Лабораторная работа №2

Создание графического приложения с использованием фреймворка Qt5 в среде QtCreator.

1. Создать проект на основе шаблона приложения Qt Widgets.

1.1. В мастере указать в качестве базового класса QDialog.

1.2. Переключить сборку в режим Выпуск. Для этого щелкнуть левой кнопкой мыши на кнопке выбора режима сборки и в появившемся списке выбрать нужный режим.

1.3. Дополнить проект конфигурацией развертывания библиотек Qt для обеспечения возможности автономного запуска исполняемого модуля приложения. Для этого открыть файл конфигурации проекта (расширение pro) и добавить следующий код:

```
CONFIG(release, debug|release) { BUILDTYPE = release }
CONFIG(debug, debug|release) { BUILDTYPE = debug }
QMAKE_POST_LINK = windeployqt $$shell_quote($${OUT_PWD}/$${BUILDTYPE}/$${TARGET}.exe)
```

2. Создать интерфейс пользователя с помощью специализированного редактора.

2.1. Открыть редактор графического интерфейса пользователя двойным щелчком на файле описания интерфейса (расширение ui).

2.2. Добавить блок для размещения полей ввода исходных данных. Для этого на панели инструментов найти компонент Horizontal Layout и перетащить его на область окна. В инспекторе объектов в правой части окна среды разработки должен появиться объект типа QHorizontalLayout.

2.3. Установить схему размещения "По вертикали" для основного окна приложения. Для этого вызвать локальное меню щелчком правой кнопки мыши на области окна в редакторе интерфейса, и выбрать действие Компоновка->Скомпоновать по вертикали. Если этот пункт недоступен, сначала выбрать Компоновка->Удалить компоновщик.

2.4. Добавить в блок horizontalLayout поля для ввода данных и подписи к ним. Для этого на панели инструментов найти компоненты Line Edit и Label, и добавить по 2 штуки каждого типа перетащив их в область компонента horizontalLayout. В инспекторе объектов они должны отобразиться как дочерние для него.

2.5. Разместить компоненты в правильном порядке, в соответствии с необходимым видом интерфейса (см. рисунок).

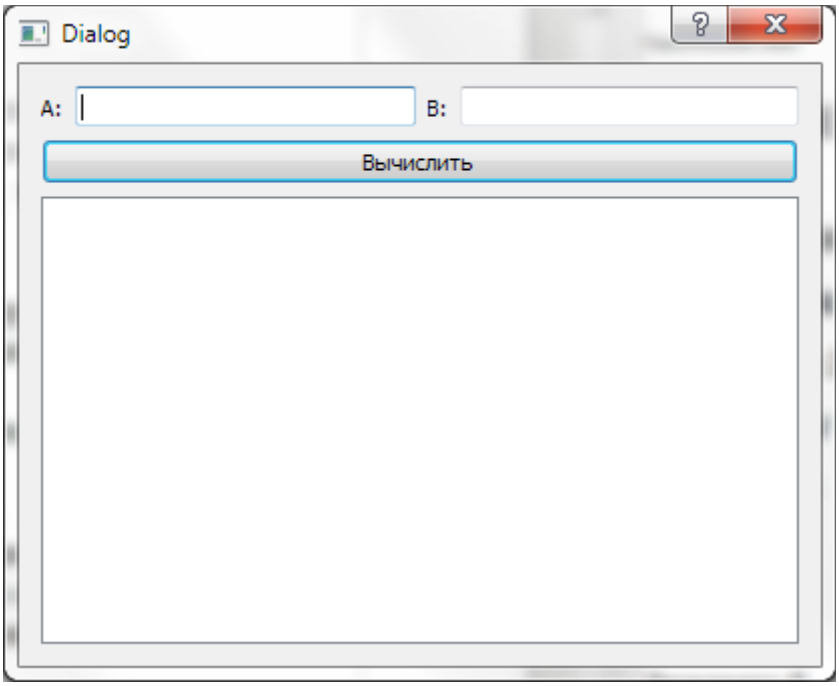
2.6. Переименовать компоненты, назвав объекты типа QLabel, соответственно, labelA и labelB, а объекты типа QLineEdit - lineEditA и lineEditB.

2.7. В редакторе свойств изменить значения свойства text для labelA и labelB на "A:" и "B:" соответственно.

