Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

СЕТЕВОЙ ЧАТ

БГУИР КП 1-40 02 01 508 ПЗ

Студент: группы 250505,   
Гнетецкий Д.Г.

Руководитель:

Басак Д.В.

Минск 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc166147552)

[1 ОБЗОР ЛИТЕРАТУРЫ 6](#_Toc166147553)

[1.1 Анализ существующих аналогов 6](#_Toc166147554)

[1.1.1 WhatsApp 6](#_Toc166147555)

[1.1.2 Slack 7](#_Toc166147556)

[1.1.3 Telegram 8](#_Toc166147557)

[1.2. Постановка задачи 8](#_Toc166147558)

[2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 10](#_Toc166147559)

[2.1 Модуль авторизации 10](#_Toc166147560)

[2.2 Модуль пользовательского интерфейса 10](#_Toc166147561)

[2.3 Модуль сервера 10](#_Toc166147562)

[2.4 Модуль протокола общения 10](#_Toc166147563)

[2.5 Модуль чтения и записи данных 10](#_Toc166147564)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 11](#_Toc166147565)

[3.2 Разработка диаграммы классов 11](#_Toc166147566)

[3.3 Описание классов 11](#_Toc166147567)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 12](#_Toc166147568)

[4.1 Разработка алгоритмов 12](#_Toc166147569)

[4.2 Исходный текст программы 13](#_Toc166147570)

[5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЯ 14](#_Toc166147571)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 15](#_Toc166147572)

[ЗАКЛЮЧЕНИЕ 16](#_Toc166147573)

[ЛИТЕРАТУРА 17](#_Toc166147574)

[ПРИЛОЖЕНИЕ А 18](#_Toc166147575)

[ПРИЛОЖЕНИЕ Б 19](#_Toc166147576)

[ПРИЛОЖЕНИЕ В 20](#_Toc166147577)

[ПРИЛОЖЕНИЕ Г 21](#_Toc166147578)





# ВВЕДЕНИЕ

Язык C++ является одним из самых мощных и универсальных языков программирования на сегодняшний день. Он предоставляет разработчикам широкие возможности для создания сложных и высокопроизводительных приложений. Возможности C++ включают в себя объектно-ориентированное программирование, стандартную библиотеку шаблонов (STL), а также обширные средства для многопоточного и системного программирования.

В рамках данного курсового проекта разработан сетевой чат с использованием C++ и фреймворка Qt. Несмотря на кажущуюся простоту, процесс разработки сетевого чата требует определённых навыков, таких как обработка событий, работа с графикой и пользовательским интерфейсом, реализация сетевой логики, использование алгоритмов и структур данных.

Этот проект представляет собой приложение, которое позволяет обмениваться сообщениями в режиме реального времени через сеть. Проект включает в себя два приложения с графическим интерфейсом пользователя (GUI) - сервер и клиент. Клиент может отправлять сообщения серверу и другим клиентам, а сервер может отправлять сообщения клиентам и управлять их соединением.

В рамках данного курсового проекта необходимо ознакомиться с принципами работы сетевого чата и технологией Qt. В процессе разработки предполагается углубить знания по языку C++ и в области объектно-ориентированного программирования, а также осуществить разработку удобного пользовательского интерфейса, используя фреймворк Qt. В конце проекта необходимо протестировать полученный продукт и провести эксперимент на нескольких устройствах.

Данная тема актуальна, так как сетевые чаты являются важной частью многих современных приложений и сервисов, и они широко используются пользователями.

Таким образом, создание сетевого чата отлично подходит для демонстрации навыков программирования на высокоуровневом объектно-ориентированном языке программирования C++.

# 1 ОБЗОР ЛИТЕРАТУРЫ

## Анализ существующих аналогов

Выбор темы курсового проекта был обусловлен желанием углубить знания в области языка программирования C++ и объектно-ориентированного программирования, а также получить опыт в проектировании пользовательских приложений с использованием популярного фреймворка Qt.

