

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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

Отчет по практическому заданию для лекции №8

Выполнила:

студентка группы 382006-2

Кулёва Анна Андреевна

Проверил:

Карчков Денис Александрович

Нижний Новгород

2023

Содержание

Содержание	2
1. Цель практического занятия.....	3
2. Постановка задачи	4
3. Руководство пользователя	6
4. Руководство программиста.....	8
Заключение.....	12
Приложение.....	13

1. Цель практического занятия

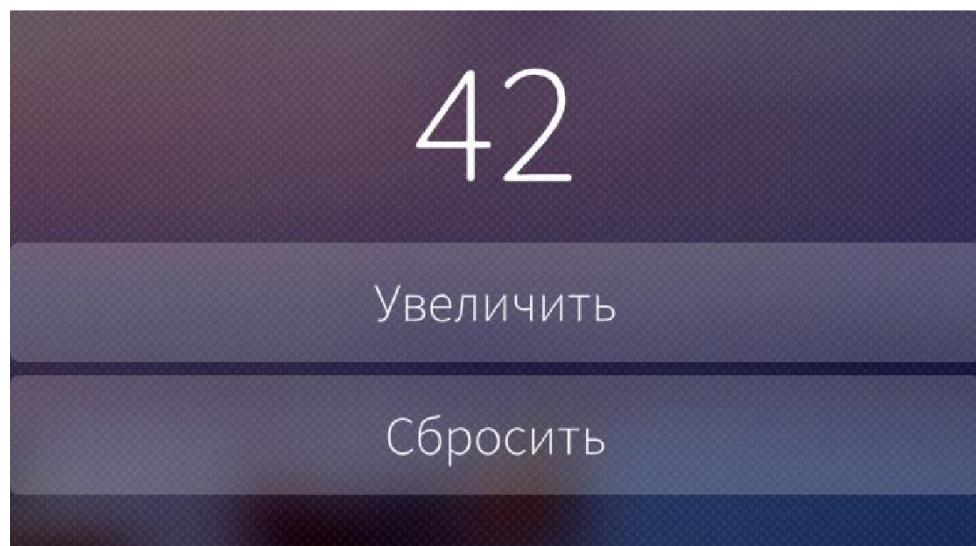
Цель данного практического занятия состоит в том, чтобы научиться использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении.

2. Постановка задачи

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

2. Использовать мета-объект класса-счётчика для создания объекта и вызова его методов (использовать функцию `main`, результат изменения состояния проверять выводом на консоль).

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.



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

5. Создать класс, содержащий список из строк. Класс должен содержать методы для добавления строки в список и удаления последней добавленной строки.

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

7. Реализовать 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. Свойство должно моментально реагировать на изменение содержимого списка, первое слово начинается с заглавной буквы.

Авокадо

Добавить слово

Удалить слово

Яблоко, груша, апельсин, манго, авокадо,
ананас

3. Руководство пользователя

При запуске программы пользователь увидит главную страницу с кнопками, по которым сможет перейти к 1 и 2 заданиям. На странице с заданием 1 пользователь сможет взаимодействовать с двумя кнопками: «Увеличить» - увеличивает значение переменной на 1 и «Сбросить» - сбрасывает значение до 0 (рисунок 1). Значение переменной отображается в текстовом поле.

