|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУК «Информатика и управление»\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_ ***ИУК5 «Системы обработки информации»***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

***Разработка Peer-to-peer чата при помощи Windows API***

по дисциплине ***Системное программирование***

Студент гр.\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2021

Калужский филиал   
федерального государственного бюджетного образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»   
(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУК5\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_(Е.В. Вершинин)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине ***Системное программирование***

Студент ***Мицих Владислав Павлович ИУК5-42Б*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, инициалы, индекс группы)

Руководитель ***Фролов Павел Валерьевич*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, инициалы)

График выполнения проекта: 25% к\_4\_нед., 50% к\_7\_нед., 75% к\_10\_нед., 100% к\_14\_нед.

***1. Тема курсового проекта***

***Разработка Peer-to-peer чата при помощи Windows API***

***2. Техническое задание***

*Разработать приложение с использованием функций Windows API для обмена сообщениями между пользователями, на основе одноранговой (пиринговой) сети*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***3. Оформление курсового проекта***

3.1. Расчетно-пояснительная записка на\_\_\_\_\_\_\_\_ листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О.)

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**СОДЕРЖАНИЕ**

1. Техническое задание…………………………………………………………...5-9
   1. Наименование……………………………………………………………...5
   2. Основание для разработки………………………………………………...5
   3. Исполнитель………………………………………………………………..6
   4. Цель разработки……………………………………………………………6
   5. Содержание работы………………………………………………………..7
      1. Задачи, подлежащие решению………………………………………...7
      2. Требования к архитектуре АСОИ……………………………………...7
      3. Требования к составу программных компонентов…………………...7
      4. Требования к прикладным программам………………………………7
      5. Требования к входным/выходным данным…………………………...8
      6. Требования к временных характеристикам…………………………..8
      7. Требования к составу технических средств…………………………..8
   6. Этапы разработки……………………..…………………………………...9
   7. Техническая документация, предъявляемая по окончании работы……9
   8. Дополнительные условия……………………..…………………………..9
2. Научно-исследовательская часть…………………………………………...10-19
   1. Постановка задачи проектирования……………………………………..10
   2. Описание предметной области…………………………………………..10
   3. Анализ аналогов и прототипов……………………..……………………12
   4. Перечень задач, подлежащих решению в процессе разработки……….17
   5. Обоснование инструментов и платформы для разработки……............18
3. Проектно-конструкторская часть…………………………………………..20-32
   1. Разработка структуры приложения……………………………………...20
   2. Используемые функции Win32API……………………………………...20
   3. Разработка архитектуры приложения…………………………………...25
   4. Разработка алгоритмов приложения…………………………………….26
   5. Разработка интерфейса взаимодействия пользователя системой…………………………………………………………………..31
4. Проектно-технологическая часть…………………………………………..33-36
   1. Разработка руководства пользователя………….……………………….33
   2. Установка и запуск.… …………………………………………………….35

5. Заключение…………………………………..…………………………………..39

6. Список использованной литературы…………………………………………..40

# ТЕХНИЧЕСКОТЕХНИЧЕСКОЕ ЗАДАНИЕ

**1.1. Наименование**

Peer-to-peer чат на основе Windows API «P2P chat».

**1.2. Основание для разработки**

В основе технологии лежит принцип децентрализации: все узлы в сети P2P равноправны, что обеспечивает такие преимущества технологии P2P перед клиент-серверным подходом, как отказоустойчивость при потере связи с несколькими узлами сети, увеличение скорости получения данных за счет копирования одновременно из нескольких источников, возможность разделения ресурсов без “привязки” к конкретным IP-адресам.

Достоинства и недостатки P2P:

1) Связность пользователей - P2P основано на прямом взаимодействии конечных пользователей. Но P2P не может обеспечить гарантированное качество предоставляемых ресурсов, в связи с высокой дифференциацией пропускной способности подключенных узлов.

2) Безопасность - владелец одного узла, чаще всего, не знаком с владельцами других узлов в P2P сети. Узлы в P2P-сети, скорее всего, принадлежат отельными пользователям, а не организациям. Предоставление ресурсов в P2P сети обычно происходит без предварительной договоренности между узлами; Большинство узлов в P2P сети, в большинстве случаев, действуют исключительно в своих интересах и не заинтересованы в предоставлении ресурсов своих систем кому-либо.

3) Масштабируемость - P2P сети ориентированы на большое количество подключенных узлов. Таким образом, должны быть решены вопросы масштабируемости предоставляемых ресурсов. При увеличении количества узлов возникают вопросы с работой как P2P сети с единым индексом (растет нагрузка на сервер индекса), так и широковещательных сетей (многократно растет загрузка сети «широковещательными» запросами).

Задачи P2P сетей: уменьшение затрат; объединение ресурсов; повышенная масштабируемость; анонимность.

**1.3. Исполнитель**

Студент группы ИУК5-42Б Мицих Владислав Павлович

**1.4. Цель разработки**

Целью курсовой работы является формирование практических навыков по разработке и реализации программного приложения с использованием интерфейса прикладного программирования (АРI) операционных систем.

Задачи проектирования:

1. овладение первичными навыками ведения научно-исследовательской, проектной и производственно-технологической деятельности, развитие творческих способностей индивидуально для каждого студента;
2. подготовка к выполнению выпускной квалификационной работы;
3. усвоение методов грамотного ведения, оформления и редактирования технической документации.

Целью разработки является создание однорангового (peer-to-peer) чата, который позволяет отдельным элементам сети взаимодействовать без помощи серверов.

**1.5. Содержание работы**

**1.5.1. Задачи, подлежащие решению:**

исследование существующих протоколов передачи данных, а также изучение работы сокетов;

изучение принципов системного программирования и работы WinAPI функций;

реализация децентрализованного peer-to-peer чата на основе WinAPI;

1. тестирование разработанного программного обеспечения;
2. подготовка расчетно-пояснительной записки и графических листов;
3. подготовка презентации и речи для защиты курсовой работы;
4. защита курсовой работы.

**1.5.2. Требования к архитектуре АСОИ**

К архитектуре предъявляются следующие требования:

тип приложения – оконное;

децентрализованная одноранговая архитектура взаимодействия между клиентами чата;

 отсутствие выделенных серверов

**1.5.3. Требования к составу программных компонентов**

Программный комплекс должен состоять из следующих программных компонентов:

1. исполняемый файл PE формата с расширением .exe, реализующий клиентскую часть децентрализованного Peer-to-peer чата.

