МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное автономное бюджетное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационных систем

Волков Андрей Алексеевич

Институт информационных технологий и управления в технических системах

курс 4 группа ИС(б) – 41-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

о лабораторных работ №4-5

по дисциплине «Кроссплатформенное программирование»

на тему: «Исследование способов построения интерфейса пользователя с помощью языка разметки qml. Исследование способов интеграции интерфейса пользователя на языке qml и функциональности на языке c++»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. преподаватель \_\_\_\_\_\_\_\_\_\_ Киселев

(должность) (подпись) (инициалы, фамилия)

Севастополь 2017

1. Цель работы

Исследование способов взаимодействия языка C++ и языка разметки QML. Приобретение навыков разработки приложений на основе QML-интерфейса.

Изучить основы языка разметки QML. Приобрести практические навыки создания графических интерфейсов Qt-приложений на основе разметки.

1. Вариант задания

|  |  |
| --- | --- |
| Вариант | Описание |
| 1 | Внешний вид приложения показан на рисунке 1. При нажатии на кнопку «Сравнить», осуществить сравнение текста в первом поле ввода с текстом во втором. Результат сравнения вывести соответствующим сообщением в текстовую метку («Текст совпадает» или «Текст не совпадает») |
| 1 | Добавить кнопки, позволяющие сохранять и загружать текст из выбранного поля ввода. |

1. Ход работы

Код main.qml:

import QtQuick 2.7

import QtQuick.Controls 2.0

import QtQuick.Layouts 1.3

ApplicationWindow {

visible: true

width: 380

height: 480

color: "#000000"

title: *qsTr*("4 lab KPP")

Rectangle {

x: 10

y: 20

width: 360

height: 150

border.width: 1

border.color: "#000000"

radius: 10

TextEdit {

id: *inputEdit*

x: 105

y: 25

width: 100

height: 100

color: "#000000"

text: "Text"

selectedTextColor: "#ff1493"

selectionColor: "#ff1493"

cursorVisible: true

font.bold: true

font.pixelSize: 30

}

}

Rectangle {

x: 10

y: 180

width: 360

height: 150

border.width: 1

border.color: "#000000"

radius: 10

TextEdit {

id: *outputEdit*

x: 105

y: 25

width: 100

height: 100

color: "#000000"

text: "Text"

selectedTextColor: "#ff1493"

selectionColor: "#ff1493"

cursorVisible: true

font.bold: true

font.pixelSize: 30

}

}

Label {

id: *lab*

x: 27

y: 360

color: "#00FFFF"

text: "Результат: "

horizontalAlignment: Label.AlignRight

font.bold: true

font.pixelSize: 30

}

Button1 {

id: *button1*

x: 20 // координата Х

y: 410 // координата Y

textColor: "#000000"

labelSize: 14

label: "Сравнить"

buttonWidth: 100

buttonHeight: 50

onButtonClick: {

*lab*.text = " "

if (*inputEdit*.text == *outputEdit*.text)

*lab*.text = *lab*.text + " Текст совпадает"

else *lab*.text = *lab*.text + " Текст не совпадает"

}

}

Button1 {

id: *button2*

x: 140 // координата Х

y: 410 // координата Y

textColor: "#000000"

labelSize: 14

label: "Сохранить"

buttonWidth: 100

buttonHeight: 48

onButtonClick: {

*mainwindow*.add(*inputEdit*.text)

}

}

Button1 {

id: *button3*

x: 260 // координата Х

y: 410 // координата Y

textColor: "#000000"

labelSize: 14

label: "Загрузить"

buttonWidth: 100

buttonHeight: 50

onButtonClick: {

*outputEdit*.text = *mainwindow*.getResult();

}

}

}

Код Button1.qml:

import QtQuick 2.7

Rectangle {

id: *button1*

property int buttonHeight: 75

property int buttonWidth: 150

property string label

property color textColor: *buttonLabel*.color

property color onHoverColor: "lightsteelblue"

property color borderColor: "transparent"

property color buttonColor: "lightblue"

property real labelSize: 14

radius: 6

antialiasing: true

border { width: 2; color: *borderColor* }

width: *buttonWidth*; height: *buttonHeight*

Text

{

id: *buttonLabel*

anchors.centerIn: *parent*

text: *label* //привязываем текст к тексту "родителя"

color: "#000000"

font.pointSize: *labelSize*

}

signal buttonClick()