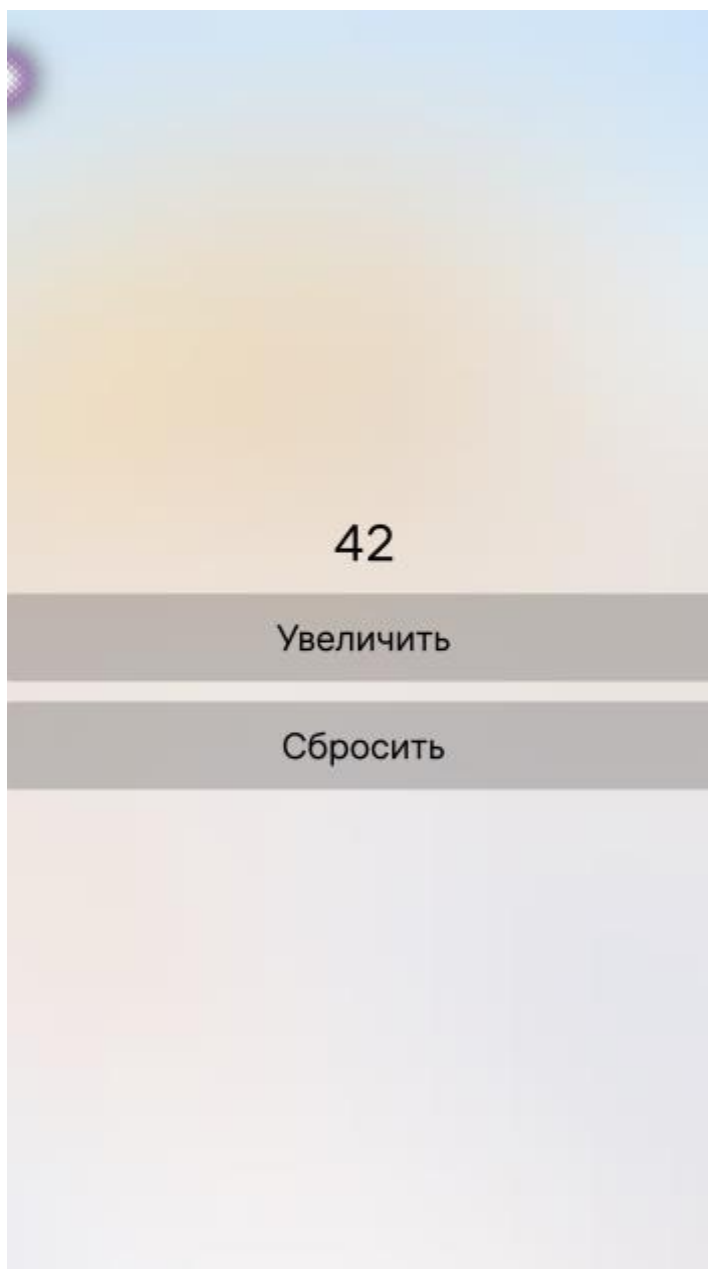
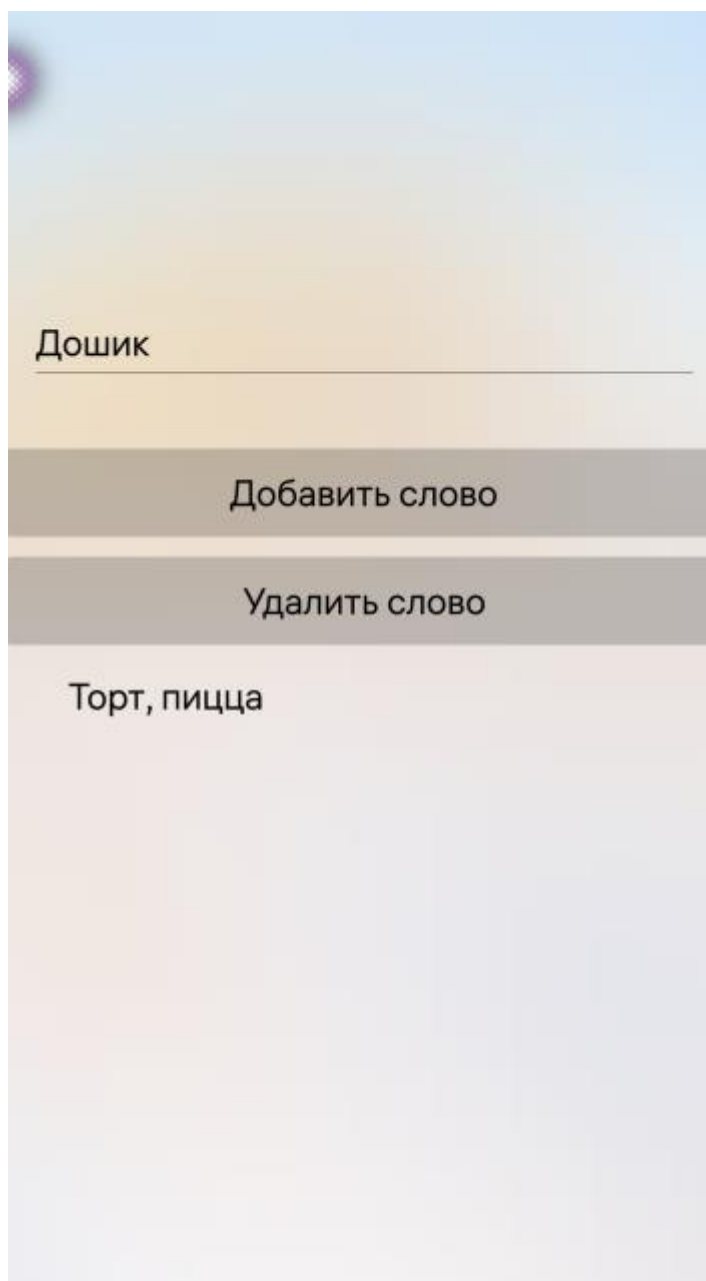


Рисунок 1. Счётчик

На странице с заданием 2 текстовое поле, куда пользователь может ввести слово. Это слово можно добавить в конец списка с помощью кнопки «Добавить слово». Также по кнопке «Удалить слово» можно удалить последнее слово списка (рисунок 2). Слова записываются через запятую, первое – с заглавной буквы.