**1.5.4. Требования к прикладным программам**

Для работы программного комплекса необходимы:

1. Microsoft Windows;
2. Microsoft Visual Studio;
3. минимальный набор драйверов, обеспечивающих   
   работоспособность ПК.

**1.5.5. Требования к входным/выходным данным**

Входные данные:

1. собственный порт в диапазоне от 1024 до 49151;
2. порт собеседника в диапазоне от 1024 до 49151;
3. IP – адрес версии IPv4;
4. текстовые сообщения пользователей;
5. сохранение данных в бинарный файл.

Выходные данные:

1. всплывающее окно типа «Контакт», «Сообщения»;
2. список контактов в текстовом формате для экспорта данных.

**1.5.6. Требования к временным характеристикам**

Требования к временным характеристикам программы не предъявляются.

**1.5.7. Требования к составу технических средств**

Для функционирования системы необходимы:

* процессор: 1 ГГц и быстрее;
* сетевой адаптер;
* RAM: 1 Гбайт (32 бит) или 2 Гбайт (64 бит);
* HDD: 500 Мбайт (32 бит) или 700 Мбайт (64 бит);
* видеокарта: поддержка Microsoft DirectX 9 с драйвером WDDM;
* монитор;
* мышь;
* клавиатура.

**1.6. Этапы разработки**

реализация сетевого протокола взаимодействия клиентов децентрализованного Peer-to-peer чата;

создание пользовательского интерфейса с помощью WinAPI;

1. тестирование реализованной технологии;
2. исправление выявленных ошибок.

**1.7. Техническая документация, предъявляемая по окончании работы**

По окончанию работы предъявлена расчетно-пояснительная записка в состав которой входят:

* техническое задание;
* научно-исследовательская часть;
* проектно-конструкторская часть;
* проектно-технологическая часть.

Также должна быть предоставлена графическая часть работы, выполненная формате А1 на 2 листах, в которую входят:

* демонстрационные чертежи;
* алгоритмические схемы.

**1.8. Дополнительные условия**

язык программирования С++;

использование Windows API функций;

1. интерфейс должен предоставлять список контактов пользователя, окно для ввода и получения сообщений, форма настройками клиента.

**Научно-исследовательская часть  
  
2.1. Постановка задачи проектирования**Задача курсовой работы состоит в разработке децентрализованного peer-to-peer чата на основе Windows API, где сообщения передаются с одного компьютера на другой.

**2.2. Описание предметной области**

Чат – средство обмена сообщениями в реальном времени с помощью компьютера, смартфона, планшета в основном через интернет. Чат появился в результате развития информационных технологий. Ранней альтернативой чату был телефон, который позволял общаться в режиме реального времени.

Существует несколько разновидностей программной реализации чатов:

* HTTP или веб-чаты. Такой чат выглядит как обычная веб-страница, где можно прочесть последние несколько десятков фраз, написанные участниками чата и модераторами. Страница чата автоматически обновляется с заданной периодичностью.
* Чаты, использующие технологию Adobe Flash. Вместо периодической перезагрузки страницы между клиентом и сервером открывается сокет, что позволяет моментально отправлять или получать сообщения, расходуя меньше трафика.
* IRC, специализированный протокол для чатов.
* Программы-чаты для общения в локальных сетях (например, Vypress Chat, Intranet Chat, Pichat). Часто есть возможность передачи файлов.
* Чаты, реализованные поверх сторонних протоколов (например, чат, использующий ICQ).
* Чаты, работающие по схеме клиент-сервер, это позволяет использовать их в сетях со сложной конфигурацией, а также управлять клиентскими приложениями (например, Mychat, Jabber)
* Чаты, работающие в одноранговых сетях. У них нет потребности в отдельном сервере, они часто используют возможности технологий DHT и TCP Relay (пример: Tox)
* Чаты, использующие технологию Push. Вместо периодической отправки запросов серверу о новых сообщениях, используются входящие сообщения от сервера, что позволяет отправлять и получать сообщения, расходуя меньше трафика (например, WinGeoChat)
* Полностью анонимные чаты. В них собеседник не предполагает с кем общается и при каждом новом соединении общается с новым человеком. (например, ChatVdvoem (ЧатВдвоем))

Идея peer-to-peer общения заключается в том, что каждый peer знает и поддерживает информацию о других участниках. Когда новый клиент подключается к сети, он может узнать у любого пира информацию о том, где и какие файлы сейчас доступны. Когда клиент начинает скачивает файл себе на компьютер, то скачанные части этот файла сразу становятся доступны для скачивания другим пользователям. Никто не даёт гарантию, что каждый клиент будет находиться длительное время в сети и давать скачивать информацию, напротив -- ситуация, когда клиент пропадает в процессе загрузки, является естественной. В данном случае будет найден новый клиент, который может продолжить передачу данных. Для поддержания списка активных peer-ов каждый клиент посылает другим клиентам heartbeat. Heartbeat - это сообщение, которое один сервер посылает другому, чтобы сказать ему, что он жив. Соответственно, если heartbeat долго не приходит, значит этот сервер нужно удалить из списка активных peer-ов. Постоянно обмениваться heartbeat-ом с большим количеством серверов трудоёмко. Поэтому у каждого сервера есть два параметра - нижняя и верхняя граница на размер списка активных клиент. Когда это количество становится ниже нижней границы, запускается поиск новых участников. клиент запрашивает у других клиент список активных peer-ов.

**2.3. Анализ аналогов и прототипов**

**RetroShare** – известное, свободное кроссплатформенное программное обеспечение для бессерверного обмена письмами, мгновенными сообщениями, построенное на P2P.

