Специальность: 09.02.07 Информационные системы и программирование

Проверил:

\_\_\_\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «» \_\_\_\_\_ 2020 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

(Университет ИТМО)

факультет среднего профессионального образования

Отчет

о Лабораторной работе №15

по теме: Работа с сокетами

по дисциплине: Разработка программных модулей

Выполнил:

студент группы 2333

\_\_\_\_\_\_\_\_\_\_\_ Сафин Р. М.

Санкт-Петербург 2020

Содержание

[1 Постановка задачи 3](#_Toc35962118)

[1.1 Цель 3](#_Toc35962119)

[1.2 Задачи 3](#_Toc35962120)

[1.3 Задание 3](#_Toc35962121)

[2 ИСХОДНЫЙ КОД 4](#_Toc35962122)

[3 ДИАГРАММЫ 1](#_Toc35962122)0

[4 ЛИСТИНГ ВЫполнения программы 1](#_Toc35962123)2

[5 Вывод 14](#_Toc35962124)

1. Постановка задачи
   1. Цель

Познакомиться с основами работы QTcpSocket в приложениях Qt.

* 1. Задачи
* изучить основы работы с сокетными приложениями;
* создать простейшее приложение-чат.
  1. Задание

Написать клиент и сервер программы чата.

Требования к серверу:

* сервер должен хранить последние 10 сообщений;
* при подключении нового пользователя оповестить существующих о подключении нового пользователя;
* новому пользователю отправить последние 10 сообщений чата;
* обработать команду от пользователя: отключение от чата (Qt::disconnect);
* при закрытии программы последние 10 сообщений должны быть сохранены в файл;

Требования к клиенту

* клиент должен задать ник перед началом общения с чатом;
* сообщения на клиенте должны отображаться в виде: <%nick-name%>:<%message%>;
* клиент может отключиться от сервера в любой момент;

Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код, используя аннотации Doxygen.

.

1. Исходный код

Клиент:

Листинг 1 – Код файла main.cpp

/\*\*

\* @file main.cpp

\* @author Roman Safin

\*/

#include "mainwindow.cpp"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}Листинг 2 – Код файла mainwindow.cpp

/\*\*

\* @file mainwindow.cpp

\* @author Roman Safin

\*/

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QTcpSocket>

#include <QTextStream>

QFile file("..\\file.txt");

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

socket = new QTcpSocket(this);

connect(socket, &QTcpSocket::readyRead,

[&](){

QTextStream T(socket);

auto text = T.readAll();

ui->textEdit->append(text);

});

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_send\_clicked()

{

QTextStream T(socket);

T << ui->nickname->text() << ":" << ui->message->text();

list.push\_back(ui->nickname->text() + ":" + ui->message->text());

if(!file.open(QIODevice::WriteOnly)){

qInfo() << "File isn't open.";

}

k++;

if (k > 10){

list.removeFirst();

}

QTextStream stream(&file);

foreach(QString str, list){

if (str != ""){

stream << str << endl;

}

}

file.close();

}

void MainWindow::on\_connect\_clicked()

{

socket->connectToHost

("localhost",3333);

}

Листинг 3 – Код файла mainwindow.h

/\*\*

\* @file mainwindow.h

\* @author Roman Safin

\*/

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFile>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class QTcpSocket;

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

/\*\*

\* @brief send message

\*/

void on\_send\_clicked();

/\*\*

\* @brief connect to server

\*/

void on\_connect\_clicked();

private:

Ui::MainWindow \*ui;

QTcpSocket \*socket;

QStringList list;

int k = 0;

};

#endif // MAINWINDOW\_H

Сервер:

Листинг 4 – Код файла main.cpp

/\*\*

\* @file main.cpp

\* @author Roman Safin

\*/

#include "server.h"

#include <QCoreApplication>

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

{

QCoreApplication a(argc, argv);

server server;

if (!server.startServer(3333))

{

qDebug() << "Error:" << server.errorString();

return 1;

}

qDebug() << "Server started!";

//MainWindow w;

//w.show();

return a.exec();

}

*Листинг 5 — Код файла server.cpp*

/\*\*

\* @file server.cpp

\* @author Roman Safin

\* @brief Custom server class

\*/

#include "server.h"

#include "socket.h"

#include <QDebug>

#include <QTextStream>

#include <QFile>

QFile file("..\\file.txt");

server::**server**(QObject \*parent)

: QTcpServer(parent)

{

if(!file.*open*(QIODevice::WriteOnly)){

qInfo() << "File isn't open.";