2.8. Добавить кнопку для запуска процесса вычислений. Для этого в панели инструментов найти компонент Push Button и перетащить его на область окна. В инспекторе объектов должен появиться объект типа QPushButton, являющийся дочерним для объекта главного окна.

2.9. Изменить свойство text кнопки, задав значение "Вычислить" в редакторе свойств.

2.10. Добавить компонент списка для вывода полученных значений. Для этого найти на панели инструментов компонент List Widget (не List View!) и перетащить его на область окна. В инспекторе объектов должен появиться объект типа QListWidget. Готовый интерфейс должен выглядет так, как показано на рисунке.



3. Задать реакцию на событие нажатия кнопки.

3.1. Открыть редактор сигналов и слотов главного окна. Для этого вызвать локальное меню объекта главного окна (в редакторе или инспекторе объектов), и выбрать команду Изменить сигналы/слоты...

3.2. Добавить слот calculate() в список слотов. Для этого нажать кнопку добавления слота и ввести описание слота в появившемся поле ввода.

3.3. Переключиться в редактор сигналов и слотов, выбрав соответствующую вкладку в интерфейсе. Если такой вкладки нет, включить ее командой Окно->Обзоры->Редактор сигналов и слотов.

3.4. Добавить связь сигнала clicked() со слотом calculate(). Для этого нажать кнопку добавления связи в редакторе, и в появившейся строке указать следующие значения:

отправитель - имя объекта кнопки (pushButton)
сигнал - clicked()
получатель - имя объекта главного окна (Dialog)
слот - calculate()

3.5. Сохранить сделанные изменения.

4. Добавить объявление и реализацию слота calculate() в класс главного окна.

4.1. В заголовке модуля главного окна в объявление класса главного окна добавить раздел public slots. В него - объявление метода void calculate(). Текст должен выглядеть так:

```
public slots:
    void calculate();
```

4.2. В файле реализации модуля главного окна добавить реализацию метода calculate, например (имя класса должно совпадать с именем класса главного окна):

```
void Dialog::calculate() {

}
```

5. Проверить работу механизма взаимодействия.

5.1. В файл реализации модуля главного окна подключить библиотеку класса вывода отладочных сообщений в консоль:

```
#include <QDebug>
```

5.2. Для проверки работоспособности решения добавить в реализацию метода calculate() вывод отладочного сообщения:

```
qDebug() << "Function called: " << Q_FUNC_INFO;
```

5.3. Произвести сборку и запуск проекта. При нажатии на кнопку в окне приложения в окне "Вывод приложения" среды разработки должна появляться информация о вызванной функции, например:

```
Function called: void Dialog::calculate()
```

5.3. Завершить выполнение приложения, нажав кнопку закрытия окна.

5.4. Удалить команду вывода отладочного сообщения.

6. Добавить в проект модуль получения списка простых чисел.

6.1. В локальном меню проекта в инспекторе проектов выбрать пункт "Добавить существующие файлы"

6.2. В появившемся диалоговом окне выбрать файлы заголовка и реализации модуля и нажать кнопку "Открыть".

6.3. Проконтролировать, что в инспекторе проектов появились соотвествующие файлы, и что они также присутствуют в файле конфигурации проекта в значениях параметров SOURCES и HEADERS.

7. Реализовать ввод исходных данных и вывод результатов расчета.

7.1. Подключить в файл реализации модуля главного окна заголовок модуля получения списка простых чисел, используя относительный путь, который указан в файле конфигурации проекта.

7.2. Добавить в метод calculate() команды чтения значений из полей ввода левой и правой границ диапазона:

```
int a = ui->lineEditA->text().toInt();
int b = ui->lineEditB->text().toInt();
```

Здесь:

ui - указатель на автогенерируемый объект главного окна, создаваемый при сборке на основе описания интерфейса
lineEditA и lineEditB - указатели на объекты полей ввода значений.
text() - метод объекта класса QLineEdit, позволяющий получить его содержимое в виде объекта QString
toInt() - метод объекта класса QString, позволяющий получить целое число по его строковому представлению

7.3. Вызвать функцию вычисления списка и последовательно добавить результаты как элементы в объект QListWidget (здесь предполагается, что функция называется primelist):

```
for(int n: primelist(a, b)) {
    ui->listWidget->addItem(QString::number(n));
}
```

7.4. Выполнить сборку и запуск проекта. Проверить правильность работы. При необходимости - выполнить отладку, используя отладчик среды или механизм отладочных сообщений.