MouseArea {

id: *buttonMouseArea*

anchors.fill: *parent* //размер равен размеру "родителя"

onClicked: *buttonClick*()

hoverEnabled: true

//отобразить рамку, если навели курсор

onEntered: *parent*.border.color = *onHoverColor*

//удалить рамку при удалении курсор а с кнопки

onExited: *parent*.border.color = *borderColor*

}

color: *buttonMouseArea*.pressed ? *Qt*.darker(*buttonColor*, 1.5) : *buttonColor*

//анимация смены цвета

Behavior on color { ColorAnimation{ duration: 55 } }

scale: *buttonMouseArea*.pressed ? 1.1 : 1.00

//анимация при увеличении кнопки

Behavior on scale { NumberAnimation{ duration: 55 }}

}

Код mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QObject>

#include <QQuickItem>

class mainwindow : public QObject

{

Q\_OBJECT

Q\_PROPERTY(QString curResult READ getResult WRITE add NOTIFY curResultChange)

public:

QString test;

explicit mainwindow(QObject \*parent = nullptr);

Q\_INVOKABLE void add(QString result);

Q\_INVOKABLE void getResult();

~*mainwindow*();

private:

QFile FILENAME;

QString m\_curResult;

signals:

void curResultChange(QString curResult);

public slots:

};

#endif // MAINWINDOW\_H

Код mainwindow.cpp:

#include "mainwindow.h"

mainwindow::mainwindow(QObject \*parent) : QObject(parent)

{

this->FILENAME = "results.txt";

}

mainwindow::~*mainwindow*()

{

}

void mainwindow::add(QString result)

{

QFile file(FILENAME);

curResult = getResult();

if (file.*open*(QFile::ReadWrite | QIODevice::Truncate)) {

QTextStream inStream(&file);

curResult += result;

QTextStream outStream(&file);

outStream << curResult;

}

file.*close*();

emit curResult();

}

void mainwindow::getResult()

{

QFile file(FILENAME);

if (file.*open*(QFile::ReadWrite)) {

QTextStream inStream(&file);

curResult = inStream.readAll().toStdString();

inputEdit.text = curResult;

}

file.*close*();

emit curResult();

}

Код main.cpp:

#include <QtGui/QGuiApplication>

#include <QQmlContext>

#include <QQmlEngine>

#include <QQuickView>

#include <QObject>

#include <QQuickItem>

#include "mainwindow.h"

int main(int argc, char \*argv[])

{

QGuiApplication app(argc, argv);

QQuickView view;

QQmlContext \*ctxt = view.rootContext();

ctxt->setContextProperty("mainwindow", 0);

view.setSource(QUrl ("qrc:/main.qml"));

return app.exec();

}

Далее приведен тест полученной программы:

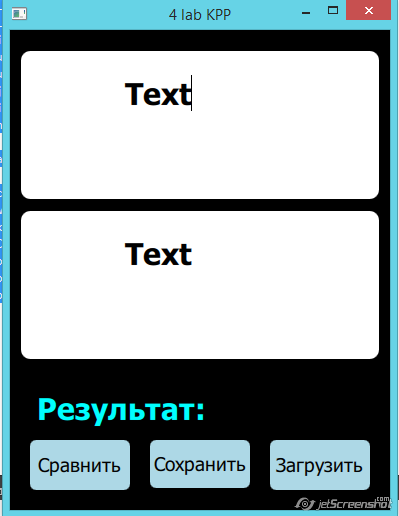


Рисунок 1 – Тест программы л.р. №4-5

Выводы

В ходе выполнения лабораторных работ исследовал способы взаимодействия языка C++ и языка разметки QML. Приобрел навыки разработки приложений на основе QML-интерфейса. Изучил основы языка разметки QML. Приобрел практические навыки создания графических интерфейсов Qt-приложений на основе разметки.