Целью данного проекта не является создание конкурентоспособного продукта на рынке мессенджеров, однако для создания функционального и корректно работающего приложения необходимо провести анализ существующих аналогов.

Изучение рынка позволит получить представление о функционале, который предлагают современные мессенджеры, а также оценить их преимущества и недостатки. Проведем анализ основных аналогов на рынке мессенджеров, чтобы получить необходимую информацию для дальнейшей работы над проектом.

### 1.1.1 WhatsApp

WhatsApp — это кроссплатформенный мессенджер, позволяющий обмениваться текстовыми, голосовыми и видеосообщениями, а также файлами различных форматов. Пользователи могут создавать групповые чаты и делиться своим местоположением. WhatsApp использует протоколы шифрования end-to-end, чтобы обеспечить безопасность обмена сообщениями. Однако, в WhatsApp отсутствует возможность управления соединениями со стороны сервера, что является ключевой функцией нашего приложения. На рисунке 1.1 представлен интерфейс WhatsApp:

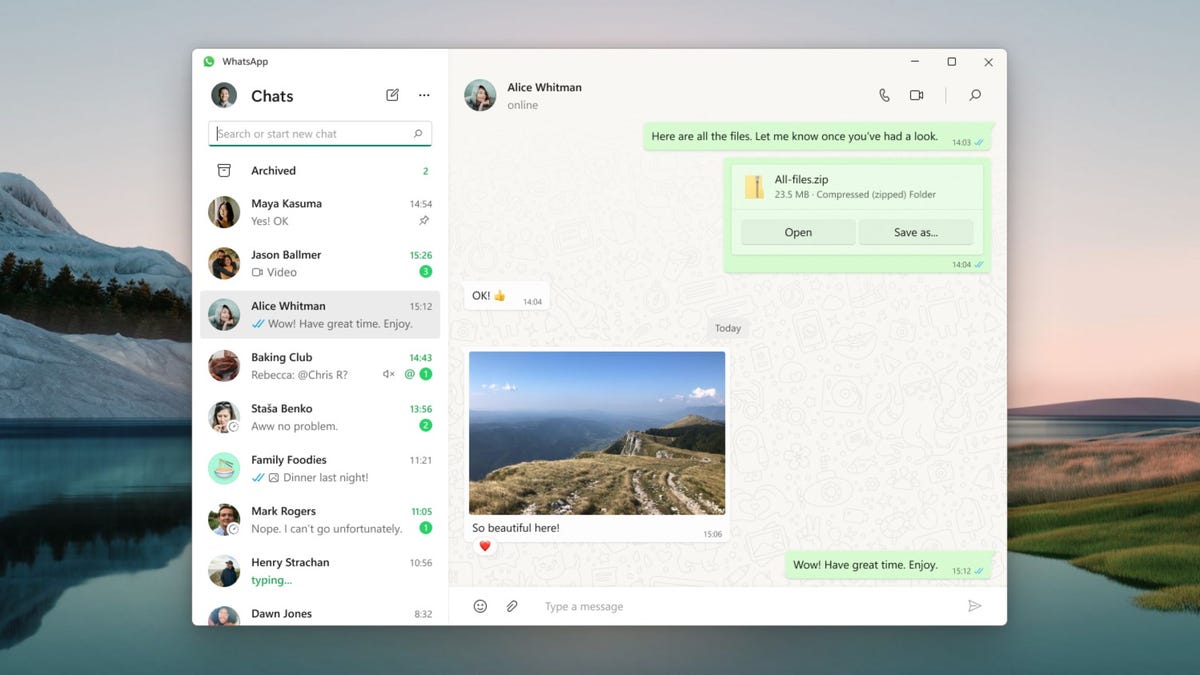


Рисунок 1.1 — Скриншот программы WhatsApp

WhatsApp был разработан компанией WhatsApp Inc. и выпущен в январе 2009 года. Приложение написано на языках Erlang и C++, и поддерживается на всех популярных ОС. WhatsApp активно обновляется и имеет более 2 миллиардов пользователей по всему миру.

