

Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
по курсу «Программирование»
на тему «Разработка программы сложной структуры
методом нисходящего программирования»

Выполнил
студент группы 21BO1
Соловьев Д. Д.

Принял
Митрохин М. А.

Пенза 2022

Содержание

Введение	4
1 Постановка задачи.....	6
2 Выбор решения.....	7
3 Описание разработки программы	8
4 Отладка и тестирование.....	9
5 Описание программы.....	10
6 Руководство пользователя	14
Заключение	15
Список использованных источников	16
Приложение А Исходный код программы	17
Приложение А.1 Сервер	18
Приложение А.1.1 Файл «main.cpp»	18
Приложение А.1.2 Файл «server.cpp»	18
Приложение А.1.3 Файл «server.h»	20
Приложение А.2 Клиент	18
Приложение А.2.1 Файл «main.cpp»	21
Приложение А.2.2 Файл «mainwindow.cpp»	21
Приложение А.2.3 Файл «mainwindow.h»	23
Приложение Б Снимки экрана	24

Пояснительная записка состоит из 25 листов, 4 рисунков и 2 приложений.

Введение

Сетевое программирование - широко-направленное развитие программных возможностей. Современный мир почти полностью зависим от того, как передается информация в сетях. Эти технологии используются почти в любой современной технике- от персональных гаджетов до оборонной промышленности. Задача сетевого программиста- установить взаимосвязь между приложениями и устройствами, расположенными друг от друга на расстоянии.

Работа с контроллерами, разного вида подключаемыми устройствами, взаимодействие через интернет, выполнение совместных задач на разных компьютерах. Все это требует методов и способов взаимодействия, а именно- системы обмена данными, которая способна передавать результаты измерения с одного устройства на другое.

Данная структура может использоваться для обмена измерительными данными между компьютерами или получения данных прямо с микроконтроллера измерительного или измеряемого приборов.

В результате, система обеспечит считывание и использование данных в реальном времени с одного устройства на другом, позволит качественно обмениваться и распоряжаться ими. Уменьшит нагрузку на одно устройство и увеличит дальность расположения устройств друг от друга.

В ходе проектирования получены навыки работы с сетью, изучена модель OSI, работа с сокетами и протоколом TCP. Так же, повышены

навыки применения возможностей языка C++, повышен уровень освоения среды QT и ее графических возможностей.

1 Постановка задачи

Система должна представлять из себя как минимум 2 отдельных исполняемых части - это клиентское и серверное приложения. Должен происходить постоянный обмен данными с клиента на сервер, на котором данные будут отправлены обратно для отображения на графическом интерфейсе с выводом временных задержек.

Для создания системы необходимо выбрать используемые сетевые протоколы; возвращать подтверждение, об успешности получения данных, назад на клиентскую часть.

2 Выбор решения

В разработке используется платформа QT Creator версии 7.0.0, в качестве сокетов используются объекты QT базового класса QAbstractSocket. В качестве протокола используется TCP, в качестве отправляемых данных будет время отправки с клиента, оно же будет возвращаться обратно в качестве ответа для выявления времени задержек. Программы будут использовать систему сигналов и слотов QT.

3 Описание разработки программы

Разработка программы включает в себя создание связи между двумя программами, находящимися на разных устройствах и не имеющих друг к другу непосредственный доступ. Обе программы используют класс `QAbstractSocket`, в котором прописываются основные функции обработки и выполнения запросов друг друга, на клиентской части также созданы дополнительные функции для непрерывной отправки сообщений на сервер и автоматического подключения к нему после запуска.

4 Отладка и тестирование

В программах установлен некий контроль над ошибками, проверяется успешность подключения сокетов, проверка на полное получение данных, на успешность отправки данных. диагностика ошибок происходит за счет вывода сообщений с конкретными ошибками с помощью `QDebug()`. Тестирование проводилось на двух разных устройствах, проверялась полнота передачи данных, соединение по локальной и беспроводной сети.