(рисунок 2).



The image shows a web form with a light blue header. Below the header is a text input field containing the word "Дошик". Underneath the input field are two buttons: "Добавить слово" (Add word) and "Удалить слово" (Delete word). Below these buttons is a list area containing the text "Торт, пицца".

Рисунок 2. Список

4. Руководство программиста

Программа реализована на языке программирования QML.

1. Реализуем класс-счетчик в файле Counter.h с полем для хранения текущего значения `int m_count` и методами для увеличения значения на единицу `Q_INVOKABLE void inc()` и сброса до нуля `Q_INVOKABLE void reset`:

```
class Counter : public QObject
{
    Q_OBJECT
    Q_PROPERTY(int m_count READ getCount WRITE setCount NOTIFY
countChanged)
public:

    Counter(){}
    Q_INVOKABLE int getCount() { return m_count; };
    Q_INVOKABLE void inc() { m_count++; emit countChanged(); };
    Q_INVOKABLE void reset() { m_count = 0; emit countChanged(); };

    Q_INVOKABLE void setCount(const int temp) { m_count = temp; emit
countChanged(); };

signals:
    void countChanged();

private:
    int m_count = 0;
};
```

2. Создаём мета-объект класса-счетчика Counter с id: counter:

```
Counter {
    id: counter
    m_count: 42
}
```

3. Создаём приложение с текстовым полем Label и двумя кнопками Button, которые при нажатии вызывают `counter.inc()` и `counter.reset()` соответственно. При создании приложения используется класс-счетчик в QML:

```
Column {
    id: column
    width: parent.width
    anchors.centerIn: parent
    spacing: 20
```



```

Label {
    anchors.horizontalCenter: parent.horizontalCenter
    id: L1
    text: counter.m_count
    font.pixelSize: Theme.fontSizeExtraLarge
}

Button {
    anchors.horizontalCenter: parent.horizontalCenter
    width: parent.width
    text: "Увеличить"
    onClicked: {
        counter.inc();
    }
}

Button {
    anchors.horizontalCenter: parent.horizontalCenter
    width: parent.width
    text: "Сбросить"
    onClicked: {
        counter.reset()
    }
}

```

4. Создаём поле со значением счётчика свойством:

```

Q_PROPERTY(int m_count READ getCount WRITE setCount NOTIFY
countChanged)

```

5. Создаём класс, содержащий список из строк. Он содержит методы для добавления строки в список `Q_INVOKABLE void add(QString temp)` и удаления последней добавленной строки `Q_INVOKABLE void popBack()` :

```

class StringList : public QObject
{
    Q_OBJECT
public:
    StringList() {}
    Q_INVOKABLE void add(QString temp) { m_data << temp; };
    Q_INVOKABLE void popBack()
    {
        if (!m_data.isEmpty()) {
            m_data.pop_back();
        }
    };
private:
    QList<QString> m_data;
};

return temp;
};
private:
    QList<QString> m_data;
};

```

6. Создаём приложение, позволяющее добавить введённое слово и удалить последнее добавленное с использованием данного класса в QML:

```
Page {
    StringList {
        id: stringList
    }
    Column {
        id: column
        width: parent.width
        anchors.horizontalCenter: parent.horizontalCenter
        spacing: 20
        y: 300
        TextField {
            id: textField;
            placeholderText: "Введите слово"
        }
        Button {
            text: "Добавить слово"
            width: parent.width
            onClicked: {
                stringList.add(textField.text)
                label.text = stringList.getAll();
            }
            anchors.horizontalCenter: parent.horizontalCenter
        }
        Button {
            text: "Удалить слово"
            width: parent.width
            onClicked: {
                stringList.popBack();
                label.text = stringList.getAll();
            }
            anchors.horizontalCenter: parent.horizontalCenter
        }
    }

    TextField {
        id: label;
        width: parent.width;
        x: Theme.horizontalPageMargin
        text: stringList.getAll()
        readOnly: true;
        wrapMode: TextInput.Wrap
    }
}
```

7. 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:

```
Q_INVOKABLE QString getAll()
```

```

{
    QString temp;
    for (int i = 0; i < m_data.length(); i++)
    {
        if (i == 0) {
            QString t = m_data[i];
            t[0] = t[0].toUpper();
            temp += t;
        } else {
            temp += m_data[i].toLower();
        }

        if (i != m_data.length()-1){
            temp += ", ";
        }
    }

    return temp;
};

```

Заключение

В данной лабораторной работе я научилась использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении. Также были выполнены все шаги практического задания.

