Лабораторная работа №7. Автоматизированное тестирование.

1. Создать проект автоматизированного тестирования с использованием библиотеки QTestLib.

1.1. Открыть мастер создания проекта "Проект автотестирования". Для этого нажать кнопку "Новый проект" на вкладке "Начало", и выбрать Другие проекты/Проект автотестирования.

1.2. В Качестве среды тестирования указать QtTest. Установить отметку "Создать код инициализации и очистки". Отметки "Приложение с GUI" и "Требуется QApplication" не ставить.

1.3. В качестве имени теста ввести test_sample.

1.4. Завершить работу мастера.

2. Добавить сценарии тестов в проект тестирования.

2.1. Добавить в объявление класса поля `x` типа `int` и `s` типа `QString`.

2.2. Добавить в реализацию метода `initTestCase()` код:

```
x = 10;
s = "control value";
```

2.2. Переименовать функцию `test_case1` в `test_succeed`. Для этого щелкнуть правой кнопкой мыши по строке `test_case1` в любом месте исходного кода, и в контекстном меню выполнить команду

Рефакторинг/Переименовать. В открывшейся панели ввести новое имя функции.

2.3. В объявление класса добавить объявление слота `test_fail()`.

2.4. Добавить реализацию функции `test_fail()`. Для этого щелкнуть правой кнопкой по имени функции, и в контекстном меню выбрать команду Рефакторинг/Добавить реализацию вне класса.

2.5. В реализацию функции `test_fail()` добавить код:

```
QFAIL("Test failed");
```

2.6. Подключить в модуль заголовок библиотеки `QString`. Добавить слот `test_compare_pass()` и его реализацию, как в пп. 2.3-2.4. В реализацию вставить код:

```
QString v = "control value";
QCOMPARE(v, s);
```

2.7. Добавить слот `test_compare_fail()` и его реализацию. В реализацию вставить код:

```
QString v = "Control value";
QCOMPARE(v, s);
```

2.8. Добавить слот `test_verify()` и его реализацию. В реализацию вставить код:

```
QVERIFY(x > 0);
```

2.9. Добавить заголовки стандартной библиотеки chrono и thread. Добавить слот `test_benchmark()` и его реализацию. В реализацию вставить код:

```
QBENCHMARK {
std::this_thread::sleep_for(std::chrono::milliseconds(100
));
}
```

3. Выполнить тестирование.

3.1. Переключить проект в режим сборки "Выпуск". И запустить его.

3.2. Убедиться, что в панели "Вывод приложения" появилась вкладка с именем проекта, в которой находятся результаты тестирования.

3.3. Убедиться, что тесты test_fail() и test_compare_fail() завершились неудачей, а все остальные — успехом.

3.4. Сравнить результат выполнения теста test_benchmark() с заданным значением задержки.

4. Управление ходом тестирования.

4.1. Переключиться в инспектор тестов, выбрав соответствующий пункт в выпадающем списке (над инспектором проекта).

4.2. Раскрыть пункт QtTest и убедиться, что все созданные тесты присутствуют в списке.

4.3. Отключить те тесты, которые закончились неудачей, убрав соответствующие отметки.

4.3. Переключиться в панель вывода "Результаты тестирования" (внизу).

4.4. Запустить тестирование из этой панели, нажав на кнопку запуска в верхней части. Убедиться, что тесты функционируют согласно ожиданиям, и серия тестов заканчивается неудачей.

4.5. Запустить тестирование выбранных тестов, нажав на кнопку избирательного выполнения в верхней части панели. Убедиться, что серия тестов заканчивается успехом.

5. Модульное тестирование функции списка простых чисел.

5.1. Создать проект автоматизированного тестирования, как в п. 1.

5.2. Добавить в проект файлы модуля библиотеки, содержащей тестируемую функцию.

5.3. Добавить в файл главного модуля проекта ссылку на файл заголовка библиотеки (с учетом пути).

5.4. Создать тест, проверяющий отсутствие единицы в списке простых чисел. Для этого в реализацию тестовой функции вставить код (`list` - имя функции из библиотеки):

```
QCOMPARE(list(1, 2), std::vector<int>({2}));
```

5.5. Создать тесты, проверяющие включение границ диапазона в список чисел (отдельно для верхней и нижней границ).

5.6. Создать тесты, проверяющие работу механизма обработки ошибок, используя конструкцию `try...catch` (для всех возможных сценариев, по одному тесту на каждый случай).

5.7. Добавить тест производительности, измеряющий время нахождения всех пятизначных простых чисел.

5.8. Выполнить тестирование.

5.9. Исправить ошибки в реализации функции (не тесты!), чтобы тестирование заканчивалось успехом.