5 Описание программы

С помощью интерфейса сокетов создается объект, который выполняет низкоуровневую часть транспортных протоколов и обеспечивает связь между двумя устройствами по IP, в качестве транспортного протокола используется ТСР. Разработано две программы- клиентская и серверная часть проекта, с помощью сокетов налажен обмен данными между этими приложениями. Клиентское приложение оформлено в виде чат-окна, текстовые данные из которого поступают при отправке на сервер, после чего сервер высылает эти данные в ответ, дополнительно происходит расчет размера отправляемого блока данных с учетом занимаемого места служебной информацией. Ответ сервера клиентское приложение выводит на экран с указанием времени отправки данных(в данном случае текстовых).Сервер может получать данные от нескольких независимых клиентских приложений и использовать эти данные(например, для динамического пользовательского интерфейса).

На рисунке 1 изображен выбранная схема серверной части.

В блоке 2 происходит постоянное прослушивание сокетом портов, ожидая запроса на подключение и даже если подключение происходит, то прослушивание не останавливается и могут подключаться множество запросов(блок 2). Затем, если есть запрос на подключение (блок 3), сокет устанавливает подключение (блок 4), после чего начинается взаимообмен данными между клиентом и сервером до тех пор, пока клиент подключен(блок 5). Если данные

приходят (блок 6), то сервер отправляет ответное сообщение на клиент(блок 8).

