МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Программная инженерия»

**ОТЧЁТ**

по лабораторной работе №8

**BACKEND C++**

**Выполнил:** студент группы

382008-1

\_\_\_А.Д. Кандрин \_\_\_\_\_\_\_

Подпись

**Проверил:** младший научный сотрудник

\_\_\_\_Д.А. Карчков\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород  
2023 г.

## Содержание

[1. Постановка задачи 3](#_Toc27873363)

[2. ХОД РАБОТЫ](#_Toc27873364) 4

[3. ЗАКЛЮЧЕНИЕ](#_Toc27873365) 9

[4. ЛИТЕРАТУРА](#_Toc27873374) 10

## Постановка задачи.

Цель: научиться использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении.

## Ход работы.

1. Создадим класс-счётчик с полем для хранения текущего значения и методами для увеличения значения на единицу и сброса до нуля.

// header

#ifndef COUNTER\_H

#define COUNTER\_H

#include <QObject>

class Counter : public QObject

{

Q\_OBJECT

Q\_PROPERTY(int value READ get WRITE set RESET reset NOTIFY valueChanged)

private:

int value;

public:

Q\_INVOKABLE explicit Counter(QObject \*parent = nullptr);

Q\_INVOKABLE int get() const;

Q\_INVOKABLE void set(int value);

Q\_INVOKABLE void reset();

Q\_INVOKABLE void increment();

signals:

void valueChanged();

};

#endif // COUNTER\_H

// cpp-file

#include "counter.h"

Counter::Counter(QObject \*parent) : QObject(parent)

{

}

int Counter::get() const

{

return value;

}

void Counter::set(int value)

{

this->value = value;

emit valueChanged();

}

void Counter::reset()

{

set(0);

}

void Counter::increment()

{

set(value + 1);

}

1. Используем мета-объект класса-счётчика для создания объекта и вызова его методов (использовать функцию main, результат изменения состояния проверим выводом на консоль).

const QMetaObject\* staticObject = &Counter::staticMetaObject;

QObject\* object = staticObject->newInstance();

int value = -1;

assert(QMetaObject::invokeMethod(object, "get", Q\_RETURN\_ARG(int, value)));

qDebug() << "Init value = " << value;

assert(QMetaObject::invokeMethod(object, "increment"));

assert(QMetaObject::invokeMethod(object, "get", Q\_RETURN\_ARG(int, value)));

qDebug() << "Value after increment = " << value;

assert(QMetaObject::invokeMethod(object, "reset"));

assert(QMetaObject::invokeMethod(object, "get", Q\_RETURN\_ARG(int, value)));

qDebug() << "Value after reset = " << value;

assert(QMetaObject::invokeMethod(object, "set", Q\_ARG(int, 777)));

assert(QMetaObject::invokeMethod(object, "get", Q\_RETURN\_ARG(int, value)));

qDebug() << "Value after set = " << value;

1. Создадим приложение с текстовым полем и двумя кнопками. Использовать класс-счётчик в QML: текстовое поле будет отображать текущее значение счётчика, кнопки используются для увеличения значения счётчика на единицу и сброса значения до нуля.

// cpp-file:

qmlRegisterType<Counter>("harbour.Lab8.Counter", 0, 1, "Counter");

// qml-file:

import harbour.Lab8.Counter 0.1

Page {

// 3. Создать приложение с текстовым полем и двумя кнопками. Использовать класс-счётчик в QML:

// текстовое поле должно отображать текущее значение счётчика,

// кнопки используются для увеличения значения счётчика на единицу и сброса значения до нуля.

id: page

Counter

{

id: counter

// 4. Сделать поле со значением счётчика свойством и инициализировать его каким-либо значением при создании объекта в QML

value: 0 // проинициализровали каким-либо значением (нулем)

}

Column {

id: column

width: page.width

spacing: Theme.paddingLarge

Label {

id: label

x: Theme.horizontalPageMargin

anchors.horizontalCenter: parent.horizontalCenter

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

text: counter.value

}

Button

{

width: parent.width

text: "Увеличить"

onClicked: counter.increment()

}

Button

{

width: parent.width

text: "Сбросить"

onClicked: counter.reset()

}

}

}

1. Создадим класс, содержащий список из строк. Класс будет содержать методы для добавления строки в список и удаления последней добавленной строки, а также реализуем свойство только для чтения, которое позволяет получить список всех строк в виде одной, перечисленных через запятую и использовать это свойство для вывода добавленных строк на экран.

// h-file

#ifndef MYSTRINGLIST\_H

#define MYSTRINGLIST\_H

#include <QObject>

class MyStringList : public QObject

{

Q\_OBJECT

Q\_PROPERTY(QString text READ getText NOTIFY textChanged)

private:

QStringList stringList;

public:

explicit MyStringList(QObject \*parent = nullptr);

Q\_INVOKABLE QString getText() const;

Q\_INVOKABLE void addString(QString string);

Q\_INVOKABLE void removeLastString();

signals:

void textChanged();

};

#endif // MYSTRINGLIST\_H

// cpp-file

#include "mystringlist.h"

MyStringList::MyStringList(QObject \*parent) : QObject(parent)

{

}

QString MyStringList::getText() const

{

QString result = stringList.join(",").toLower();

result[0] = result[0].toUpper();

return result;

}

void MyStringList::addString(QString string)

{

stringList.push\_back(string);

emit textChanged();

}

void MyStringList::removeLastString()

{

if (!stringList.empty())

{

stringList.pop\_back();

emit textChanged();

}

}

// main-function:

qmlRegisterType<MyStringList>("harbour.Lab8.MyStringList", 0, 1, "MyStringList");

1. Создадим приложение, позволяющее добавить введённое слово и удалить последнее добавленное с использованием данного класса в QML. Слова сохраняются в нижнем регистре.

Page {

id: page

Column {

id: column

width: page.width

spacing: Theme.paddingLarge

MyStringList

{

id: stringList

}

TextField

{

id: textField

width: parent.width

placeholderText: "Введите слово"

}

Button

{

width: parent.width

text: "Добавить слово"

onClicked: stringList.addString(textField.text.toLocaleLowerCase())

}

Button

{

width: parent.width

text: "Удалить слово"

onClicked: stringList.removeLastString()

}

TextArea {

width: parent.width

height: 700

anchors.horizontalCenter: parent.horizontalCenter

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

text: stringList.text

readOnly: true

}

}

}

1. **Заключение.**

Мы научились использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении.

**4. Литература.**

1. Документация Qt: сайт. — URL: https://doc.qt.io (дата обращения: 01.03.2022). — Текст: электронный.