### 1.1.2 Slack

Slack — это мощный корпоративный мессенджер, разработанный с учетом потребностей команд различного размера и специализации. Он предлагает обширный набор функций для обмена текстовыми сообщениями, файлами, а также организации аудио- и видеоконференций.

Slack также предлагает интеграцию с большим количеством сторонних сервисов и приложений, что позволяет автоматизировать многие процессы и упростить работу команды.

Интерфейс Slack разделен на несколько основных областей, включая список каналов и прямых сообщений, область обмена сообщениями и панель управления. На рисунке 1.2 представлен интерфейс Slack.

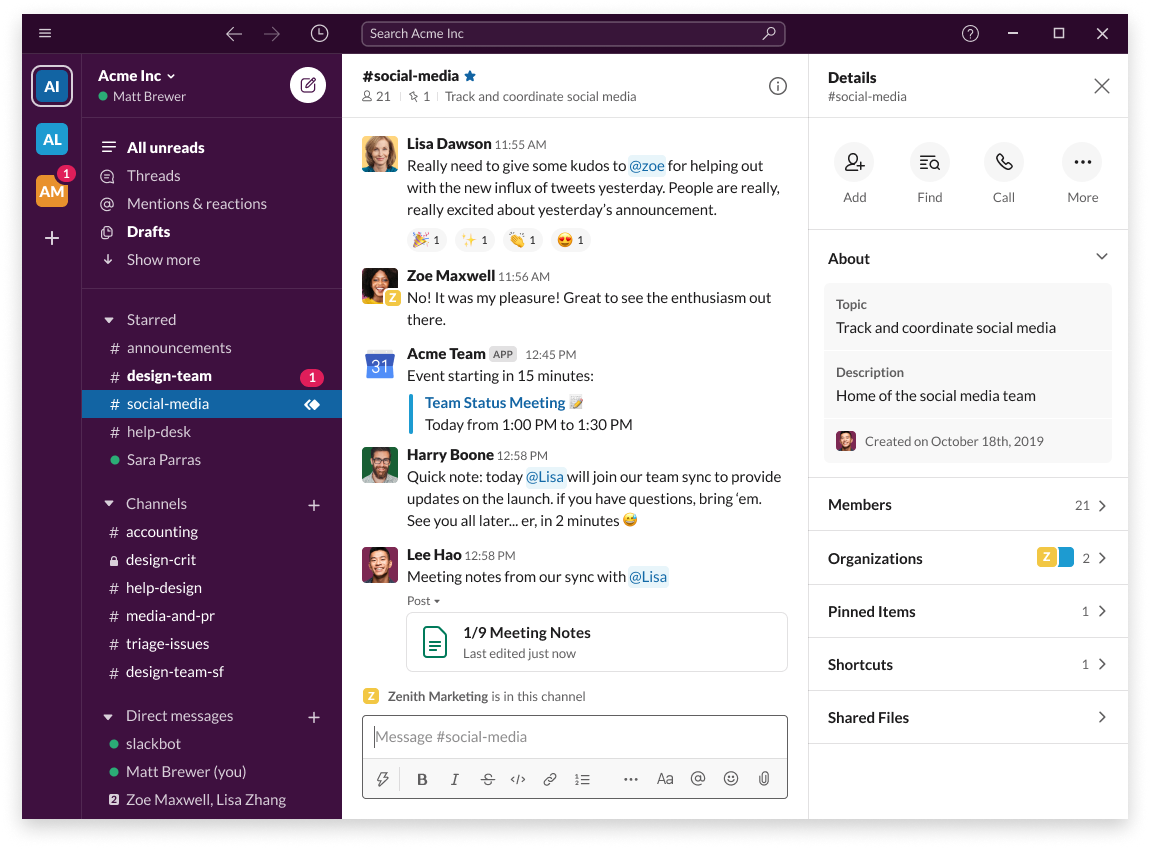


Рисунок 1.2 — Скриншот программы Slack

Slack был разработан компанией Slack Technologies и выпущен в августе 2013 года. Приложение написано на языках Elixir, JavaScript и Objective-C, и поддерживается на всех популярных ОС. Slack активно обновляется и используется многими корпоративными пользователями по всему миру.