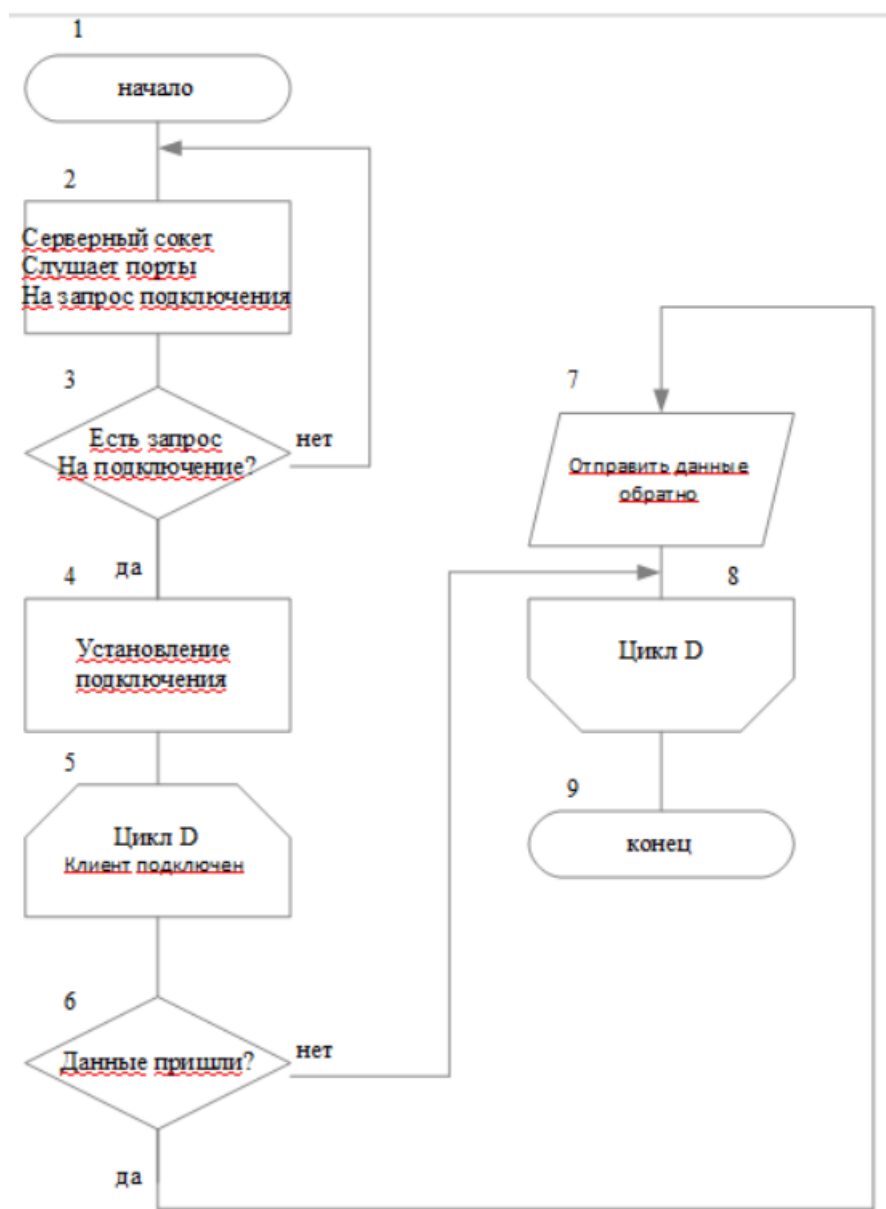


Рисунок 1 – Схема сервера

На рисунке 2 изображен выбранная схема клиентской части.

Сначала клиент отправляет запрос на подключение по указанному пользователем IP(блок 2). После, пока подключение установлено(блок 3), происходит отправка пользовательских

сообщений на сервер(блок4), после, если ответ получен(блок 5), в окно приложения выводится сообщение с данными, которые прислал сервер(блок 6).