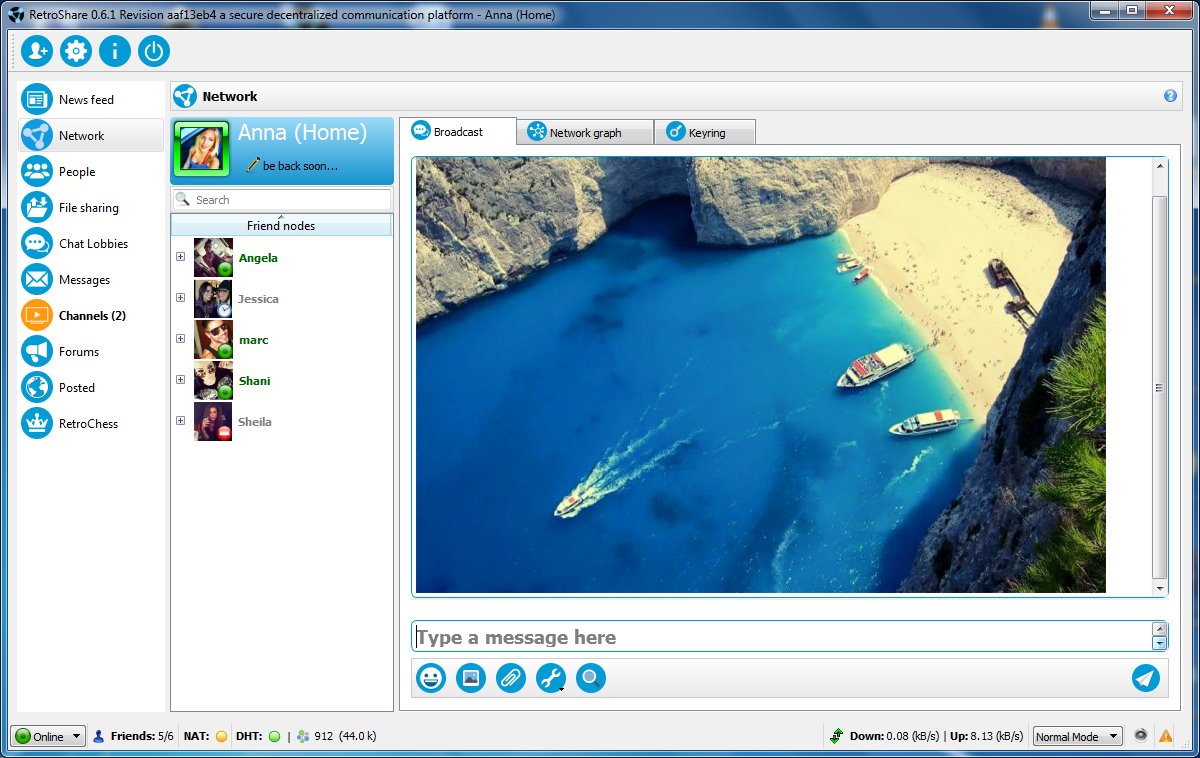


Рисунок 1 - Интерфейс программы «RetroShare»

Система позволяет организовывать обособленные бессерверные Friend-to-Friend сети или одну глобальную сеть с определённым кругом лиц — например, друзьями, семьёй, коллегами. Соединения устанавливаются напрямую только с теми участниками, которым дали разрешение. Это важная особенность, перемещающая подобные сети в рамки особого класса децентрализованных сетей, в которых несанкционированное подключение к вашему компьютеру запрещено.

Обмен сообщениями возможен от имени анонимных личностей, так и непосредственно между сетевыми узлами. В первом случае доступна оффлайн доставка, применяется хранение сообщений на узлах общих друзей.

Система анонимных личностей также предоставляет возможность общения с людьми за пределами круга друзей, сообщения передаются по цепочке из последовательно соединенных узлов. Таким образом возможна связь с любым участником сети, в том числе с тем, с которым соединение не установлено.

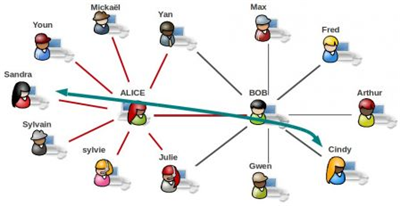
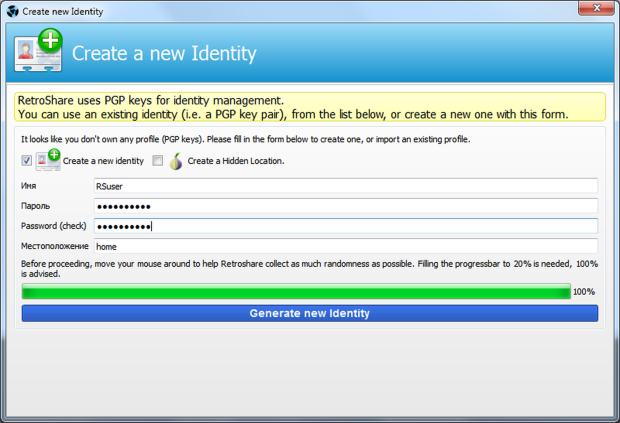


Рисунок 2 - Пример общения людей за пределом круга друзей

Процедура подключения к сети RetroShare включает в себя два этапа: создание личного сертификата и поиск доверенных пиров.

При первом запуске на экран выводится окно, помогающее пользователю создать личный сертификат.

Рисунок 3 - Создание личного сертификата



Доверенных пиров можно найти в чате внутри. К чату можно подключиться, обменявшись сертификатом с роботом. Следует отметить, что взаимный обмен сертификатами с одним или обоими роботами лишь позволяет быстро загрузить основные публичные чаты и не даёт возможности пользоваться ресурсами сети. Это значит, что вы не сможете производить поиск и просмотр контента. Для этого пользователь должен найти как минимум одного участника сети, готового осуществить взаимный обмен сертификатами. Чат полезен в случае, если требуется наладить временный приватный контакт с человеком или произвести обмен сертификатами, когда передача сертификатов через общие чат-комнаты нежелательна. Удалённый контакт реализуется через систему анонимных туннелей и возможен лишь в том случае, когда оба участника уже вошли в сеть.

Для того, чтобы начать общение необходимо, чтобы кто-то откликнулся на просьбу и дал ссылку на собственный сертификат. Пользователь добавляет доверенного пира, а также вводите пароль к сертификату, созданному ранее.

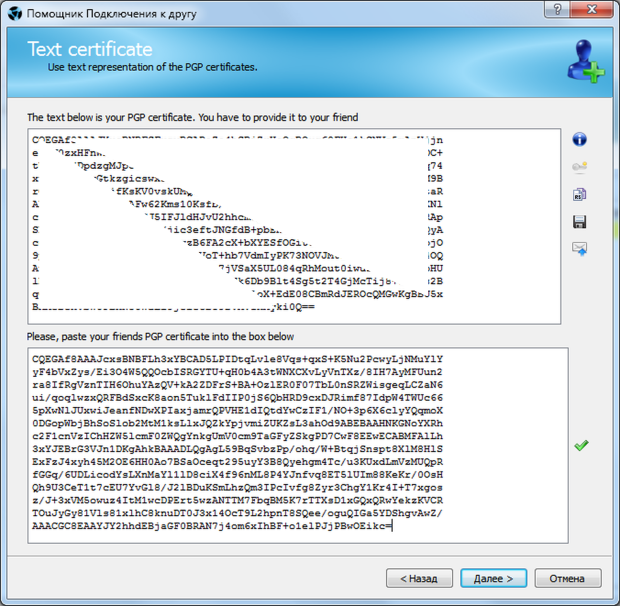


Рисунок 4 - Для подключения к роботу помещаем его сертификат в нижнее окно

Появляется окно прогресса подключения. Можно нажать «OK», и подключение к пиру будет проходить в фоновом режиме.