### 1.1.3 Telegram

Telegram — это мультиплатформенный мессенджер, обладающий функциями VoIP. Он предоставляет возможность обмена текстовыми, голосовыми и видеосообщениями, стикерами, фотографиями, а также файлами различных форматов. В Telegram можно осуществлять видео- и аудиозвонки, организовывать конференции, а также создавать многопользовательские группы и каналы.

Telegram использует технологию peer-to-peer для обмена сообщениями, что обеспечивает более быструю передачу информации. Основными преимуществами peer-to-peer являются экономия трафика и повышенное качество связи. Однако есть и недостаток — ваш собеседник может узнать ваш IP-адрес. Интерфейс Telegram представлен на рисунке 1.3.

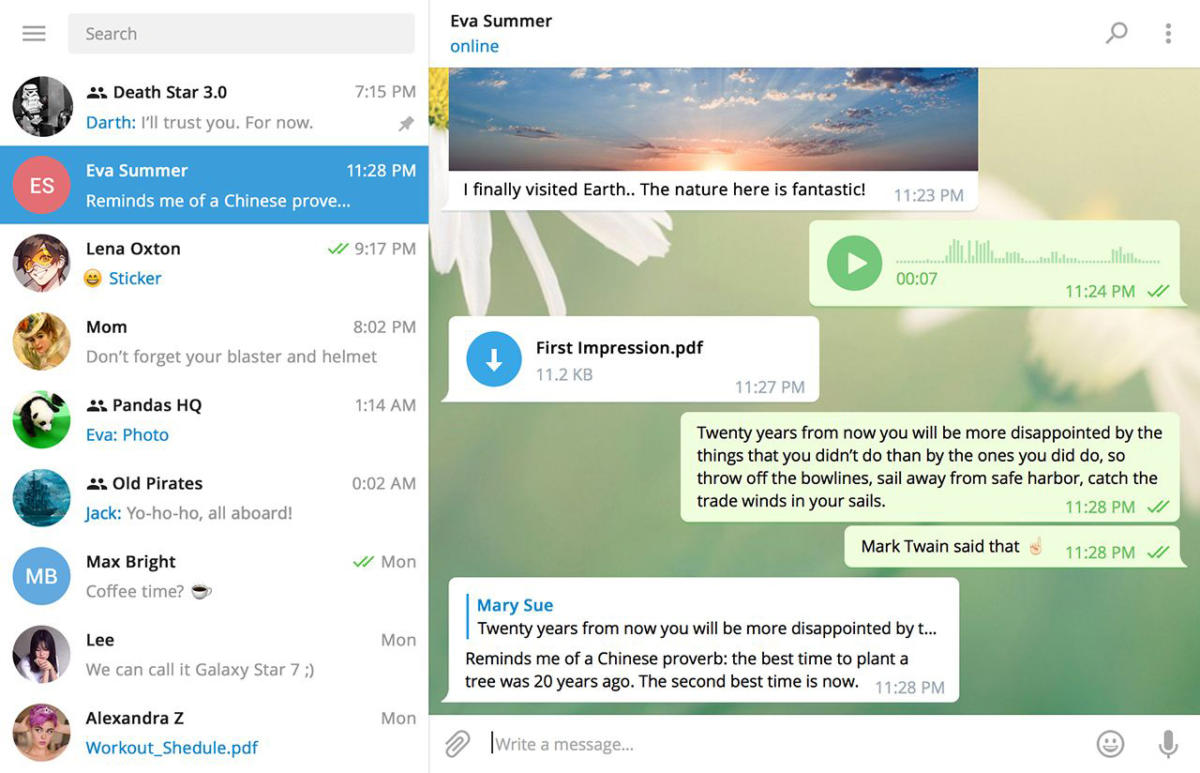


Рисунок 1.3 — Скриншот интерфейса Telegram

Telegram был запущен 14 августа 2013 года. Приложение написано на языках программирования C++ и Java, а пользовательский интерфейс Telegram Desktop реализован с использованием различный фреймворков, включая Qt. Telegram поддерживается на всех распространенных операционных системах и регулярно обновляется.

## 1.2. Постановка задачи

После анализа существующих сетевых чатов, было решено сосредоточиться на ключевых функциях, которые можно реализовать в рамках данного курсового проекта следующий функционал:

- Разработка серверной части, которая будет управлять соединениями и обменом сообщениями между клиентами.

- Разработка клиентской части, которая будет обеспечивать взаимодействие пользователя с чатом.

- Клиентская часть должна иметь интуитивно понятный пользовательский интерфейс с необходимыми элементами управления.

- В программе должна быть предусмотрена возможность хранения истории сообщений.

- Программа должна поддерживать обмен сообщениями в режиме реального времени.

- Программа должна предусматривать возможность авторизации пользователей.

В качестве языка программирования выбран C++, так как он обеспечивает высокую производительность, необходимую для обработки большого количества сообщений, поддерживает объектно-ориентированный подход к программированию и уже знаком по предыдущему опыту.

Для реализации графического интерфейса клиентской части будет использован фреймворк Qt, основанный на языке C++. Qt предлагает обширный набор базовых классов для создания пользовательских классов графического интерфейса, удобную систему взаимодействия между виджетами приложения с помощью системы сигналов и слотов, а также хорошую документацию, что облегчает изучение и использование Qt.