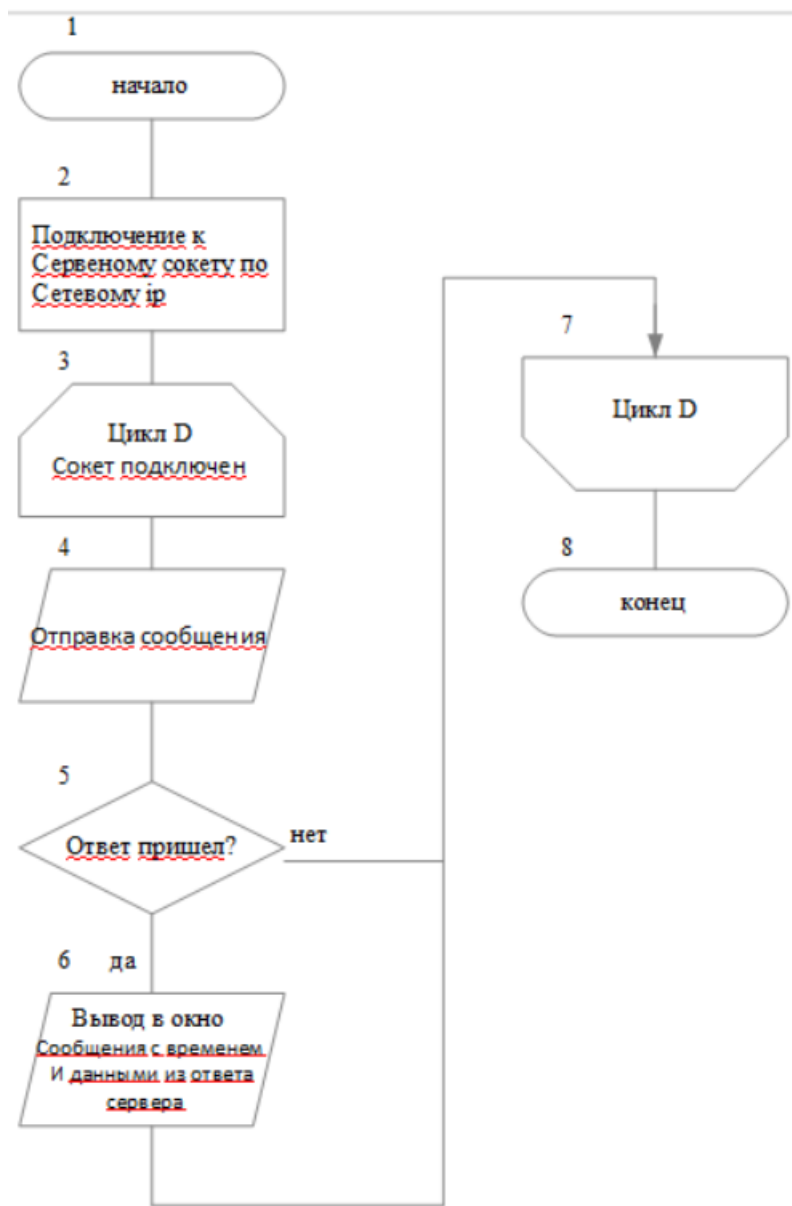


Рисунок 2 –Схема клиента

Алгоритм взаимодействия приложений выстроен следующим образом:

Серверное приложение постоянно находится в режиме прослушивания всех IP адресов. Если со стороны какого-либо адреса

поступает запрос- сервер ставит его в список сокетов в QVector и получает возможность отправлять ответ. Создается поток QDataStream, который используется для чтения потока данных, поступающих от клиента. После успешного получения данных(в нашем случае строки, которая в дальнейшем преобразовывается в целочисленный тип и записывается в файл для использования, выходящего за рамки практики) сервер отправляет в ответ эту же строку чтобы клиент мог проверить целостность поступивших данных.

Клиентское приложение отправляет запрос на ip, указанный в программе. Для теста в примерах использовался локальный ip 127.0.0.1 , для подключения по wi-fi к другому устройству используется публичный адрес, который выдает провайдер сети. Для формирования пакета отправляемых данных используется Поточковый объект QDataStream. Отправляется время и текст, вводимый пользователем в специальное окно. Как только от сервера приходит ответ- считываем его.

6 Руководство пользователя

Каждое приложение должно располагаться на разных устройствах-серверное только на одном компьютере, клиентское - на остальных, но в нашем случае приложения находятся на одном устройстве, т.к. подключение сокетов настроено автоматически для подключения к локальной сети и постоянной отправке данных. Для начала нужно запустить .exe приложение server, после чего, можно запустить client.exe. После этого появится окно на подобии чата(Приложение Б, рисунок 3), в котором начнут отображаться данные времени, возвращенные с сервера и время, которое было во время вывода на экран(Приложение Б, рисунок 4).

Заключение

Изучил модель OSI, познакомился с интерфейсом QT Creator, набрал практического опыта по построению клиент-серверных приложений для работы в локальной сети , а так же изучил работу в сетях, разновидности протоколов и их предназначение, получил опыт работы с разными ресурсами и возможностями языка с++.

Список использованных источников

1. Николай Прохоренок : QT 6 “разработка оконных приложений на C++”.
2. Шлее М. : Qt 5.3. Профессиональное программирование на C++.
3. Жасмин Бланшет,Марк Саммерфилд : программирование GUI на C++.
4. Боровский А. Н. : Qt4.7+. Практическое программирование на C++.
5. Снейдер Й. : Эффективное программирование TCP/IP.
6. Крис Касперски : Самоучитель игры на WINSOCK.

Приложение А

Исходный код программы

Приложение A.1

Сервер

Приложение A.1.1

Файл «main.cpp»

```
#include <QCoreApplication>
#include "server.h"
#include <thread>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    Server s;

    return a.exec();
}
```