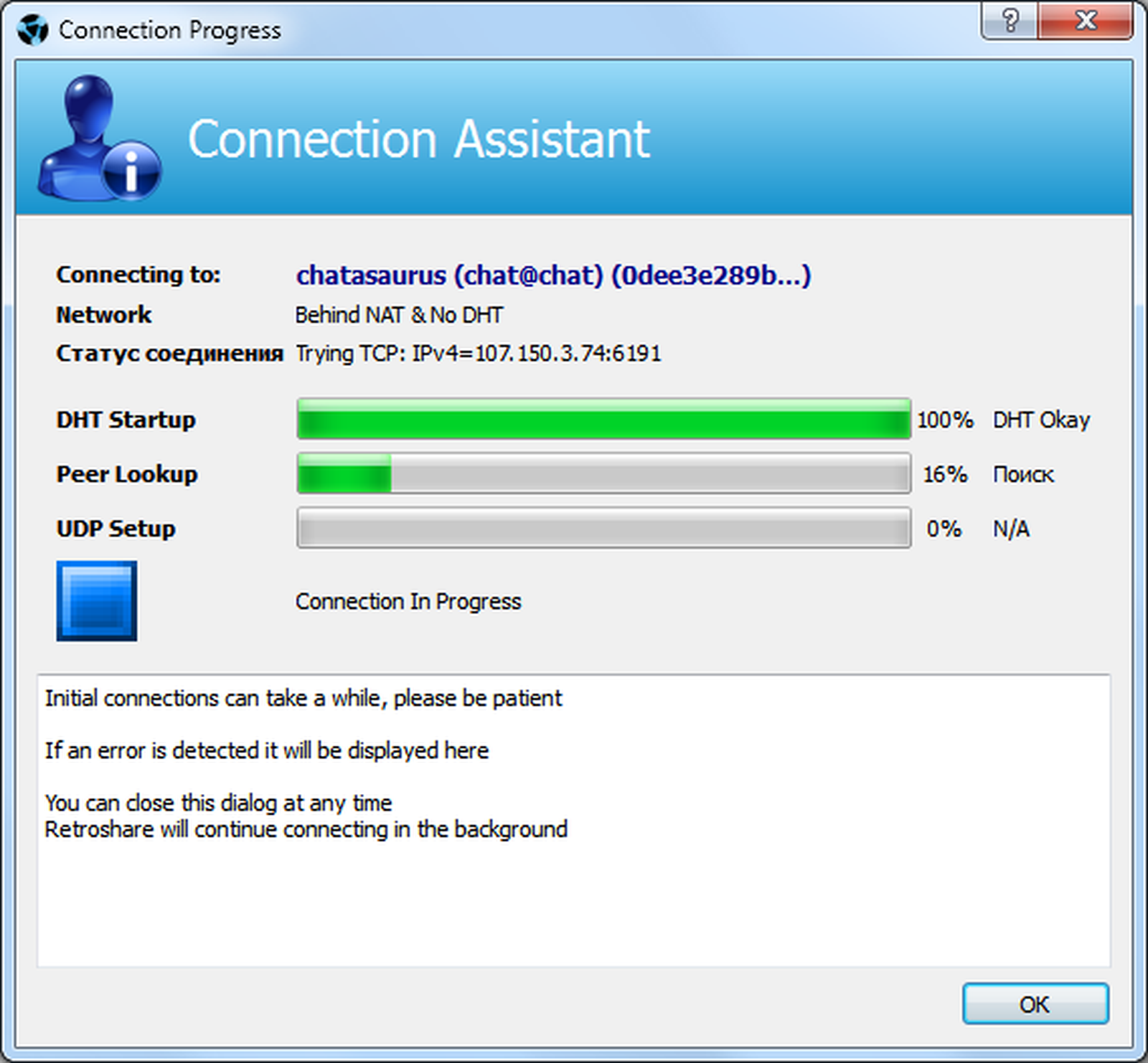


Рисунок 5 - Процесс подключения клиента к роботу

После этого, во вкладке друзья отображается полноценный доверенный участник. При подключении хотя бы к одному действующему участнику сети, пользователь также автоматически становится действующим участником.

Публичные чаты не модерируются и не цензурируются. Это значит, что там отсутствует администрация в той или иной форме, а также возможность наложения запрета на доступ к чат-комнате — бана. Для защиты от спама имеется возможность на уровне клиента установить так называемый «немой режим» (mute) для нарушителя. В результате клиент будет игнорировать сообщения от одного или нескольких пользователей чата, занимающихся рассылкой спама.

**MyChat -** клиент-серверный чат, идеально подходит для передачи сообщений и обмена файлами в офисе или компании, может работать внутри корпоративной сети, а также через Интернет. Будет полезен компаниям, которым нужно средство корпоративного общения, где недопустимо использование таких публичных IM систем, как ICQ или Skype, в связи с ограниченным доступом к Интернету или корпоративными требованиями к безопасности для предотвращения утечки информации.

Клиент позволяет подключаться к отдельным каналам, производить регистрацию в каналах, вести приватные разговоры, передавать файлы, просматривать доски объявлений, искать пользователей, создавать оповещения с различными параметрами.