Выбранный набор инструментов позволяет реализовать все поставленные задачи в рамках курсового проекта.

# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

## 2.1 Модуль авторизации

Модуль авторизации будет отвечать за вход пользователя в чат. Он будет собирать имя пользователя и проверять его на уникальность в подключенных пользователей через базу данных.

## 2.2 Модуль пользовательского интерфейса

Модуль пользовательского интерфейса будет обеспечивать основной функционал приложения. Он будет отвечать за отправку и получение сообщений, отображение активных пользователей в чате и выбор пользователя, которому необходимо отправить сообщение. Кроме того, этот модуль будет отвечать за отображение истории сообщений и обработку ввода пользователя.

## 2.3 Модуль сервера

Модуль сервера будет слушать входящие подключения, создавать новую вкладку для каждого клиента, обрабатывать и пересылать сообщения между клиентами. Сервер также сможет отправлять собственные сообщения каждому клиенту, отслеживать активность пользователей и управлять их соединениями, включая возможность принудительного отключения. В случае возникновения ошибок, сервер будет обрабатывать их и предоставлять обратную связь пользователю.

## 2.4 Модуль протокола общения

Модуль протокола общения будет определять формат и структуру данных, используемых для обмена сообщениями между сервером и клиентами. Он будет сериализировать и десериализировать данные, а также определит различные типы сообщений, такие как отправка текста, отправка имени пользователя, индикатор ввода и другие. Этот модуль будет также отвечать за обработку входящих сообщений, извлечение данных из них и предоставление этих данных другим частям приложения.

## 2.5 Модуль чтения и записи данных

Модуль чтения и записи данных будет отвечать за обработку входящих и исходящих сообщений. Он будет считывать введенные пользователем сообщения и отправлять их другим пользователям, а также обрабатывать входящие сообщения и добавлять их в историю чата.

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описывается функционирование и структура разрабатываемого приложения.

