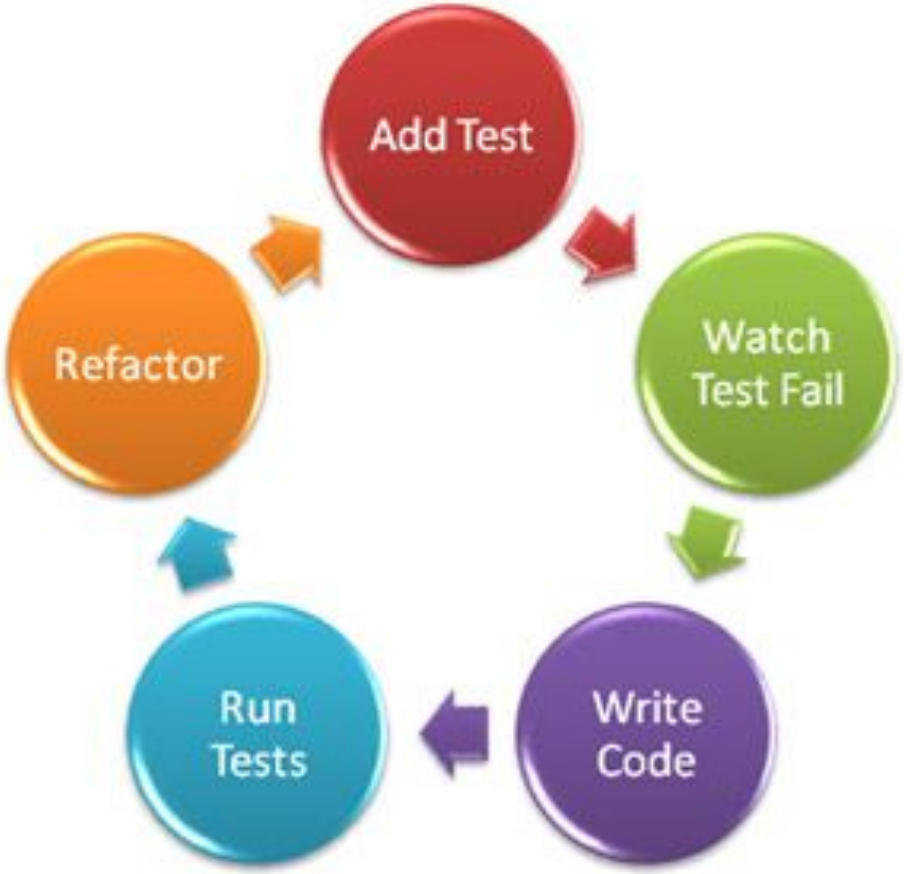
Когда тестировать?

Есть два подхода:

1. Написание Unit Tests предшествует написанию кода

2. Создание тестов после окончания реализации функциональности

Test Driven Developmet



Пример теста

```
#include "UnitTest++\UnitTest++.h"
#include "..\FeatureExtractionCore\SharedLinks.h"

#include "..\FeatureExtractionCore\BoundCalculator.h"

using namespace SightPower::XGIP::Plugins::FeatureExtractionCore;

TEST(BoundTest1)
{
	Vertices points;
	points.push_back(Vertex( 2.4,  5.6,  -2.06));
	points.push_back(Vertex( 3.67, 7.8,   0.8));
	points.push_back(Vertex(-1.5,  -2.9,  4.5));
	points.push_back(Vertex(-0.6,  -7.6, -0.12));

	Bound b;
	ComputeBound(b, points.begin(), points.end(), [&](const Vertex&)->bool{return true;});

	CHECK_EQUAL(-1.5, b.X);
	CHECK_EQUAL(-7.6, b.Y);
	CHECK_EQUAL(-2.06, b.Z);

	CHECK_EQUAL(3.67, b.FX);
	CHECK_EQUAL(7.8, b.FY);
	CHECK_EQUAL(4.5, b.FZ);
}
```

UnitTest++
Виды тестов

- Простой тест

```
TEST(YourTestName)
{
}
```

- Тест с состоянием - Fixture

```
struct SomeFixture
 {
        SomeFixture() { /* some setup */ }
         ~SomeFixture() { /* some teardown */ }
         int testData;
};
TEST_FIXTURE(SomeFixture, YourTestName)
 {
        int temp = testData;
}
```

UnitTest++
Макросы проверки

- CHECK(false);
- CHECK_EQUAL(10, 20);
- CHECK_CLOSE(3.14, 3.1415, 0.01)
- CHECK_THROW(throw TestException(), TestException);
- UNITTEST_TIME_CONSTRAINT(50);

Вывод результатов

Вывод результатов тестирования в Visual Studio:

Tests.vcxproj -> q:\XGIP\Platform\SightPower.XGIP.Plugins.FeatureExtractionTests.exe
EigenTest.cpp(63): error : Failure in EigenTest: Expected 0.5 +/- 1e-005 but was 0.423481
PlaneTest.cpp(38): error : Failure in PlaneTest3: Expected 78.3 +/- 0.0001 but was 4.3
FAILURE: 2 out of 11 tests failed (2 failures).
Test time: 0.66 seconds.

Преимущества использования Unit Testing

1. Выявление проблем на стадии разработки
2. Создание атомарного малосвязанного кода
3. Выявление проблем связанных с зависимостями
4. Автоматический Regression Testing

Недостатки
Unit Testing

1. Дополнительное время на разработку
2. Дополнительное время на компиляцию
3. Сложность применения при активно изменяющемся коде
4. Недостаточная интеграция в среду разработки