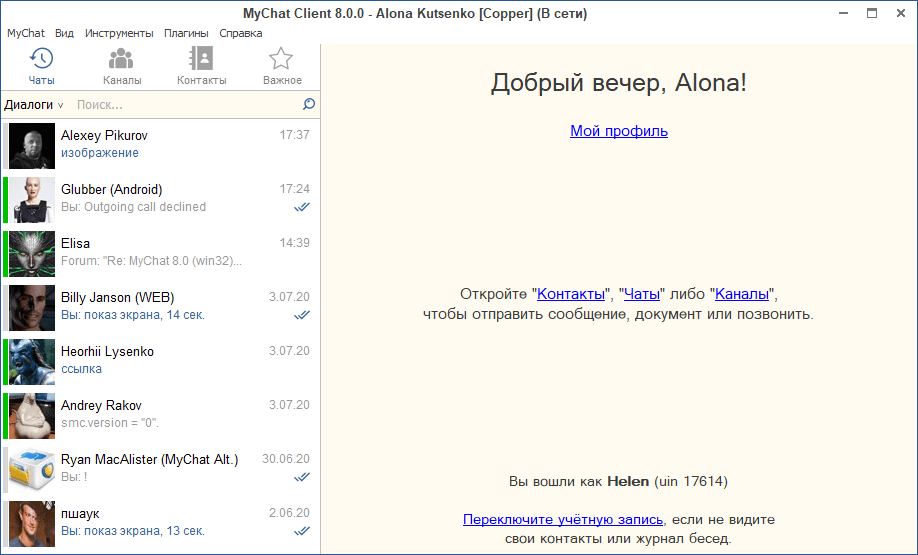


Рисунок 6 - Интерфейс программы «MyChat»

В силу того, что в оповещениях нельзя вести диалог, как в обычном приватном разговоре, существует отправка многоадресных приватных сообщений. Выделяете несколько людей в списке контактов мышкой с зажатой клавишей Ctrl или Shift и отправляете приватное сообщение:

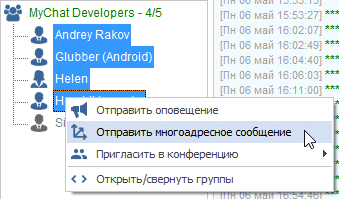


Рисунок 7 - Процесс отправления многоадресных сообщений

Все отправленные сообщения попадают к в историю, но диалоги не открываются, чтобы не плодить лишние окошки. Если придёт ответ, программа сразу это покажет.

Для частых однотипных отправок можно пользоваться собственными уже готовыми списками людей, как в оповещениях:

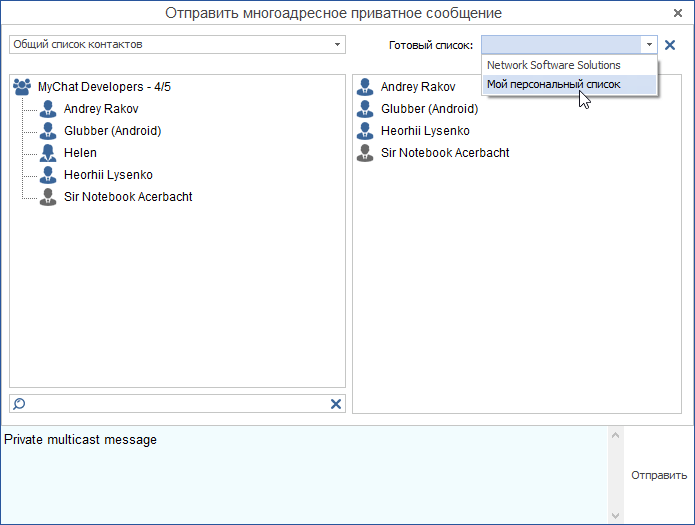


Рисунок 8 - Списки контактов клиента

**2.4. Перечень задач подлежащих решению в процессе разработки**

1. Должны быть реализованы стандартные функции работы чата: задание псевдонима,ввод портов для подключения, отправка и получение сообщений.

2. Программа должна позволять пользователю менять ряд настроек. В частности, имя пользователя в чате, шрифт и размер шрифта текстового поля, а также должна быть реализована возможность сохранения логфайлов.

3. Программный продукт должен иметь дружественный интерфейс, не требующий специальной подготовки пользователей, должен способствовать более эффективному распределению и экономии времени.

**2.5. Обоснование выбора инструментов и платформы для разработки**

В операционной системе Windows реализована объектноориентированная идеология. Базовый объект системы – окно, поведение которого определяется методом, называемым функцией окна. Графический образ окна на экране дисплея – прямоугольная рабочая область. Независимо от своего типа любой объект Windows идентифицируется описателем или дескриптором (handle). Все взаимоотношения программного кода с объектом осуществляются только через его дескриптор. Интерфейс прикладного программирования (API – Application Programming Interface) представляет собой совокупность 32-битных функций (Win32 API), которые предназначены для создания приложений (программ), работающих под управлением Microsoft Windows. Функции объявлены в заголовочных файлах. Главный из них − файл windows.h, в котором содержатся ссылки на другие заголовочные файлы.

API — это аббревиатура названия Application Programming Interface (интерфейс прикладного программирования). API представляет собой совокупность функций и инструментов, позволяющих программисту создавать приложения (программы), работающие в некоторой среде.

Win32 API — это набор функций для создания программ, работающих под управлением Microsoft Windows 98, Windows NT или Windows 2000. Все функции этого набора являются 32 битными, что отражено в названии интерфейса.

Для выполнения курсовой работы был выбран язык программирования C++.

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования.

Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Одними из отличительных особенностей и очень важных преимуществ C++ является:

Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщённое программирование, метапрограммирование (шаблоны, макросы).

Предсказуемое выполнение программ является важным достоинством для построения систем реального времени. Весь код, неявно генерируемый компилятором для реализации языковых возможностей (например, при преобразовании переменной к другому типу), определён в стандарте. Также строго определены места программы, в которых этот код выполняется. Это даёт возможность замерять или рассчитывать время реакции программы на внешнее событие.

Автоматический вызов деструкторов объектов при их уничтожении, причём в порядке, обратном вызову конструкторов. Это упрощает (достаточно объявить переменную) и делает более надёжным освобождение ресурсов (память, файлы, семафоры и т. п.).

Пользовательские функции-операторы позволяют кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.

# 3. Проектно-конструкторская часть

## 

## **3.1 Разработка структуры приложения**

Для разработки приложения использовалась среда Microsoft Visual Studio, включающая в себя широкий набор функциональных инструментов.

Ввод данных в приложении осуществляется через графический интерфейс с использованием клавиатуры. При этом вводимые данные могут быть на любом языке с использованием чисел и системных символов.

После входа в приложение пользователь должен ввести своё имя или псевдоним, в соответствующее текстовое поле, а также собственный порт и порт собеседника, после чего может начать общение в чате. Ввод обоих портов является обязательным, так как наличия неадресуемого пользователя в чате недопустимо.

Написав сообщение его нужно послать, при помощи специальной кнопки отправить. Следствием правильной работы приложения является отображение отправленного сообщения в окне диалога.

Общее назначение программного средства - общение в локальной сети по средствам отправки текстовых сообщений.

