МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №1

по дисциплине

Разработка кроссплатформенных приложений

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Проверил: зав. кафедры

«Информатика» Халабия Р.Ф.

Москва, 2025 г

**Задание**

— Создать виджет с надписью «Hello, World!», задать размер виджета,

вывести его на экран. В качестве виджета использовать кнопку (объект button

класса QPushButton).

— Создать текстовую метку при помощи класса QLabel, в текстовой

метке записать текст «Hello, Qt!». Вывести сообщение с использованием указателей и класса QLabel.

— Повторить предыдущий пункт задания без использования указателей.

— Создать кнопку при помощи класса QPushButton с текстом «Quit».

При нажатии на кнопку должно происходить завершение работы программы.

— Создать объект класса QDate, содержащий текущую дату, создать

объект класса QTime, содержащий текущее время, вывести полученную информацию о дате и времени в текстовую метку с использованием объектов

класса QString.

— Используя события таймера, через каждые 200 мс выводить в текстовую метку сообщение «Blink».

— Используя Qt Designer, разместить на форме объект класса QLabel

(текстовую метку), в которую необходимо выводить текущее время (ежесекундно).

— Используя объекты класса QFile, скопировать содержимое одного

бинарного файла в другой блоками по 2 символа, при этом осуществить проверку на возможность открытия файлов для чтения и записи.

— Повторить предыдущий пункт задания с использованием объектов

класса QByteArray.

— Используя объекты класса QBuffer, осуществить вывод текстовой

строки в консоль.

— Используя объекты класса QTextStream, построчно считать содержимое текстового файла и вывести его в консоль.

— Повторить предыдущий пункт задания с использованием метода

readAll(). Записать текстовую информацию в файл, предварительно отформатировав текст

**ВЫПОЛНЕНИЕ**

Выполнение заданий представлено на рисунках 1-

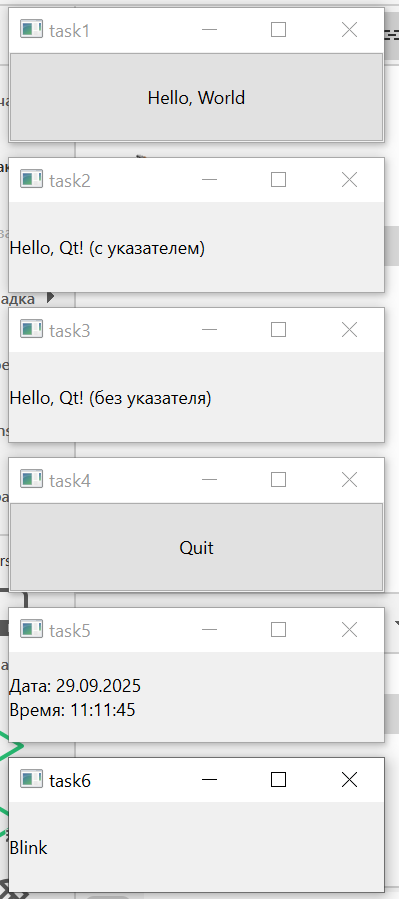
****

Рисунок 1 – Окна на экране

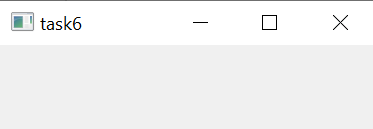


Рисунок 2 – Сообщение «blink» мигает в нужном окне

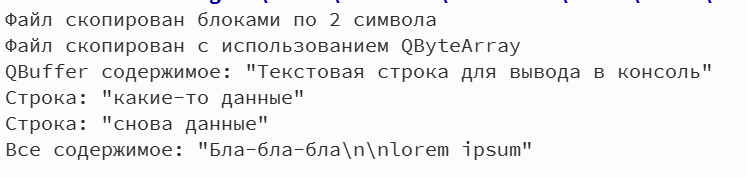
****

Рисунок 3 – Содержимое консоли



Рисунок 4 – Содержимое файла «source.bin»



Рисунок 5 – Содержимое файла «destination.bin». Скопировано успешно



Рисунок 6 – Содержимое файла «destination2.bin». Скопировано успешно

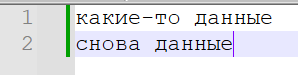


Рисунок 7 – Содержимое файла «textfile.txt».

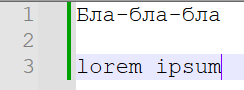


Рисунок 8 – Содержимое файла «read.txt».

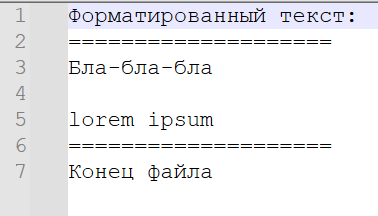


Рисунок 9 – Содержимое файла «write.txt». Успешно добавлено форматирование

**Алгоритм мигания текста Blink:**

**Шаг 1.** Создать метку с текстом "Blink".

**Шаг 2.** Создать таймер QTimer и настроить его срабатывание каждые 200 миллисекунд.

**Шаг 3.** При каждом срабатывании таймера выполнить действие:

* Если текст метки пустой, то установить "Blink".
* Если в метке уже "Blink", то стереть текст (сделать пустую строку).