}

file.*resize*(0);

}

bool server::**startServer**(quint16 port)

{

return listen(QHostAddress::Any, port);

}

void server::***incomingConnection***(qintptr handle)

{

qDebug() << "Client connected with handle:" << handle;

auto sock = new socket(handle, this);

LSockets << sock;

sendRecent(sock);

for (auto i: LSockets){

QTextStream T(i);

T << "Connected:" << handle;

i->flush();

}

connect(sock, &socket::SReadyRead, [&](socket \*S){

qDebug() << "SReadyRead";

QTextStream T(S);

auto text = T.readAll();

for (auto i : LSockets) {

QTextStream K(i);

K << text;

i->flush();

}

});

connect(sock, &socket::SStateChanged, [&](socket \*S, int ST)

{

qDebug() << "SStateChanged";

if (ST == QTcpSocket::UnconnectedState)

{

qDebug() << "Disconnected state with handle" << S->*socketDescriptor*();

LSockets.removeOne(S);

for (auto i : LSockets) {

QTextStream K(i);

K << "Client " <<

S->*socketDescriptor*() << " has been disconnected"; //рассказать всем, что сокет отключился

i->flush();

}

}

});

}

void server::**sendRecent**(socket\* s)

{

if(!file.*open*(QIODevice::ReadOnly)){

qInfo() << "File isn't open.";

}

socket\* clSock = s;

QTextStream socket\_stream (clSock);

QTextStream file\_stream(&file);

while(!file\_stream.atEnd()){

QString string;

socket\_stream<<endl;

file\_stream >> string;

socket\_stream << string;

}

socket\_stream<<endl;

file.*close*();

}

*Листинг 6 — Код файла server.h*

/\*\*

\* @file server.h

\* @author Roman Safin

\* @brief Custom server class

\*/

#ifndef SERVER\_H

#define SERVER\_H

#include <QTcpServer>

class **socket**;

class **server** : public QTcpServer

{

public:

public:

server(QObject \*parent = nullptr);

/\*\*

\* @brief start server

\* @param port

\* @return started

\*/

bool startServer(quint16 port);

/\*\*

\* @brief connect/disconnect socket

\* @param socket handle

\*/

void incomingConnection(qintptr handle) override;

/\*\*

\* @brief send last 10 message

\* @param socket

\*/

void sendRecent(socket\* s);private:

QList<socket \*> LSockets;

};

#endif // SERVER\_H

*Листинг 7 — Код файла socket.cpp*

/\*\*

\* @file socket.cpp

\* @author Roman Safin

\* @brief Custom socket class

\*/

#include "socket.h"

socket::**socket**(qintptr handle, QObject \*parent)

{

*setSocketDescriptor*(handle);

connect(this, &socket::readyRead, [&]()

{

emit SReadyRead(this);

});

connect(this, &socket::stateChanged, [&](int S)

{emit SStateChanged(this, S);

});

}

*Листинг 8 — Код файла socket.h*

/\*\*

\* @file socket.h

\* @author Roman Safin

\* @brief Custom socket class

\*/

#ifndef SOCKET\_H

#define SOCKET\_H

#include <QTcpSocket>

class **socket** : public QTcpSocket

{

Q\_OBJECT

public:

**socket**(qintptr handle, QObject \*parent = nullptr);

signals:

void **SReadyRead**(socket \*);

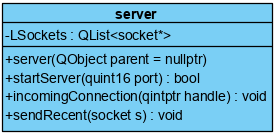
void **SStateChanged**(socket \*, int);

};

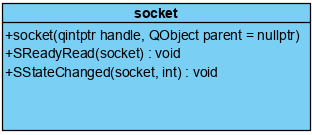
#endif // SOCKET\_H

1. диаграммы

Диаграммы классов:

