Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Севастопольский государственный университет»

ИССЛЕДОВАНИЕ СПОСОБОВ ИСПОЛЬЗОВАНИЯ СИГНАЛОВ И СЛОТОВ В QT-ПРИЛОЖЕНИЯХ

Методические указания

к лабораторной работе по дисциплине

«Кроссплатформенное программирование»

для студентов, обучающихся по направлению

09.03.02 “Информационные системы и технологии”

очной и заочной форм обучения

Севастополь
2018

УДК 004.415.2
Исследование способов использования сигналов и слотов в Qt-приложениях. Методические указания/Сост. Строганов В.А. – Севастополь: Изд-во СевГУ, 2018.–14 с.

Методические указания предназначены для оказания помощи студентам при выполнении лабораторных работ по дисциплине «Кроссплатформенное программирование».

Методические указания составлены в соответствии с требованиями программы дисциплины «Кроссплатформенное программирование» для студентов направления 09.03.02 и утверждены на заседании кафедры «Информационные системы»,
протокол № от « »_____ 2018 г.

Содержание

1. Цель работы 4
2. Основные теоретические положения 4
2.1. Сигналы 5
2.2. Слоты 5
2.3. Соединение сигналов и слотов 6
2.4. Разъединение сигналов и слотов 7
2.5. Предопределенные и собственные сигналы и слоты 8
2.6. Пример разработки приложения с использованием сигналов и слотов 8
3. Порядок выполнения лабораторной работы 12
4. Содержание отчета 13
5. Контрольные вопросы 13
Библиографический список 14

1. ЦЕЛЬ РАБОТЫ

Исследовать принцип работы механизма сигналов и слотов фреймворка Qt. Приобрести практические навыки применения сигналов и слотов при разработке Qt-приложений.

2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Сигналы и слоты используются для коммуникации между объектами. Механизм сигналов и слотов – это главная особенность Qt и, вероятно, та часть, которая отличается от возможностей, предоставляемых другими фреймворками.

В программировании графического интерфейса, когда мы меняем один виджет, мы часто хотим что бы другой виджет получил об этом уведомление. В общем случае, мы хотим что бы объекты любого типа могла общаться с другими. Например, если пользователь нажимает кнопку “Закрыть”, мы вероятно хотим что бы была вызвана функция окна close().

Другие библиотеки добиваются такого рода общения используя обратный вызов. Обратный вызов – это указатель на функцию. Таким образом, если мы хотим что бы функция уведомила нас о каких-нибудь событиях, мы передаем указатель на другую функцию (обратно вызываемую) этой функции. Функция в таком случае делает обратный вызов, когда необходимо. Обратный вызов имеет два основных недостатка. Во-первых, он не является типо-безопасным: мы никогда не можем быть уверены что функция делает обратный вызов с корректными аргументами. Во-вторых, обратный вызов жестко связан с вызывающей его функцией, так как эта функция должна точно знать какой обратный вызов надо делать.

Сигналы и слоты – это фундаментальный механизм Qt, позволяющий связывать объекты друг с другом. Связанным объектам нет необходимости что-либо "знать" друг о друге. Сигналы и слоты гораздо удобнее механизма функций обратного вызова (callbacks) и четко вписываются в концепцию ООП.

Для использования этого механизма объявление класса должно содержать специальный макрос Q_OBJECT на следующей строке после ключевого слова class:

```
class MyClass {
Q_OBJECT

public:
...

};
```

После макроса Q_OBJECT не нужно ставить точку с запятой. Перед выполнением компиляции Meta Object Compiler (MOC) анализирует такие классы и автоматически внедряет в них всю необходимую информацию (используются отдельные файлы, разработчик может их игнорировать).

Наследники QObject могут иметь любое количество сигналов и слотов.

2.1. Сигналы

Сигнал может быть определен следующим образом:

```
class MyClass: public QObject {
Q_OBJECT

public:
...

signals:
   void mySignal();

};
```

Для того чтобы инициировать сигнал (выслать сигнал) нужно использовать ключевое слово `emit`:

```
void MyClass ::sendMySignal()
{
   emit mySignal();
}
```

Сигналы могут использовать параметры для передачи дополнительной информации.

2.2. Слоты

Слоты практически идентичны обычным членам-методам C++, при их объявлении можно использовать стандартные спецификаторы доступа public, protected или private. Слоты можно вызывать напрямую, как обычные члены-функции класса C++. Главная особенность слотов – это возможность связывания с сигналами, в этом случае слоты будут вызваться автоматически при каждом возникновении соответствующих сигналов. В слотах нельзя использовать параметры по умолчанию.