## 3.2 Разработка диаграммы классов

Диаграмма классов представлена в приложении А.

## 3.3 Описание классов клиента

### 3.3.1 ChatMessageInfo

Класс наследуется от библиотечного класса QWidget. Данный класс используется для отображения информации о сообщении в чате.

Поля:

— Ui::ChatMessageInfo \*ui — указатель на объект, представляющий пользовательский интерфейс для этого виджета.

— static QMap<QString, QColor> userColorMap — статическая карта, которая хранит соответствие между именами пользователей и цветами, используемыми для отображения их сообщений.

Методы:

— ChatMessageInfo — конструктор, принимает QWidget\* — ссылку на родительский виджет.

— ~ChatMessageInfo — деструктор, удаляет объект ui.

— void displayMessage — принимает const QString& — сообщение, const QString& — имя пользователя, bool — флаг, указывающий, является ли сообщение сообщением от пользователя. Отображает сообщение, устанавливает его цвет и выравнивание.

— QColor getUserColor — принимает const QString& — имя пользователя. Возвращает цвет, который должен быть использован для отображения его сообщений.

— void setMessageAlignment — принимает bool — флаг, указывающий, является ли сообщение сообщением от пользователя. Устанавливает выравнивание сообщения в соответствии с этим флагом.

— void setMessageColor — принимает const QColor& — цвет. Устанавливает его как цвет текста сообщения.

### 3.3.2 ClientManager

Класс наследуется от QObject и используется для управления взаимодействием клиента с сервером.

Поля:

— QTcpSocket \*socket — сокет для подключения к серверу.

— QHostAddress ip — IP-адрес сервера.

— ushort port — порт сервера.

— ClientProtocol protocol — объект протокола клиента для обработки сообщений.

Методы:

— ClientManager — конструктор, принимает QHostAddress — IP-адрес сервера, ushort — порт сервера, QObject\* — ссылку на родительский объект.

— void connectToServer — подключается к серверу.

— void composeAndSendMessage — принимает QString — сообщение, QString — получатель. Составляет и отправляет сообщение.

— void composeAndSendName — принимает QString — имя. Составляет и отправляет сообщение с именем.

— void disconnectFromServer — отключается от сервера.

— void readyRead — обрабатывает входящие данные от сервера.

— void setupClient — настраивает клиентский сокет и соединения.

Сигналы:

— void connected — сигнал о подключении к серверу.

— void disconnected — сигнал об отключении от сервера.

— void chatMessageReceived — сигнал о получении сообщения в чате.

— void newNameReceived — сигнал о получении нового имени.

— void connectionAcknowledged — сигнал о подтверждении подключения.

— void newClientConnectedToServer — сигнал о подключении нового клиента к серверу.

— void clientNameUpdated — сигнал об обновлении имени клиента.

— void clientDisconnected — сигнал об отключении клиента.

— void errorOccurred — сигнал о возникновении ошибки.

### 3.3.3 ClientProtocol

Класс используется для обработки сообщений между клиентом и сервером.

Поля:

— MessageType messageType — тип сообщения.

— QString chatMessage — текст сообщения чата.

— QString newName — новое имя.

— QString messageReceiver — получатель сообщения.

— QString currentClientName — текущее имя клиента.

— QString previousName — предыдущее имя.

— QStringList clientNames — список имен клиентов.

— QString myName — мое имя.

— QString messageSender — отправитель сообщения.

Методы:

— ClientProtocol — конструктор.

— QByteArray composeTextMessage — принимает QString — сообщение, QString — получатель. Составляет и возвращает сообщение чата.

— QByteArray composeNameMessage — принимает QString — имя. Составляет и возвращает сообщение с именем.

— void parseData — принимает QByteArray — данные. Разбирает данные.

— const QString &getChatMessage — возвращает текст сообщения.

— const QString &getNewName — возвращает новое имя.

— MessageType getMessageType — возвращает тип сообщения.

