Для тестирования выполнения исходного кода используется Unit-тестирование. Поскольку разработка проекта ведется на С++ с использованием среды разработки Qt, в качестве инструмента для автоматизации тестирования выбран фреймворк QTestLib.

Для модульного тестирования разработаны следующие тест-кейсы:

1. Тестирование DiGrpah (реализация интерфейса AbstractDigraph)
   1. Добавление новой связи между двумя вершинами и проверка их связанности
   2. Добавление связей между несколькими вершинами (по типу «Звезда») и проверка смежности
   3. Добавление последовательной связи между 3 вершинами и проверка достижения
   4. Добавление связей между несколькими вершинами и проверка степени входящих связей для отдельно выбранной вершины
   5. Добавление связей между несколькими вершинами и проверка степени исходящих связей для отдельно выбранной вершины
   6. Добавление связей между несколькими вершинами и проверка последовательности их вывода итератором
   7. Загрузка информации о вершинах графа и проверка создания нового графа на основании данных потока
   8. Загрузка информации о вершинах графа в поток и создание нового графа на основании данных потока, проверка наличия и связей между вершинами
2. Тестирование EulerianCircuit

Данный класс проверяет является ли граф Эйлеровым, а также отдает последовательность вершин, входящих в Эйлеров цикл, если такой имеется.

* 1. Загрузка данных Эйлерова графа, проверка его на пригодность
  2. Загрузка данных не Эйлерова графа, проверка непригодности
  3. Загрузка данных Эйлерова графа, проверка вывода вершин Эйлерова цикла

Архитектуру модуля Unit-тестирования можно выразить через диаграмму классов, которая представлена ниже:

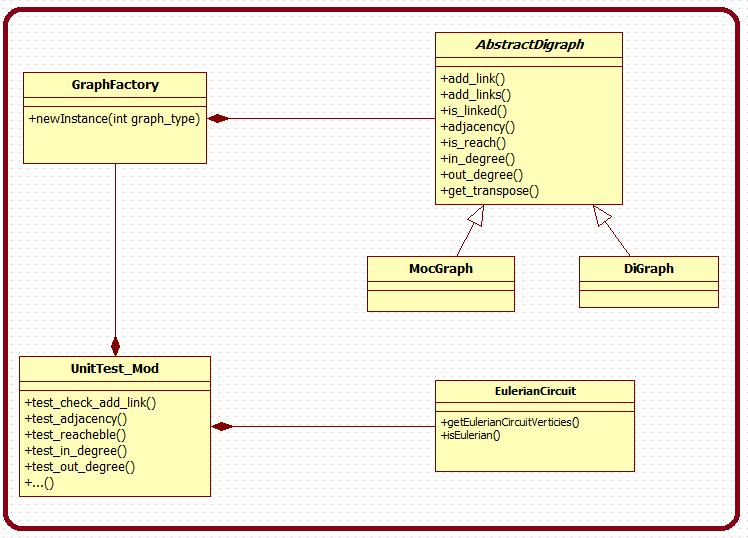
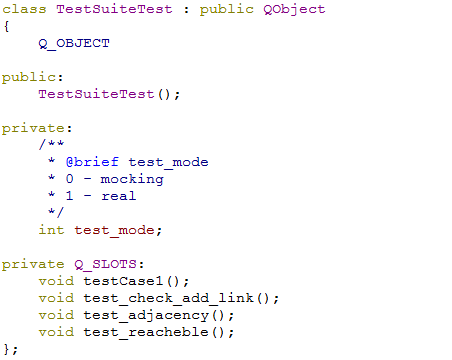
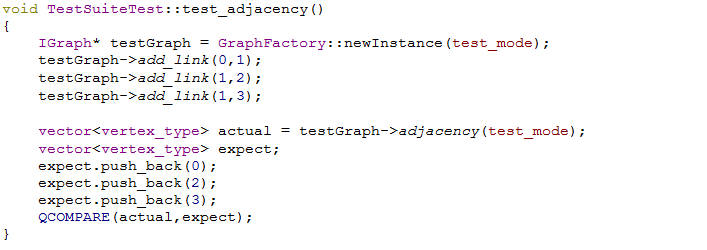


Рис. 1 Диаграмма классов модуля Unit-тестирования

Запуск тестов происходит через вызов макроса QTEST\_APPLESS\_MAIN(TestSuiteTest) на этапе сборки проекта, который по указанному имени класса находит модульные тесты. Согласно TDD разработка тестов велась с применением заглушек (mocks). В классе TestSuiteTest имеется установка режима тестирования, которые отвечает за выбор объектов для проверки (реальные объекты или заглушки):



Объекты создаются через фабрику (GraphFactory). Пример кода теста, отвечающего за проверку смежности вершин:



Результаты тестирования:

\*\*\*\*\*\*\*\*\* Start testing of TestSuiteTest \*\*\*\*\*\*\*\*\*

Config: Using QtTest library 5.5.1, Qt 5.5.1 (i386-little\_endian-ilp32 shared (dynamic) debug build; by GCC 4.9.2)

PASS : TestSuiteTest::initTestCase()

PASS : TestSuiteTest::testCase1()

PASS : TestSuiteTest::test\_check\_add\_link()

PASS : TestSuiteTest::test\_adjacency()

PASS : TestSuiteTest::test\_reacheble()

PASS : TestSuiteTest::test\_in\_degree()

PASS : TestSuiteTest::test\_out\_degree()

PASS : TestSuiteTest::test\_int\_vertex()

PASS : TestSuiteTest::test\_transpose()

PASS : TestSuiteTest::graphTestLinkage()

PASS : TestSuiteTest::graphTestAdjacency()

PASS : TestSuiteTest::graphTestReachability()

PASS : TestSuiteTest::graphTestForeach()

PASS : TestSuiteTest::graphShoultBeCreated()

PASS : TestSuiteTest::graphShouldCheckExistingElementsRight()

PASS : TestSuiteTest::generatorTest()

PASS : TestSuiteTest::eulerianCircleTest()

PASS : TestSuiteTest::test\_is\_eulerian()

PASS : TestSuiteTest::test\_is\_not\_eulerian()

PASS : TestSuiteTest::cleanupTestCase()

Totals: 20 passed, 0 failed, 0 skipped, 0 blacklisted

\*\*\*\*\*\*\*\*\* Finished testing of TestSuiteTest \*\*\*\*\*\*\*\*\*