Слот может быть определен следующим образом:

```
class MyClass: public QObject {
Q_OBJECT

public slots:
   void mySlot()
   {
      ...
   }
};
```

В теле слота можно узнать, какой объект выслал сигнал.

2.3. Соединение сигналов и слотов

Для соединения сигналов и слотов можно использовать статический метод connect, определенный в классе QObject. В общем виде соединение выглядит следующим образом:

```
connect(sender, SIGNAL(signal), receiver, SLOT(slot));
```

sender и receiver – это указатели на Qobject,
signal и slot – сигнатуры сигнала и слота.

Пример соединения:

```
QObject::connect(spinBox,
SIGNAL(valueChanged(int)),slider, SLOT(setValue(int)));
```

В приведенном выше примере, сигнал, возникающий при каждом изменении объекта spinbox, связывается с соответствующим слотом объекта slider. Вызов слота slider.setValue(int) происходит автоматически при каждом возникновении сигнала spinBox.valueChanged(int).

Существует множество различных вариантов соединения сигналов и слотов:

- один сигнал может быть соединен со многими слотами:

```
connect(slider, SIGNAL(valueChanged(int)),spinBox,
SLOT(setValue(int)));
connect(slider, SIGNAL(valueChanged(int)),this,
SLOT(updateStatusBarIndicator(int)));
```

При возникновении сигнала, слоты вызываются один за другим, порядок не определен.

- множество сигналов может быть соединено с единственным слотом:

```
connect(sender0, SIGNAL(overflow()),receiver1,
SLOT(handleMathError()));
connect(sender1, SIGNAL(divisionByZero()),receiver1,
SLOT(handleMathError()));
```

- сигналы могут быть соединены между собой:

```
connect(sender1, SIGNAL(function1()),receiver,
SIGNAL(function2()));
```

В этом случае при возникновении первого сигнала, автоматически генерируются все связанные сигналы. Не считая этого, соединение сигнал-сигнал неотличимо от соединения сигнал-слот.

Соединение сигналов и слотов осуществляется:

- в классах наследниках от Qobject функцией connect();
- в прочих местах программы с помощью статической функции-члена Qobject::connect().

Аргументами этой функции являются:

- указатель на объект-отправитель,
- сигнал объекта-отправителя,
- указатель на объект-получатель,
- слот объекта-получателя,
- тип соединения.

При соединении сигналов и слотов можно передавать параметры от сигнала к слоту, если существует соответствующая пара сигнал/слот:

- без параметров:

```
QPushButton *btn_quit = new QPushButton("Quit",wgt);
...
QObject::connect(btn_quit, SIGNAL(clicked()), qApp, SLOT(quit()));
```

В этом примере сигнал clicked() от кнопки btn_quit соеденяется со слотом quit() приложения (qApp - глобальный указатель на экземпляр приложения).

- с параметрами. При соединении сигналов и слотов с передачей параметров следует помнить замечание указанное в описании к QObject::connect(). То есть имена аргументов должны быть опущены.

Неправильно:

```
QObject::connect(scrollBar, SIGNAL(valueChanged(int value)),
label, SLOT(setNum(int value)));
```

Правильно:

```
QObject::connect(scrollBar, SIGNAL(valueChanged(int)), label,
SLOT(setNum(int)));
```

В этом примере сигнал valueChanged() от линейки прокрутки scrollBar соединяется со слотом setNum() текстовой метки label. Сигнал передает параметр типа int – текущая позиция ползунка. Слот метки имеет такой же тип входного аргумента (сигнатуры сигнала и слота должны всегда совпадать). После соединения этой пары сигнал/слот, при изменении положения ползунка, в текстовой метке будет отображаться текущая позиция ползунка.

2.4. Разъединение сигналов и слотов

Метод disconnect можно использовать для того, чтобы удалить соединение между сигналом и слотом:

```
disconnect(sender0, SIGNAL(overflow()),receiver1,
SLOT(handleMathError()));
```

На практике прямой вызов disconnect используется редко, так как Qt автоматически удаляет все соединения при удалении объектов.

2.5. Предопределенные и собственные сигналы и слоты

В примерах выше был описан механизм определения собственных сигналов и слотов. Стандартные классы Qt содержат множество предопределенных сигналов и слотов, описание которых можно найти в официальном руководстве Qt (клавиша F1 при курсоре на имени класса – быстрый вызов справки). Чтобы добавить собственное поведение в стандартные элементы интерфейса Qt необходимо унаследовать класс, который мы хотим расширить и добавить необходимые сигналы и слоты.

