Лабораторная работа №8. Тестирование графического интерфейса.

1. Создать проект для тестирования.

1.1. Открыть мастер создания проекта "Проект автотестирования" со следующими настройками: среда тестирования — QtTest, приложение с GUI, создать код инициализации и очистки, требуется QApplication – установлены.

1.2. Удалить из исходного кода главного модуля проекта объявление и реализацию метода `test_case1`.

1.3. Переключить проект в режим сборки "Выпуск".

1.4. Добавить в проект параметры для развертывания приложения:

```
DEPLOY_TARGET = $$shell_quote($$shell_path($$OUT_PWD/release/$
$TARGET.exe))
QMAKE_POST_LINK = windeployqt $$DEPLOY_TARGET
```

1.5. Добавить в проект файл конфигурации стилизатора.

1.6. Поместить проект в систему управления версиями.

2. Добавить в проект форму графического приложения.

2.1. Добавить в проект ссылки на файлы заголовка, реализации и описания модуля формы из проекта графического приложения (`dialog.h`, `dialog.cpp`, и `dialog.ui`).

2.2. Добавить в реализацию главного модуля проекта автотестирования ссылку на файл заголовка модуля формы (с учетом пути).

2.3. Добавить в объявление класса теста указатели на следующие графические компоненты: саму форму, поле для ввода значения A, поле для ввода значения B, кнопку запуска вычислений, компонент для вывода списка значений, например:

```
private:
Dialog * mainDialog;
QLineEdit * lineEditA;
QLineEdit * lineEditB;
QPushButton * pushButton;
QListWidget * listWidget;
```

2.4. Подключить заголовки необходимых классов (в вышеприведенном примере — `QLineEdit`, `QPushButton`, `QListWidget`).

2.5. Добавить в реализацию функции `initTestCase` создание формы и получение указателей на ее компоненты:

```
mainDialog = new Dialog();
lineEditA = mainDialog->findChild<QLineEdit*>("lineEditA");
lineEditB = mainDialog->findChild<QLineEdit*>("lineEditB");
pushButton = mainDialog->findChild<QPushButton*>("pushButton");
listWidget = mainDialog->findChild<QListWidget*>("listWidget");
mainDialog->show();
```

2.6. Добавить в реализацию функции `cleanupTestCase` уничтожение формы:

```
delete mainDialog;
```

2.7. Выполнить сборку и запуск проекта из панели "Результаты тестирования" и убедиться, что приложение функционирует.

2.8. Зафиксировать изменения в системе управления версиями.

3. Тестирование компонентов интерфейса.

3.1. Добавить тестовый сценарий ввода данных в поле для значения A со следующей реализацией:

```
// имитируем событие щелчка левой кнопкой мыши на поле ввода A
QTest::mouseClick(lineEditA, Qt::LeftButton);
// имитируем нажатие последовательности клавиш 123
QTest::keyClicks(lineEditA, "123");
// сравниваем введенное значение с образцом
QCOMPARE(lineEditA->text(), QString("123"));
// очищаем поле ввода
lineEditA->clear();
```

3.2. Добавить аналогичный тестовый сценарий для поля ввода значения B.

3.3. Добавить сценарий проверки независимой работы кнопки с использованием управления сигналами Qt:

```
// отсоединяем слот обработки данных
QObject::disconnect(pushButton, SIGNAL(clicked()), mainDialog,
SLOT(on_pushButton_clicked()));
// создаем объект инспектора сигналов, реагирующего на сигнал
clicked() кнопки
QSignalSpy clickSpy(pushButton, SIGNAL(clicked()));
// имитируем событие нажатия кнопки
```

```
QTest::mouseClick(pushButton, Qt::LeftButton);
// проверяем, что инспектор зафиксировал одно событие
QCOMPARE(clickSpy.count(), 1);
// присоединяем слот обработки данных обратно
QObject::connect(pushButton, SIGNAL(clicked()), mainDialog,
SLOT(on_pushButton_clicked()));
```

3.4. Добавить сценарий проверки последовательности перехода фокуса ввода по компонентам формы:

```
QTest::mouseClick(lineEditA, Qt::LeftButton);
QVERIFY(lineEditA->hasFocus());
QTest::keyClick(lineEditA, Qt::Key::Key_Tab);
QVERIFY(lineEditB->hasFocus());
QTest::keyClick(lineEditA, Qt::Key::Key_Tab);
QVERIFY(pushButton->hasFocus());
QTest::keyClick(pushButton, Qt::Key::Key_Tab);
QVERIFY(listWidget->hasFocus());
QTest::keyClick(pushButton, Qt::Key::Key_Tab);
QVERIFY(lineEditA->hasFocus());
```

3.5. Выполнить тестирование и, при необходимости, внести изменения в модуль формы.

3.6. Зафиксировать изменения в проекте тестирования и проекте графического приложения.

4. Интеграционное тестирование.

4.1. Добавить сценарий проверки очистки списка вывода перед заполнением, используя непосредственное взаимодействие с компонентами:

```
// устанавливаем текст в полях ввода A и B
```

```
lineEditA->setText("10");
lineEditB->setText("20");
// генерируем событие нажатия кнопки
pushButton->click();
// проверяем количество значений в списке
QCOMPARE(listWidget->count(), 4);
// повторно нажимаем кнопку
pushButton->click();
// и проверяем количество значений
QCOMPARE(listWidget->count(), 4);
```

4.2. Добавить сценарий проверки полного цикла взаимодействия с приложением:

```
QTest::keyClicks(mainDialog->focusWidget(), "10");
QTest::keyClick(mainDialog, Qt::Key::Key_Tab);
QTest::keyClicks(mainDialog->focusWidget(), "20");
QTest::mouseClick(pushButton, Qt::LeftButton);
QCOMPARE(listWidget->count(), 4);
std::vector<QString> result;
for(int i = 0; i < listWidget->count(); ++i) {
    result.push_back(listWidget->item(i)->text());
}
QCOMPARE(result, std::vector<QString>({QString("11"),
QString("13"), QString("17"), QString("19")}));
```

4.3. Добавить сценарий проверки появления окна с сообщением об ошибке при недопустимом значении A:

```
lineEditA->setText("");
lineEditB->setText("20");
seenMessage = false;
// через 100 мс выполняем функцию close_messagebox()
QTimer::singleShot(100, this, &test_primelist::close_messagebox);
```

```
pushButton->click();
if(!seenMessage) {
    QFAIL("no messagebox");
}
QCOMPARE(messageText, "a must be a number");
```

4.4. Добавить необходимые для работы предыдущего сценария поля в класс теста:

```
bool seenMessage;
QString messageText;
```

4.5. Добавить объявление и реализацию функции `close_messagebox()` в класс теста (функция не должна быть слотом!):

```
void test_primelist::close_messagebox() {
    QMessageBox * msgbox =
qobject_cast<QMessageBox*>(QApplication::activeModalWidget());
    if(msgbox) {
        seenMessage = true;
        messageText = msgbox->text();
        msgbox->close();
    }
}
```

4.6. Выполнить тестирование и, при необходимости, внести изменения в модуль формы.

4.7. Зафиксировать изменения в проекте тестирования и проекте графического приложения.

5. Добавить интеграционные тесты для всех сообщений об ошибках. по аналогии с пп. 4.3-4.7.