**3.2 Используемые функции Win32API**

* **wWinMain -** вызывается системой как начальная точка входа, для базирующейся на Windows, прикладной программы.

int WINAPI wWinMain(

HINSTANCE hInstance, // дескриптор текущего экземпляра окна

HINSTANCE hPrevInstance, // дескриптор предыдущего экземпляра окна

LPSTR lpCmdLine, // указатель на командную строку

int nCmdShow // показывает состояние окна

);

* **DefWindowProc** - вызывается оконной процедурой по умолчанию, чтобы обеспечить обработку по умолчанию любого сообщения окна, которые приложение не обрабатывает. Эта функция гарантирует то, что обрабатывается каждое сообщение.

DefWindowProc(

HWND hWnd, // Дескриптор процедуры, которая получает сообщение

UINT Msg, // Определяет сообщение

WPARAM wParam,// Определяет дополнительную информацию о сообщении

LPARAM lParam // Определяет дополнительную информацию о сообщении

);

* **CreateWindowEx** - создает перекрывающее, выпрыгивающее или дочернее окно с расширенным стилем.

CreateWindowEx(

DWORD dwExStyle,      // улучшенный стиль окна  
LPCTSTR lpClassName,  // указатель на зарегистрированное имя класса  
LPCTSTR lpWindowName, // указатель на имя окна  
DWORD dwStyle,        // стиль окна  
int x,                // горизонтальная начальная позиция окна  
int y,                // вертикальная начальная позиция окна  
int nWidth,           // ширина окна  
int nHeight,          // высота окна  
HWND hWndParent,      // дескриптор родительского или окна владельца  
HMENU hMenu,          // дескриптор меню или ID дочернего окна  
HINSTANCE hInstance,  // дескриптор экземпляра прикладной программы  
LPVOID lpParam        // указатель на данные создания окна

);

* **GetWindowText -** копирует текст заголовка определяемого окна (если окно имеет его) в буфер. Если заданное окно является органом управления, копируется его текст.

GetWindowText(

HWND hWnd, // дескриптор окна или элемента управления с текстом

LPTSTR lpString, // адрес буфера для текста

int nMaxCount // максимальное число символов для копирования

);

* **TranslateMessage -** переводит сообщения виртуальных клавиш в символьные сообщения. Символьные сообщения помещаются в очереди сообщений вызывающего потока для прочтения.

TranslateMessage(

const MSG\* lpMsg // указатель на структуру MSG

);

* **ShowWindow -** устанавливает состояние показа определяемого окна.

ShowWindow(

HWND hWnd, // дескриптор окна

int nCmdShow // состояние показа окна

);

* **RegisterClass -** регистрирует класс окна для последующего использования при вызове функции CreateWindow или CreateWindowEx

RegisterClass(

CONST WNDCLASS\* lpWndClass // указатель на структуру WNDCLASS

);

* **DispatchMessage -**  распределяет сообщение оконной процедуре. Обычно она используется, чтобы доставить сообщение, извлеченное функцией GetMessage.

DispatchMessage(

const MSG\* lpmsg // указатель на структуру MSG, которая содержит сообщение.

);

* **SendMessage -**  отправляет заданное сообщение окну или окнам. Функция вызывает оконную процедуру для заданного окна и не возвращает значение до тех пор, пока оконная процедура не обработает сообщение.

SendMessage(

HWND hWnd, // дескриптор окна, процедура принимает сообщение

UINT Msg, // определяет сообщение, которое будет отправлено

WPARAM wParam, // определяет дополнительную информацию

LPARAM lParam // определяет дополнительную информацию

);

* **WSAStartup -** вызывается для инициации использования WS2 \_32.dll.

WSAStartup(

WORD wVersionRequired, // указать версию интерфейса Windows Sockets

LPWSADATA lpWSAData // должен содержать указатель на структуру типа WSADATA

);

* **inet\_addr -** принимает аргумент типа char \*, представляющий собой указатель на нуль-терминированную строку, содержащую строковое представление ip-адреса.

inet\_addr(

const char \*cp);

* **htons** - преобразует u\_short из хоста в сетевой порядок байтов TCP / IP (который является прямым порядком байтов).

htons(

u\_short hostshort

);

* **bind** - связываем адрес IP с сокетом

bind(

SOCKET s, // дескриптор, идентифицирующий несвязанный сокет.

const sockaddr \*addr, // указатель на  sockaddr локального адреса

Int namelen // Длина в байтах значения, на которое указывает name

);

* **sendto -** отправляет данные на подключенный сокет.

sendto(

SOCKET s,

const char \*buf,

int len,

int flags

const sockaddr \*to,

int tolen

);

* **recvfrom -** получает данные из подключенного сокета или привязанного сокета без установления соединения.

recvfrom (

SOCKET s,

char \*buf,

int len,

int flags );

## **3.3 Разработка архитектуры приложения**

Архитектура приложения клиент - клиент (каждый клиент является одновременно клиентом и сервером).

Классическая архитектура одноранговой сети - тип, в которой все рабочие станции имеют равные возможности и права. Для решения задачи создается одноранговая (peer-to-peer) вычислительная среда, которая позволяет отдельным элементам сети взаимодействовать без помощи серверов. Каждый участвующий компьютер вносит свой вклад в виде файлов, дискового пространства, процессорного времени. Пользователи Сети могут оказывать друг другу услуги на основе кооперации без жесткой зависимости от централизованных служб.

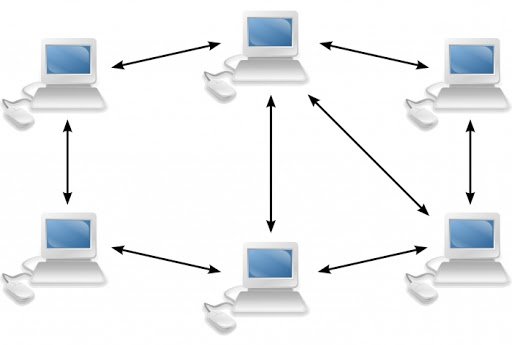


Рисунок 9 - Обобщенная схема однорангового взаимодействия

UDP — User Datagram Protocol

UDP — очень быстрый протокол, поскольку в нем определен самый минимальный механизм, необходимый для передачи данных. Конечно, он имеет некоторые недостатки. Сообщения поступают в любом порядке, и то, которое отправлено первым, может быть получено последним.

Доставка сообщений UDP вовсе не гарантируется, сообщение может потеряться, и могут быть получены две копии одного и того же сообщения. Последний случай возникает, если для отправки сообщений в один адрес использовать два разных маршрута.