2.6. Пример разработки приложения с использованием сигналов и слотов

Необходимо создать форму для добавления слов из строчного поля в многострочное. При этом слово «помидор» - нецензурное слово. При добавлении такого слова, накапливаются штрафные баллы. При количестве баллов равному пяти – запретить некультурному пользователю ввод, спрятав кнопку добавления.

1. Создаем новый проект Qt GUI.
2. Нажимаем правой кнопкой на проект и выбираем «Добавить новый…»
3. Добавляем новый класс MyLabel, который наследуем от класса QLabel. Этот класс будет хранить и выводить количество штрафных баллов. При превышении штрафных баллов будет испускаться сигнал. Класс MyLabel представляет из себя следующее:

MyLabel.h:

```
#ifndef MYLABEL_H
#define MYLABEL_H

#include <QLabel>

class MyLabel : public QLabel
{
    Q_OBJECT
private:
    int value = 0;           //текущее значение баллов
    const int banLimit = 5; //максимальное количество штрафных
баллов,
                             //не приводящее к запрету добавления
public:
    MyLabel(QWidget *parent = 0);
public slots:
    void incValue();         //определим слот, позволяющий
увеличить количество штрафа
signals:
    void banUser();          //сигнал, испускаемый при превышении
макс. допустимого количества штрафа
```

```
};

      #endif // MYLABEL_H
```

MyLabel.cpp:

```
#include "mylabel.h"

MyLabel::MyLabel(QWidget *parent)
    : QLabel(parent)
{
    /*...*/

}

void MyLabel::incValue()
{
    if (++value > banLimit)
    {
        emit banUser();
    }
    this->setText(QString::number(value));
}
```

4. Создадим форму и добавим на неё: QLineEdit, QPushButton, QTextEdit, и две QLabel.

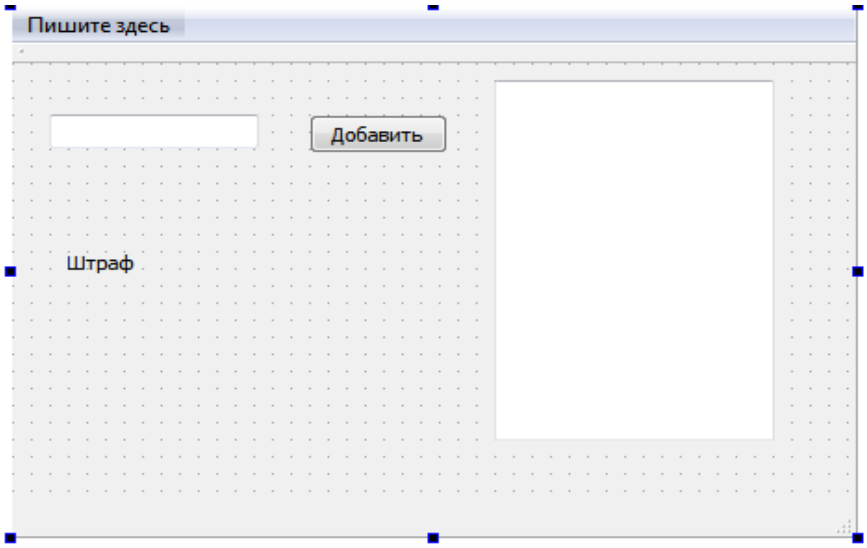

Рисунок 2.1 – Внешний вид формы приложения

5. Запускаем qmake, для того чтобы Qt корректно мог работать с нашим MyLabel, переопределяющим QLabel.