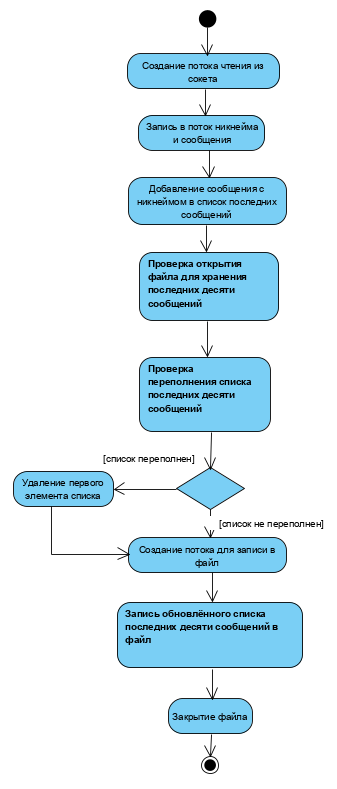


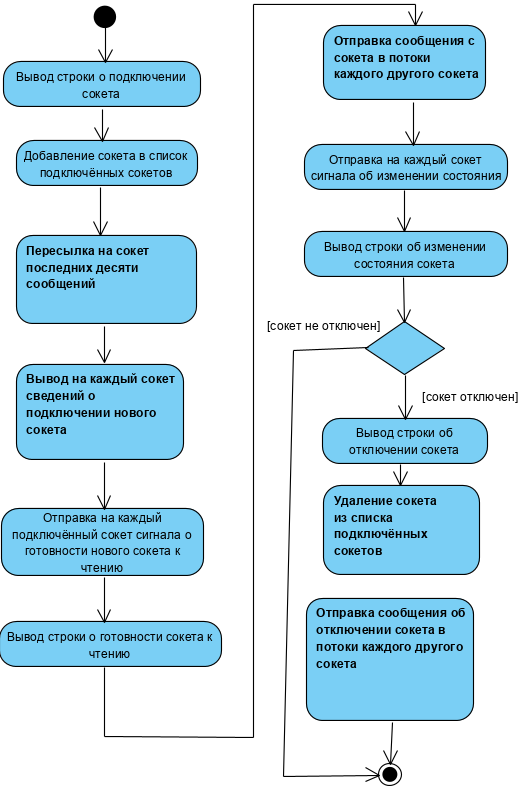
*Рисунок 1 — Диаграмма класса server*

**

*Рисунок 2 — Диаграмма класса socket*

Диаграммы активности:

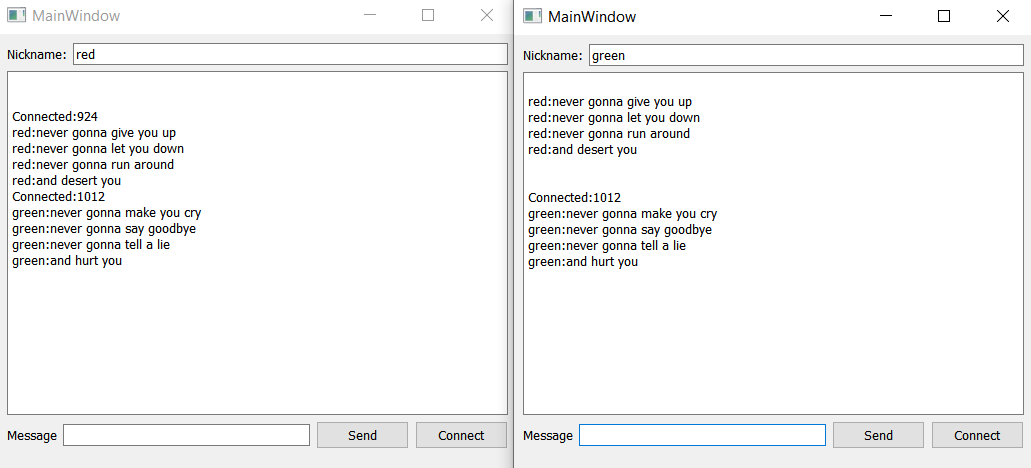
*Рисунок 3 — Диаграмма активности функции on\_send\_clicked*



*Рисунок 4 — Диаграмма активности функции incomingConnections*

1. ЛИСТИНГ ВЫполнения программы

На рисунках 5-6 представлены интерфейсы работы программы.



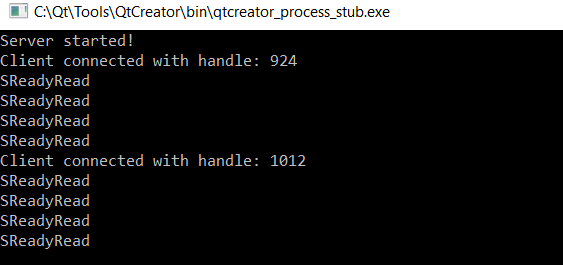
Рисунок 5 – Интерфейс чата

Рисунок 6 – Консоль сервера

1. Вывод

В ходе выполнения лабораторной работы была написана программа, которая реализует работу простейшего чата. При подключении/удалении пользователя всем существующим отправляется сообщение. Последние 10 сообщений сохраняются в файл и восстанавливаются при новом запуске.