UDP не требует открывать соединение, и данные могут быть отправлены сразу же, как только они подготовлены. UDP не отправляет подтверждающие сообщения, поэтому данные могут быть получены или потеряны. Если при использовании UDP требуется надежная передача данных, ее следует реализовать в протоколе более высокого уровня.

## **3.4 Разработка и реализация алгоритмов приложения**

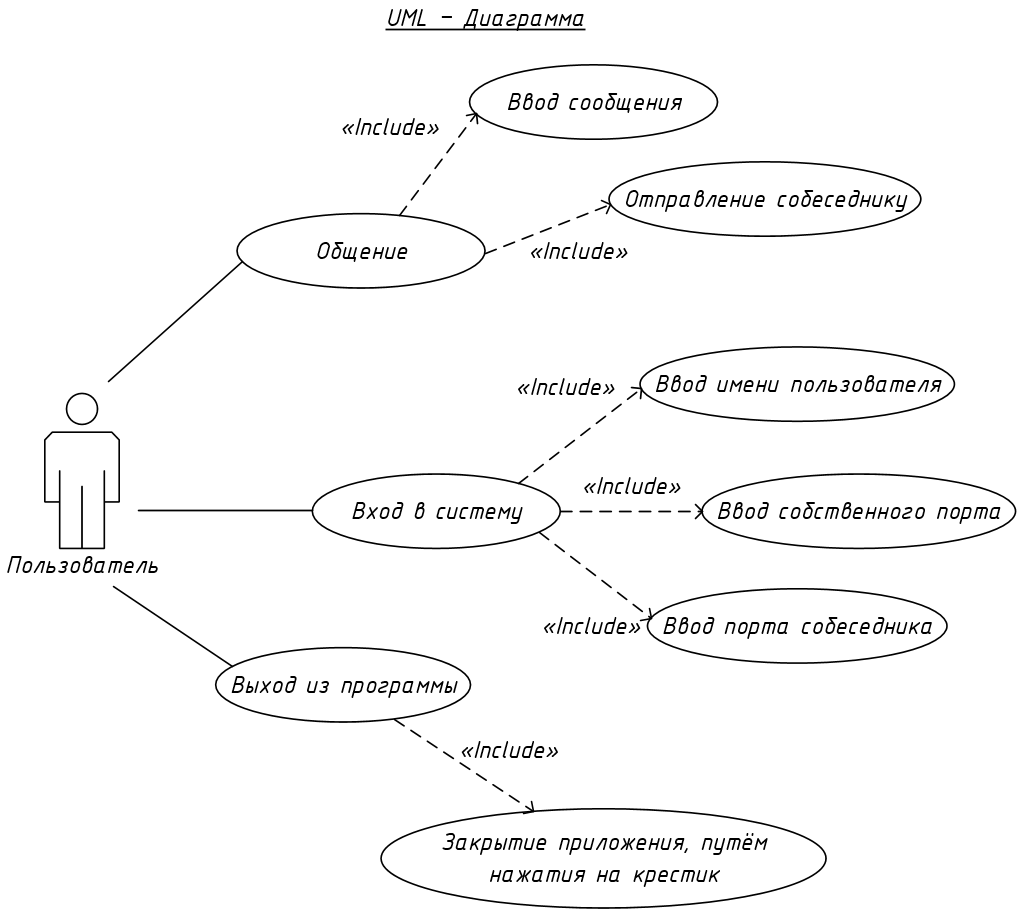


Рисунок 10 - Сценарий использования (Use case) программы

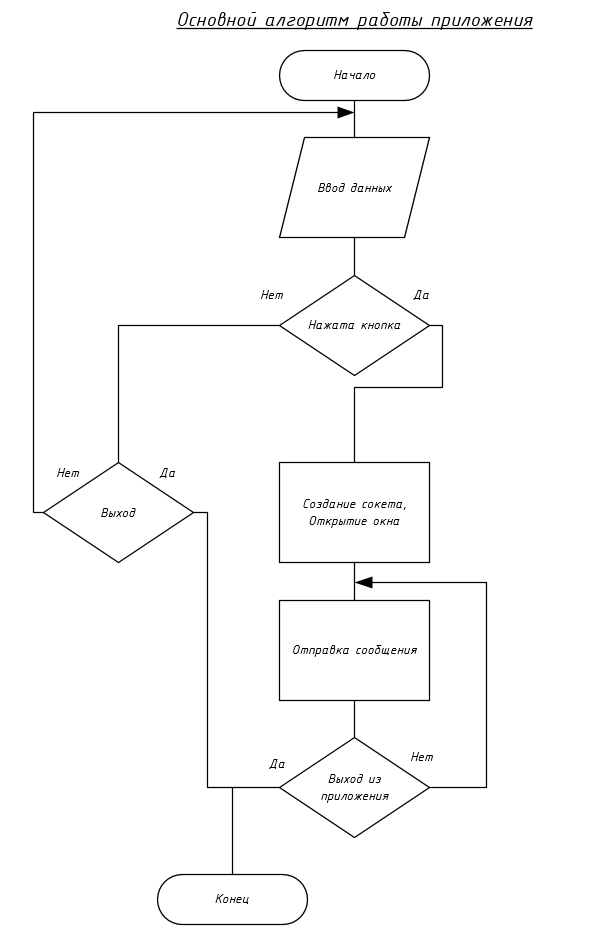


Рисунок 11 - Основная блок-схема, описывающая работу приложения в целом

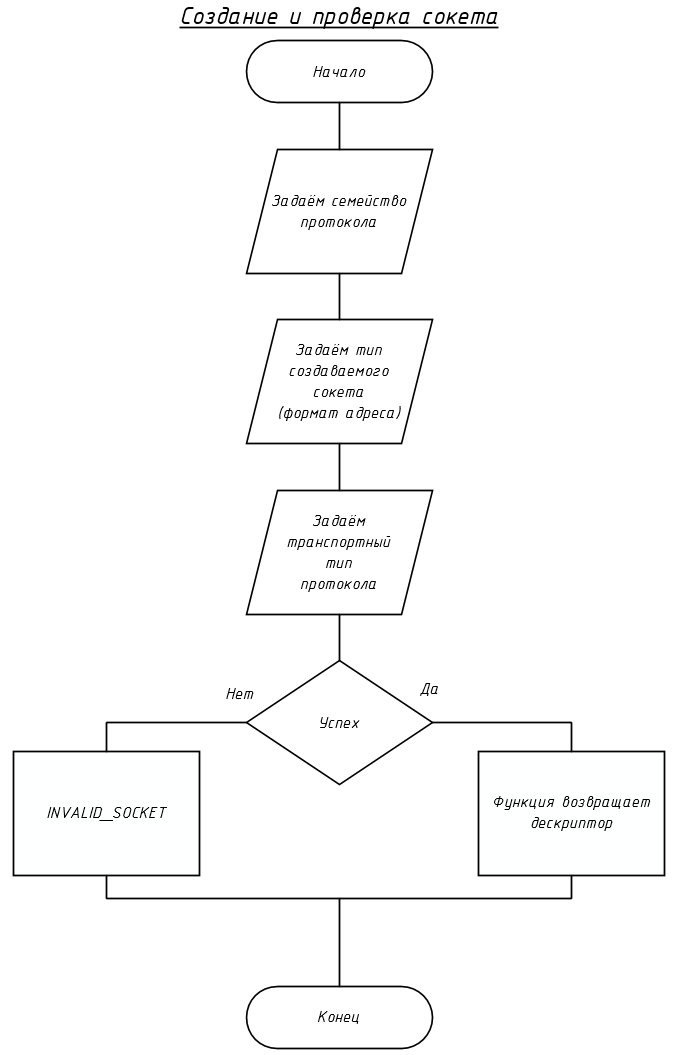


Рисунок 12 - Создание и проверка сокета