Приложение

Counter.h

```
#ifndef COUNTER_H
#define COUNTER_H

#include <QObject>
#include <QString>

class Counter : public QObject
{
    Q_OBJECT
    Q_PROPERTY(int m_count READ getCount WRITE setCount NOTIFY countChanged)
public:
    // explicit Counter(QObject *parent = nullptr);
    Counter() {}
    Q_INVOKABLE int getCount() { return m_count; };
    Q_INVOKABLE void inc() { m_count++; emit countChanged(); };
    Q_INVOKABLE void reset() { m_count = 0; emit countChanged(); };

    Q_INVOKABLE void setCount(const int temp) { m_count = temp; emit
countChanged(); };

signals:
    void countChanged();

private:
    int m_count = 0;
};

#endif // COUNTER_H
```

StringList.h

```
#ifndef STRINGLIST_H
#define STRINGLIST_H

#include <QObject>
#include <QString>

class StringList : public QObject
{
    Q_OBJECT
public:
    // explicit StringList(QObject *parent = nullptr);
    StringList() {}
    Q_INVOKABLE void add(QString temp) { m_data << temp; };
    Q_INVOKABLE void popBack()
    {
        if (!m_data.isEmpty()) {
            m_data.pop_back();
        }
    };

    Q_INVOKABLE QString getAll()
    {
        QString temp;
    }
};
```

```

        for (int i = 0; i < m_data.length(); i++)
        {
            if (i == 0) {
                QString t = m_data[i];
                t[0] = t[0].toUpper();
                temp += t;
            } else {
                temp += m_data[i].toLower();
            }

            if (i != m_data.length()-1) {
                temp += ", ";
            }
        }

        return temp;
    };
private:
    QList<QString> m_data;
};

#endif // STRINGLIST_H

```

main.cpp

```

#include <QScopedPointer>
#include <QGuiApplication>
#include <QQuickView>
#include "counter.h"
#include "StringList.h"
#include "Calc.h"

#include <sailfishapp.h>

int main(int argc, char *argv[])
{
    QScopedPointer<QGuiApplication>
application(SailfishApp::application(argc, argv));
    application->setOrganizationName(QStringLiteral("ru.auroraos"));
    application->setApplicationName(QStringLiteral("Lab7"));

    qmlRegisterType<Counter>("harbour.appname.counter", 1, 0, "Counter");
    qmlRegisterType<StringList>("harbour.appname.stringlist", 1, 0,
"StringList");
    qmlRegisterType<Calc>("harbour.appname.calc", 1, 0, "Calc");

    QScopedPointer<QQuickView> view(SailfishApp::createView());
    view->setSource(SailfishApp::pathTo(QStringLiteral("qml/Lab7.qml")));
    view->show();

    Counter myCounter;
    int cnt = myCounter.getCount();
    //printf("m_count: %d", cnt);
    qDebug() << cnt << "\n";
    myCounter.inc();
    cnt = myCounter.getCount();
    //printf("m_count: %d", cnt);
    qDebug() << cnt << "\n";
    myCounter.reset();

    return application->exec();
}

```

Page1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import harbour.appname.counter 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Counter {
        id: counter
        m_count: 42
    }

    Column {
        id: column
        width: parent.width
        anchors.centerIn: parent
        spacing: 20

        Label {
            anchors.horizontalCenter: parent.horizontalCenter
            id: ll
            text: counter.m_count
            font.pixelSize: Theme.fontSizeExtraLarge
        }

        Button {
            anchors.horizontalCenter: parent.horizontalCenter
            width: parent.width
            text: "Увеличить"
            onClicked: {
                counter.inc();
            }
        }
        Button {
            anchors.horizontalCenter: parent.horizontalCenter
            width: parent.width
            text: "Сбросить"
            onClicked: {
                counter.reset()
            }
        }
    }
}
```

Page2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import harbour.appname.stringlist 1.0

Page {

    StringList {
        id: stringList
    }

    Column {
        id: column
```

```

width: parent.width
anchors.horizontalCenter: parent.horizontalCenter
spacing: 20
y: 300

TextField {
    id: textField;
    placeholderText: "Введите слово"
}

Button {
    text: "Добавить слово"
    width: parent.width
    onClicked: {
        stringList.add(textField.text)
        label.text = stringList.getAll();
    }
    anchors.horizontalCenter: parent.horizontalCenter
}

Button {
    text: "Удалить слово"
    width: parent.width
    onClicked: {
        stringList.popBack();
        label.text = stringList.getAll();
    }
    anchors.horizontalCenter: parent.horizontalCenter
}

TextField {
    id: label;
    width: parent.width;
    x: Theme.horizontalPageMargin
    text: stringList.getAll()
    readOnly: true;
    wrapMode: TextInput.Wrap
}
}

```