— QString getMessageReceiver — возвращает получателя сообщения.

— const QString &getCurrentClientName — возвращает текущее имя клиента.

— const QString &getPreviousName — возвращает предыдущее имя.

— const QStringList &getClientNames — возвращает список имен клиентов.

— const QString &getMyName — возвращает мое имя.

— QString getMessageSender — возвращает отправителя сообщения.

— QByteArray prepareData — принимает MessageType — тип, QString — данные. Подготавливает и возвращает данные.

### 3.3.4 ClientWindow

Класс представляет собой главное окно клиента.

Поля:

— Ui::ClientWindow \*ui — интерфейс пользователя.

— ClientManager \*client — менеджер клиента.

— LoginWindow \*loginWindow — окно входа.

Методы:

— ClientWindow — конструктор.

— ~ClientWindow — деструктор.

— void connectToServer — подключается к серверу.

— void setupClient — настраивает клиента.

— void createMessage — принимает QString — имя пользователя, QString — сообщение, bool — является ли сообщение моим. Создает сообщение.

— void processMessageAndSend — обрабатывает и отправляет сообщение.

— void on\_btnSend\_clicked — обработчик нажатия кнопки "Отправить".

— void receiveChatMessage — принимает QString — отправитель, QString — сообщение. Получает сообщение чата.

— void onConnectionAcknowledgement — принимает QString — мое имя, QStringList — имена клиентов. Обрабатывает подтверждение подключения.

— void onNewClientConnectedToServer — принимает QString — имя клиента. Обрабатывает подключение нового клиента к серверу.

— void onClientNameUpdated — принимает QString — предыдущее имя, QString — имя клиента. Обрабатывает обновление имени клиента.

— void onClientDisconnected — принимает QString — имя клиента. Обрабатывает отключение клиента.

— void on\_nameEdit\_returnPressed — обработчик нажатия кнопки "Ввод" в поле редактирования имени.

— void on\_editMessage\_returnPressed — обработчик нажатия кнопки "Ввод" в поле редактирования сообщения.

### 3.3.5 LoginWindow

Класс представляет собой окно входа.

Поля:

— Ui::LoginWindow \*ui — интерфейс пользователя.

— QSqlDatabase db — база данных.

— QString nickName — никнейм.

Методы:

— LoginWindow — конструктор.

— ~LoginWindow — деструктор.

— QString getNickname — возвращает никнейм.

— void removeNickname — принимает const QString &nickname. Удаляет никнейм.

— bool doesNicknameExist — принимает const QString &nickname. Проверяет, существует ли никнейм.

— void addNickname — принимает const QString &nickname. Добавляет никнейм.

— void setNickName — принимает const QString &newNickName. Устанавливает никнейм.

— void setupDatabase — настраивает базу данных.

— void on\_nickname\_returnPressed — обработчик нажатия кнопки "Ввод" в поле никнейма.

# 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## 4.1 Разработка алгоритмов

Метод swap(List& left, List& right) класса List (обмен содержимым между двуми списками):

Шаг 1: Начало.

Шаг 2:

Входные данные:

List& left, List& right – объекты которые надо поменять местами.

Шаг 3:Объявление переменных:

FakeListNode\* tmpl – предыдущий узел правого списка. FakeListNode\* tmpr – следующий узел правого списка.

Шаг 4: Если свойство propagate\_on\_container\_swap::value равно true, обмен владеющими указателями allocator\_ списков left и right.

Шаг 5: Временные узлы сохраняют указатели на предыдущий (tmpl) и следующий (tmpr) узел из списка right.

Шаг 6: Проверка, если размер списка left не равен 0, то узлы между left.fake\_.prev и left.fake\_.next вставляются после узла &right.fake с помощью метода insert\_nodes(), иначе – устанавливается связь между узлами &right.fake\_с помощью метода connect\_nodes.