Приложение A.1.2

Файл «server.cpp»

```
#include "server.h"
#include <fstream>
#include <QTextStream>
#include <thread>

Server::Server()
{
    if (this->listen(QHostAddress::Any, 2323)) //QHostAddress("178.178.86.48")
    {
        qDebug() << "start";
    }
    else
    {
        qDebug() << "erro";
    }
    nextBlockSize=0;
}

void Server::incomingConnection(qintptr socketDescriptor)
{
    socket = new QTcpSocket;
    socket->setSocketDescriptor(socketDescriptor);
    connect(socket, &QTcpSocket::readyRead, this, &Server::slotReadyRead);
    connect(socket, &QTcpSocket::disconnected, socket, &QTcpSocket::deleteLater);

    Sockets.push_back(socket);
    qDebug() << "client connected" << socketDescriptor;
}

void Server::slotReadyRead()
{
    socket=(QTcpSocket*) sender();
    QDataStream in(socket);
    in.setVersion(QDataStream::Qt_6_2);
    if(in.status()==QDataStream::Ok)
    {

```

```

for(;;)
{
    if(nextBlockSize==0)
    {
        qDebug()<<"nextBlockSize=0";
        if(socket->bytesAvailable()<2)
        {
            qDebug()<<"Data<2, break";
            break;
        }
        in>>nextBlockSize;
        qDebug()<<"nextBlockSize="<<nextBlockSize;
    }
    if(socket->bytesAvailable()<nextBlockSize)
    {
        qDebug()<<"Data not full, break=";
        break;
    }
    QString str;
    in>>str;
    nextBlockSize=0;
    qDebug()<<str;

    SendToClient(str);
    break;
}
}
else
{
    qDebug()<<"DataStream error";
}
}

void Server::SendToClient(QString str)
{
    Data.clear();
    QDataStream out(&Data,QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_6_2);
    out<<quint16(0)<<QTime::currentTime()<<str;
    out.device()->seek(0);
    out<<quint16(Data.size() - sizeof(quint16));
    //socket->write(Data); для одного
    for(int i=0;i<Sockets.size();i++)
    {
        Sockets[i]->write(Data);
    }
}

```

Приложение А.1.3

Файл «server.h»

```
#ifndef SERVER_H
#define SERVER_H
#include<QTcpServer>
#include<QTcpSocket>
#include<QVector>
#include<QTime>
#include <QTextStream>
#include <thread>

class Server : public QTcpServer
{
    Q_OBJECT

public:
    Server();
    QTcpSocket *socket;

private:
    QVector <QTcpSocket*> Sockets; //для подключения нескольких сокетов
    QByteArray Data; //для обмена данных
    void SendToClient(QString str); //метод отправки ответа назад к
клиенту(срабатывает после получения посылки сервером)
    quint16 nextBlockSize; //на тот случай если все данные не смогут
уместиться в одной посылке, подсчитывает размер отправляемых данных

public slots: //сокет динамический, с использованием QT
слотов и сигналов
    void incomingConnection(qintptr socketDescriptor); //метод для сопряжения
сокетов клиента и сервера, устанавливает сообщение между ними, постоянно слушает
порты
    void slotReadyRead(); //там самая объемная
часть кода, после сопряжения клиента и сервера отвечает за получение посылок с
клиента, производит над ними операции, записывает в файл и вызывает метод для
ответа клиенту назад
};

#endif // SERVER_H
```

Приложение A.2

Клиент

Приложение A.2.1

Файл «main.cpp»