**Шаг 4.** Повторять Шаг 3 бесконечно, пока приложение работает.

Листинг 1 — main.cpp

#include <QApplication>

#include <QPushButton>

#include <QLabel>

#include <QDate>

#include <QTime>

#include <QTimer>

#include <QFile>

#include <QBuffer>

#include <QTextStream>

#include <QDebug>

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

{

QApplication app(argc, argv);

**int** x = **50**; // начальная координата X

**int** y = **50**; // начальная координата Y

**int** step = **100**; // расстояние между элементами

QSize widgetSize(**250**, **60**); // общий размер для кнопок и меток

// 1. Кнопка "Hello, World"

QPushButton helloButton("Hello, World");

helloButton.resize(widgetSize);

helloButton.move(x, y);

helloButton.setWindowTitle("task1");

helloButton.show();

// 2. QLabel с указателем

y += step;

QLabel \*labelWithPointer = **new** QLabel("Hello, Qt! (с указателем)");

labelWithPointer->resize(widgetSize);

labelWithPointer->move(x, y);

labelWithPointer->setWindowTitle("task2");

labelWithPointer->show();

// 3. QLabel без указателя

y += step;

QLabel labelWithoutPointer("Hello, Qt! (без указателя)");

labelWithoutPointer.resize(widgetSize);

labelWithoutPointer.move(x, y);

labelWithoutPointer.setWindowTitle("task3");

labelWithoutPointer.show();

// 4. Кнопка "Quit"

y += step;

QPushButton \*quitButton = **new** QPushButton("Quit");

quitButton->resize(widgetSize);

quitButton->move(x, y);

quitButton->setWindowTitle("task4");

QObject::connect(quitButton, &QPushButton::clicked, &app, &QApplication::quit);

quitButton->show();

// 5. Дата и время

y += step;

QDate currentDate = QDate::currentDate();

QTime currentTime = QTime::currentTime();

QString dateTimeString = QString("Дата: %1**\n**Время: %2")

.arg(currentDate.toString("dd.MM.yyyy"))

.arg(currentTime.toString("hh:mm:ss"));

QLabel \*dateTimeLabel = **new** QLabel(dateTimeString);

dateTimeLabel->resize(widgetSize);

dateTimeLabel->move(x, y);

dateTimeLabel->setWindowTitle("task5");

dateTimeLabel->show();

// 6. Мигающая метка

y += step;

QLabel \*blinkLabel = **new** QLabel("Blink");

blinkLabel->resize(widgetSize);

blinkLabel->move(x, y);

blinkLabel->setWindowTitle("task6");

blinkLabel->show();

QTimer \*timer = **new** QTimer();

QObject::connect(timer, &QTimer::timeout, [blinkLabel]() {

**static** **bool** showText = true;

blinkLabel->setText(showText ? "Blink" : "");

showText = !showText;

});

timer->start(**200**);

// 7. Работа с файлами - копирование блоками по 2 символа

QFile sourceFile("source.bin");

QFile destFile("destination.bin");

**if** (sourceFile.open(QIODevice::ReadOnly) && destFile.open(QIODevice::WriteOnly)) {

**while** (!sourceFile.atEnd()) {

QByteArray buffer = sourceFile.read(**2**);

destFile.write(buffer);

}

sourceFile.close();

destFile.close();

qDebug() << "Файл скопирован блоками по 2 символа";

} **else** {

qDebug() << "Проблема при открытии файла";

}

// 8. Копирование с использованием QByteArray

QFile sourceFile2("source.bin");

QFile destFile2("destination2.bin");

**if** (sourceFile2.open(QIODevice::ReadOnly) && destFile2.open(QIODevice::WriteOnly)) {

QByteArray fileData = sourceFile2.readAll();

destFile2.write(fileData);

sourceFile2.close();

destFile2.close();

qDebug() << "Файл скопирован с использованием QByteArray";

}

// 9. QBuffer - вывод строки в консоль

QBuffer buffer;

buffer.open(QIODevice::WriteOnly);

buffer.write("Текстовая строка для вывода в консоль");

buffer.close();

buffer.open(QIODevice::ReadOnly);

QTextStream in(&buffer);

qDebug() << "QBuffer содержимое:" << in.readAll();

buffer.close();

// 10. QTextStream - построчное чтение файла

QFile textFile("textfile.txt");

**if** (textFile.open(QIODevice::ReadOnly | QIODevice::Text)) {

QTextStream textStream(&textFile);

**while** (!textStream.atEnd()) {

QString line = textStream.readLine();

qDebug() << "Строка:" << line;

}

textFile.close();

}

// 11. Чтение всего файла и запись с форматированием

QFile readFile("read.txt");

QFile writeFile("write.txt");

**if** (readFile.open(QIODevice::ReadOnly | QIODevice::Text) &&

writeFile.open(QIODevice::WriteOnly | QIODevice::Text)) {

QTextStream readStream(&readFile);

QTextStream writeStream(&writeFile);

QString content = readStream.readAll();

qDebug() << "Все содержимое:" << content;

writeStream << "Форматированный текст:**\n**";

writeStream << "====================**\n**";

writeStream << content;

writeStream << "**\n**====================**\n**";

writeStream << "Конец файла";

readFile.close();

writeFile.close();

}

**return** app.exec();

}