Шаг 7: Если размер списка right не равен 0, то узлы между tmpl и tmpr вставляются после узла &left.fake\_c помощью метода insert\_nodes, иначе – устанавливается связь между узлами &left.fake\_с помощью метода connest\_nodes().

Шаг 8: Обмен размерами списков c пощью std::swap().

Шаг 9: Конец.

Метод inputLogic(int charTyped) класса Text (обработка ввода пользователя в объекте класса Text):

Шаг 1: Начало.

Шаг 2: Входные данные: int charTyped – символ, введенный пользователем.

Шаг 3: Если введенный символ не является клавишами Backspace или Enter, то добавить его к полю txt класса Text, иначе переход на шаг 4.

Шаг 4: Если введенный символ является клавишей Backspace, то переходим к шагу 5, иначе переход на шаг 6.

Шаг 5: Если длина больше 0, удаляем последний символ с помощью метода deleteLastChar(), иначе переходим на шаг 6, т. к строка пустая.

Шаг 6: Устанавливаем строку текста, добавив символ курсора ("|") в конец (text.setString(txt.str() + "|").

Шаг 7: Конец.

## 4.2 Исходный текст программы

Код программы представлен в приложении Г.

# 5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЯ

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были выполнены первоначально заданные цели, результатом которых является игра «Тетрис» с графическим интерфейсом. Были также реализованы различные алгоритмы, включая проверку и удаление заполненных линий, обработку ввода пользователя и генерацию новых фигур, продолжение прошлой игры.

Игровые приложения, такие как «Тетрис», являются прекрасной демонстрацией возможностей информационных систем в области развлечений и отдыха. Они обеспечивают удобное и увлекательное средство для прохождения времени, обучения или увлечения. «Тетрис», в частности, заставляет игроков мыслить стратегически и быстро принимать решения, стимулируя таким образом развитие когнитивных навыков.

Есть возможность сохранения игрового прогресса пользователя, что дает игроку возможность отслеживать свой прогресс и ставить перед собой новые цели.

Программа может быть улучшена в дальнейшем:

Можно добавить различные уровни сложности: обычно в играх «Тетрис» уровни сложности нарастают по мере продвижения игрока. Можно предложить разнообразные варианты сложности или даже интересные "тематические" уровни.

Добавление соревновательного режима также положительно повлияет на игру, так как игроки смогут сражаться друг с другом в реальном времени.

Настройки пользовательского интерфейса предоставим игрокам возможности настроить внешний вид и управление игры также может улучшить пользовательский опыт.

# ЛИТЕРАТУРА

1. Страуструп, Б. Программирование. Принципы и практика использования C++ / Б. Страуструп. – М. : Вильямс, 2011. – 1225 с.
2. Документация Qt [Электронный ресурс]. – Электронные данные. – Режим доступа: https://doc.qt.io/.
3. Стандарты C++ [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://isocpp.org/std/the-standard>
4. Вычислительные машины, системы и сети: дипломное проектирование [Электронный ресурс]. – Минск БГУИР 2019. - Режим доступа: <https://www.bsuir.by/m/12_100229_1_136308.pdf>
5. Лафоре, Р. Объектно-ориентированное программирование в C++, 4-е издание / Р. Лафоре – СПб.: Питер, 2004.
6. Форум разработчиков на C++ [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.cplusplus.com/forum/.
7. Шлее, М. Qt 5.10. Профессиональное программирование на C++ / М. Шлее. – СПб. : БХВ-Петербург, 2018. – 896 с.
8. Форум разработчиков на Qt [Электронный ресурс]. – Электронные данные. – Режим доступа: https://forum.qt.io/.

# ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

# 

# ПРИЛОЖЕНИЕ Б

(обязательное)

Схема алгоритма

# ПРИЛОЖЕНИЕ В

(обязательное)

Схема алгоритма

# ПРИЛОЖЕНИЕ Г

(обязательное)

Исходный текст программы