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    w.on_pushButton_clicked();
    w.sts();
    return a.exec();
}
```

Приложение A.2.2

Файл «mainwindow.cpp»

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    socket= new QTcpSocket(this);
    connect(socket, &QTcpSocket::readyRead, this, &MainWindow::slotReadyRead);
    connect(socket, &QTcpSocket::disconnected, socket, &QTcpSocket::deleteLater);
    nextBlockSize=0;
}

MainWindow::~MainWindow()
{
    delete ui;
}

QString addresses="127.0.0.1";

void MainWindow::on_pushButton_clicked()
{
    //addresses=ui->lineEdit_2->text();
    socket->connectToHost(QHostAddress(addresses), 2323); // 127.0.0.1
    2a00:1fa1:840d:f557:2dab:2e5a:311e:9d37
}

void MainWindow:: SendToServer(QString str)
{
    Data.clear();
    QDataStream out(&Data, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_6_2);
}
```

```

        str=QDateTime::currentDateTime().toString("zzzz");

out<<quint16(0)<<QDateTime::currentDateTime().toString("ss.zz");//<<str;//поле
сообщения
        out.device()->seek(0);
        out<<quint16(Data.size() - sizeof(quint16));

        socket->write(Data);
        //_sleep(100);
        ui->lineEdit->clear();}

void MainWindow::slotReadyRead()
{
    QDataStream in(socket);
    in.setVersion(QDataStream::Qt_6_2);
    if(in.status()==QDataStream::Ok)
    {
        for(;;)
        {
            if(nextBlockSize==0)
            {
                if(socket->bytesAvailable()<2)
                {
                    break;
                }
                in>>nextBlockSize;
            }
            if(socket->bytesAvailable()<nextBlockSize)
            {
                break;
            }
            QString str;
            QTime time;
            in>>time>>str;
            nextBlockSize=0;
            ui->textBrowser->append(time.toString("ss.zz")+" // "+ str);

        }
    }
    else
    {
        ui->textBrowser->append("read error");
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    SendToServer(ui->lineEdit->text());
}

void MainWindow::on_lineEdit_returnPressed()
{
    SendToServer(ui->lineEdit->text());
}

```

Приложение A.2.3

Файл «mainwindow.h»

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include<QTcpSocket>
#include<QTime>
#include <QtGui>
#include <qobject.h>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:

private:
    Ui::MainWindow *ui;
    QTcpSocket *socket;
    QByteArray Data;
    void SendToServer(QString str);
    quint16 nextBlockSize;
    QTimer *timer;

public slots:
    void sts() {
        timer = new QTimer(this);
        QObject::connect(timer, SIGNAL(timeout()), this,
        SLOT(on_pushButton_2_clicked()));

        timer->start(100); //ms
    }

    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_lineEdit_returnPressed();