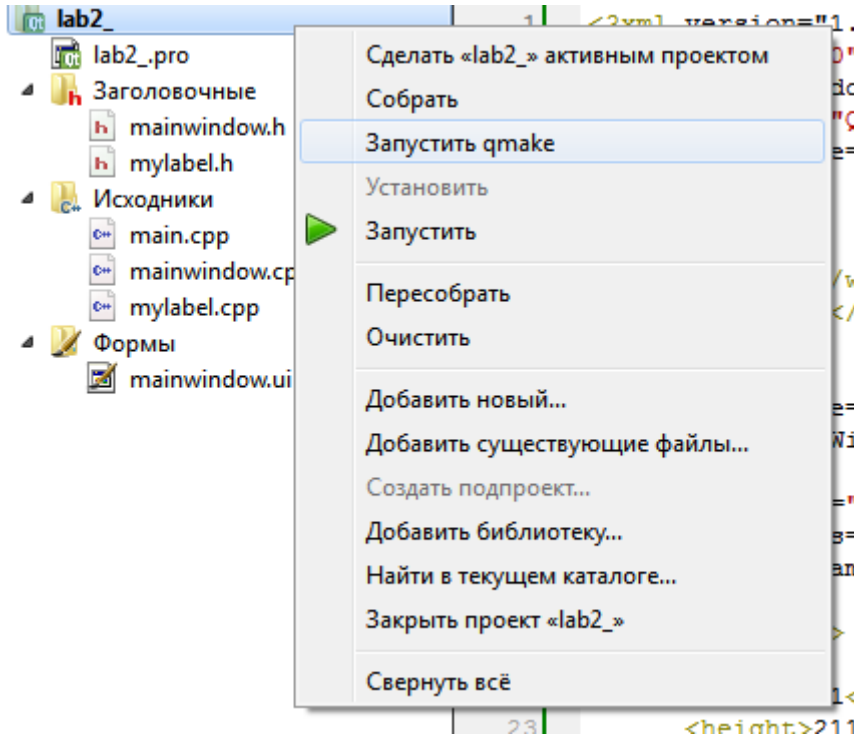

Рисунок 2.2 – Порядок запуска qmake

6. Открываем файл разметки формы для редактирования

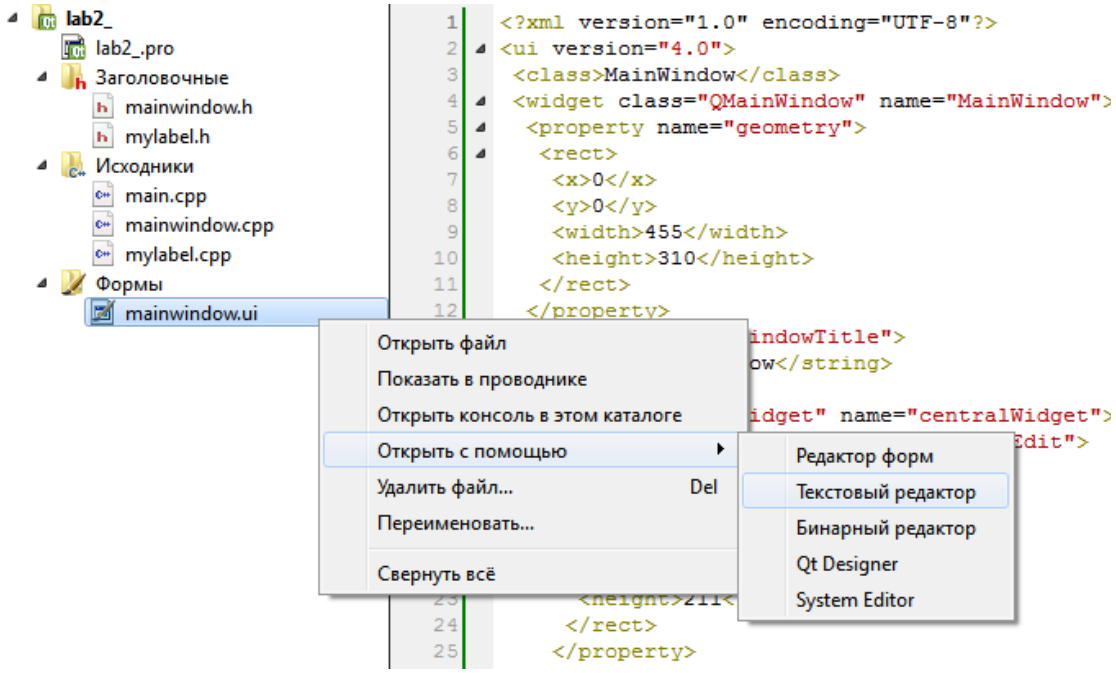

Рисунок 2.3 – Открытие разметки формы

Изменяем класс второй QLabel на наш переопределенный MyLabel:

```
<widget class="MyLabel" name="label_2" native="true">
 <property name="geometry">
  <rect>
   <x>90</x>
   <y>110</y>
   <width>16</width>
```

```
      <height>16</height>
     </rect>
    </property>
    <property name="text" stdset="0">
     <string>0</string>
    </property>
   </widget>
```

7. На главное форме QMainWindow определяем слот addWord, обеспечивающий копирование слова из строчного в многострочное поле. При обнаружении слова “помидор” – вызываем слот MyLabel, чтобы увеличить количество штрафных баллов.

Также соединяем сигнал нажатия клавиши с новым слотом addWord и сигнал MyLabel::banUser со слотом hide для кнопки.

MainWindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMessageBox>
#include <mylabel.h>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    const QString bannedWord = "помидор";

    Ui::MainWindow *ui;


private slots:
    void addWord();
};

#endif // MAINWINDOW_H
```

MainWindow.cpp

```
#include "mainwindow.h"
```

```
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    //соединяем сигналы и слоты
    connect(ui->pushButton, SIGNAL(clicked()), this,
SLOT(addWord()));
    connect(ui->label_2, SIGNAL(banUser()), ui->pushButton,
SLOT(hide()));
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::addWord()
{
    QString newWord = ui->lineEdit->text();
//слово из QLineEdit
    if (!newWord.compare(bannedWord, Qt::CaseInsensitive))  //если
это «помидор»
    {
        //выводим сообщешие
        QMessageBox msgBox;
        msgBox.setText("И не стыдно?");
        msgBox.setStandardButtons(QMessageBox::Ok);
        msgBox.exec();
        //увеличиваем количество штрафа
        ui->label_2->incValue();
    }
    else
    {
        //иначе добавляем слово в QTextEdit
        ui->textEdit->append(newWord + '\n');
    }
}
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

3.1. Изучить принципы работы механизма сигналов и слотов в Qt, способы соединения сигналов и слотов (выполняется в ходе домашней подготовки к лабораторной работе).

3.2. Создать проект Qt Gui Application.

3.3. Создать класс-наследник класса QLabel, добавив собственный сигнал, который будет посылаться, когда значение QLabel равно числу, большему десяти.

3.4. Разместить на форме виджеты QLineEdit, QPushButton, два виджета

QPlainTextEdit и виджет созданного на шаге 3.3 наследника QLabel.

3.5. Обеспечить изменение названия заголовка окна приложения на значение, введенное в QLineEdit при нажатии на кнопку.

3.6. Создать собственный слот для MainWindow, который будет копировать текст из первого QPlainTextEdit во второй, заменяя все символы ‘a’ на ‘*’.

3.7. Подключить слот, созданный на предыдущем этапе к textChanged сигналу первого QPlainTextEdit, таким образом обеспечив автоматическое копирование.

3.8. Создать собственный слот для MainWindow, который будет выводить количество ‘*’ во втором QPlainTextEdit в QLabel.

3.9. Подключить слот, созданный на предыдущем этапе к textChanged сигналу второго QPlainTextEdit, таким образом обеспечив автоматическое подсчет количества символов ‘*’.

3.10. Подключить слот setDisabled первого QPlainTextEdit к сигналу, созданному на шаге 3.3, тем самым обеспечив запрет на дальнейший ввод (setDisabled слот) при вводе более десяти символов ‘a’.

3.11. Выполнить экспериментальное исследование полученного приложения, выполняя ввод тестовых последовательностей с различным количеством символов ‘a’ и различным их положением во вводимой строке: в начале, в середине и в конце.

4. СОДЕРЖАНИЕ ОТЧЕТА

4.1. Цель работы.
4.2. Постановка задачи.
4.3. Словесное описание порядка взаимодействия элементов интерфейса.
4.4. Внешний вид окна приложения.
4.5. Текст программы.
4.6. Результаты исследования программы.
4.7. Выводы.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

5.1. Какие существуют подходы к организации взаимодействия элементов графического интерфейса?

5.2. Каковы плюсы и минусы каждого подхода к организации взаимодействия элементов графического интерфейса?

5.3. В чем состоит особенность описания класса, использующего механизм сигналов и слотов?

5.4. Что происходит при компиляции класса, использующего механизм сигналов и слотов?

5.5. Каким образом создаются собственные сигналы и слоты?
5.6. Каким образом вызвать созданный сигнал?
5.7. Как соединить сигналы и слоты?
5.8. Возможно ли соединение двух сигналов?
5.9. Возможно ли соединение сигнала с несколькими слотами?
5.10. Как можно разъединить сигнал и слот?
5.11. Каким образом возможно осуществить передачу параметра из сигнала в слот?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ж. Бланшет. Qt 3: программирование GUI на C++ / Бланшет Ж., Саммерфилд М. — М. : КУДИЦ — ОБРАЗ, 2005. - 448 с.

2. Е.Р. Алексеев. Программирование на языке C++ в среде Qt Creator /Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк, О. В. Чеснокова, А. С. Чмыхало.— М.:Альт Линукс, 2015. — 448 с.

3. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++/Г. Буч. — М. : БИНОМ ; СПб. : Невский диалект, 2001. - 560 с.

4. Шилдт, Г. C++: базовый курс, 3-е издание /Г. Шилдт. — М.: «Вильямс», 2012. — 624 с.

5. Шилдт, Г. Полный справочник по C++, 4-е издание /Г. Шилдт. — М.: «Вильямс», 2011. — 800 с.

Заказ № _____ от «_____» _____ 2018 г. Тираж экз.