    void slotReadyRead();
};
#endif // MAINWINDOW_H
```

Приложение Б

Снимки экрана

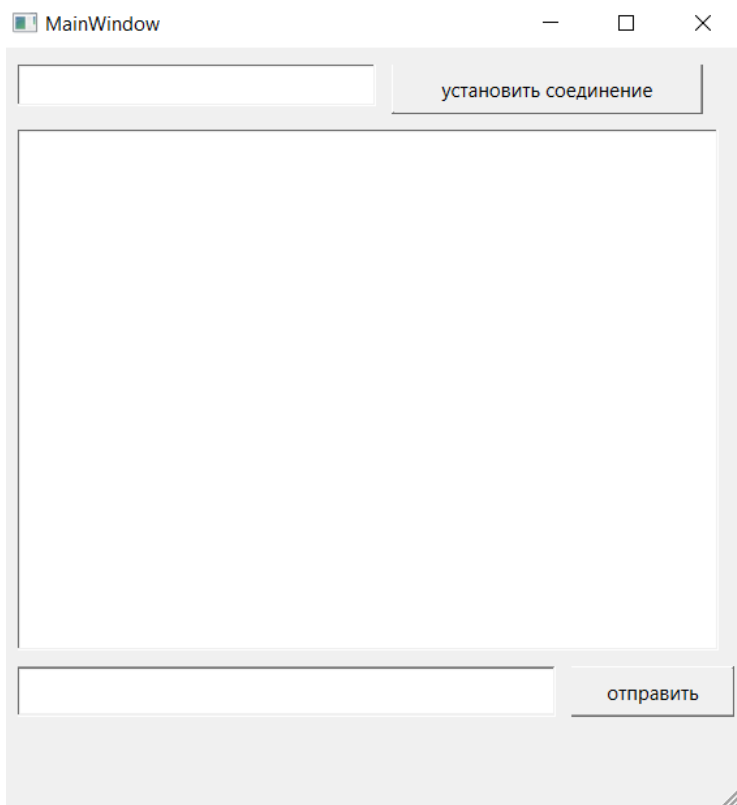


Рисунок 3 – окно чата.

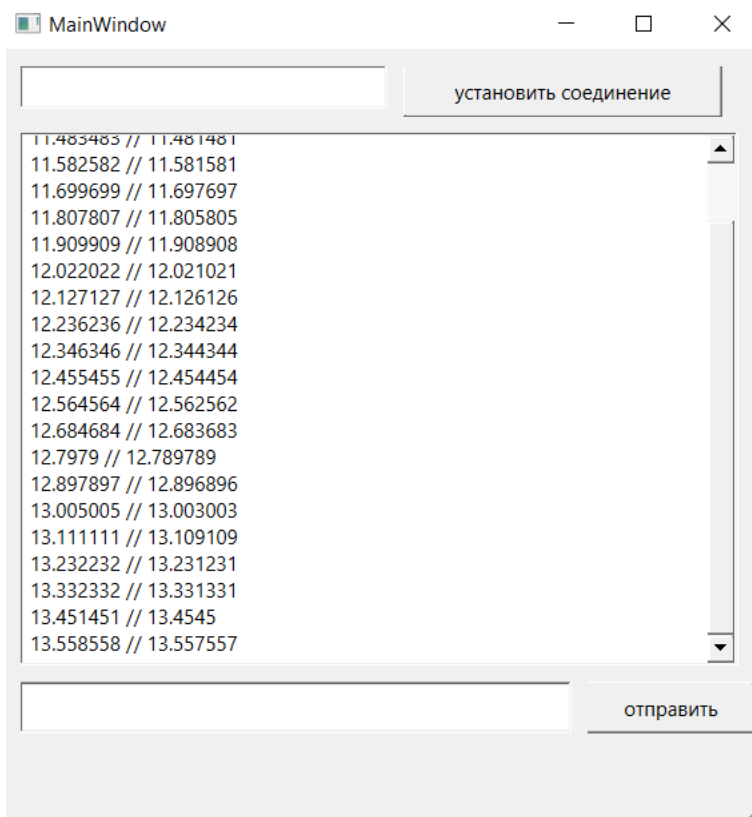


Рисунок 4 – вывод данных

Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
по курсу «Программирование»
на тему «Разработка программы сложной структуры
методом нисходящего программирования»

0-1.
26.12.22
Выполнил
студент группы 21ВО1

Соловьев Д. Д.

Принял

Митрохин М. А.

Пенза 2022

НИЖНИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра «Вычислительная техника»

«УТВЕРЖДАЮ»

Зав. кафедрой ВТ

Митрохин М. А.

«___» _____ 2022 г.

ЗАДАНИЕ
на курсовое проектирование по курсу

Программирование

Студенту Соловьеву Даниилу Денисовичу	Группа 21ВО1
Тема проекта: Разработка программы сложной структуры методом нисходящего программирования	

Исходные данные (технические требования) на проектирование

Наименование программы: Сетевое приложение "чат быстрого обмена".

Задание.

Серверная программа должна:

1. Слушать входящие запросы от клиентских программ.
2. Принимать данные, передаваемые клиентской программой.
3. Отправлять ответ.

Клиентская программа должна:

1. Отправлять запросы серверной программе.
2. Отправлять данные, которые нужно транспортировать на другое устройство.
3. Принимать ответ.

Обязательные требования к программе:

1. Использование графического интерфейса для отслеживания процессов.
2. Операционная система — Windows 10 и выше.
3. Система программирования — QT Creator 7.
4. Язык программирования — C++.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Структура модулей программы, схема алгоритма программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

1. В соответствии с графиком выполнения курсового проекта.

2. _____

3. _____

Дата выдачи задания « 1 » октября 2022

Дата защиты проекта « ____ » _____ 2022

Руководитель _____ Игнатович М.А.

Задание получил _____ « 1 » октября 2022

Студент _____ Соловьев Евгений Викторович