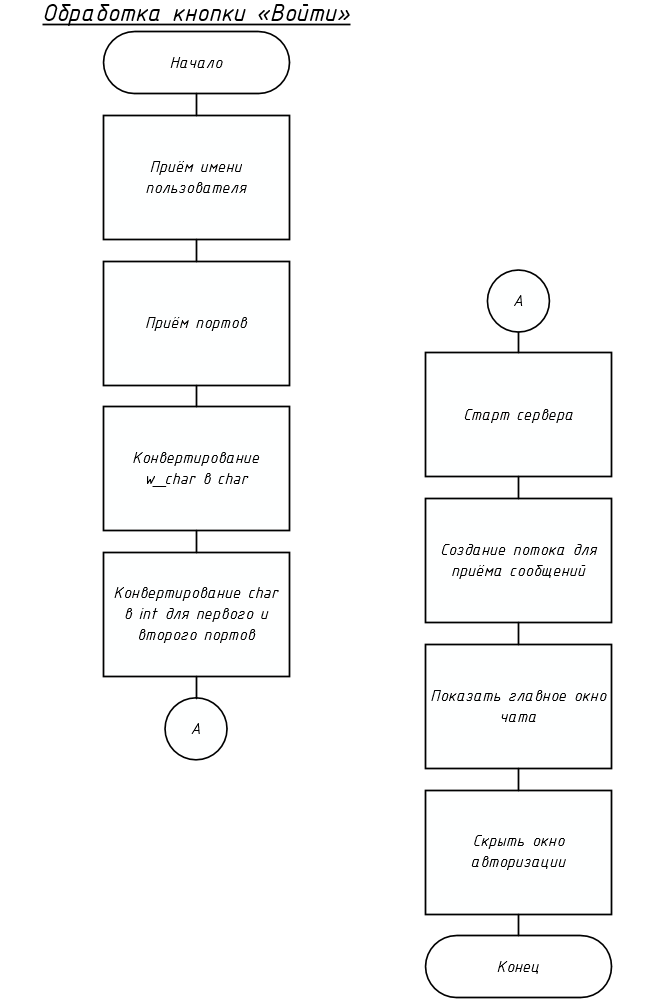


Рисунок 13 - Обработка кнопки «Войти»

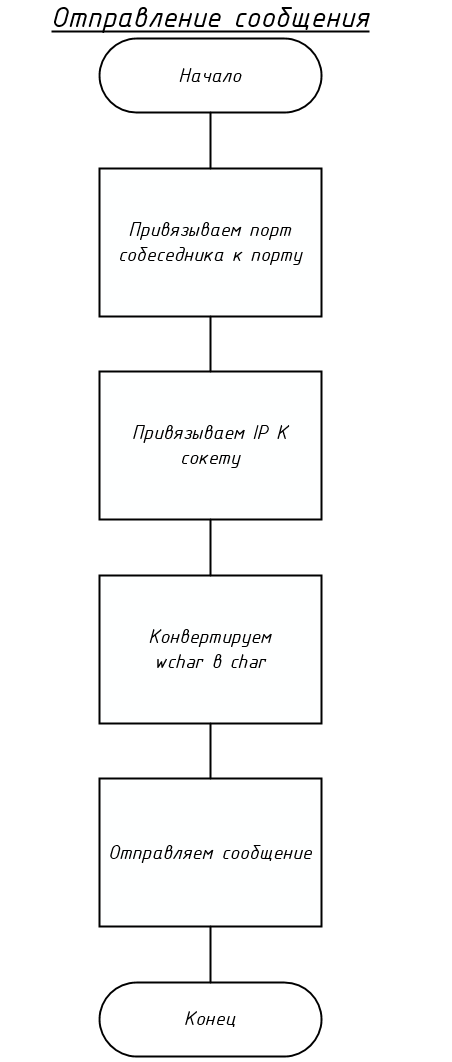


Рисунок 14 - Процесс отправления сообщения

## **3.5 Разработка интерфейса взаимодействия пользователя с системой**

Окно программы должно состоять из трёх текстовых полей для ввода информации: в первое поле необходимо записать имя клиента, во второе поле записать собственный порт, в третье же поле порт собеседника. Помимо этого, должна быть кнопка «Войти», при нажатии на которую открывается второе, новое окно – чат, в котором и происходит общение между двумя пользователями.

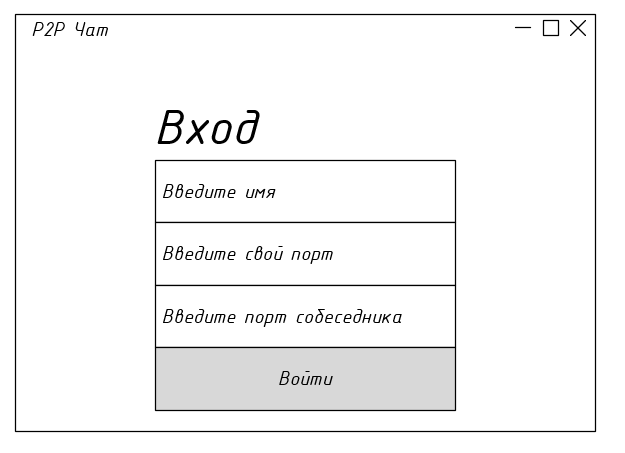


Рисунок 15 - Макет окна входа

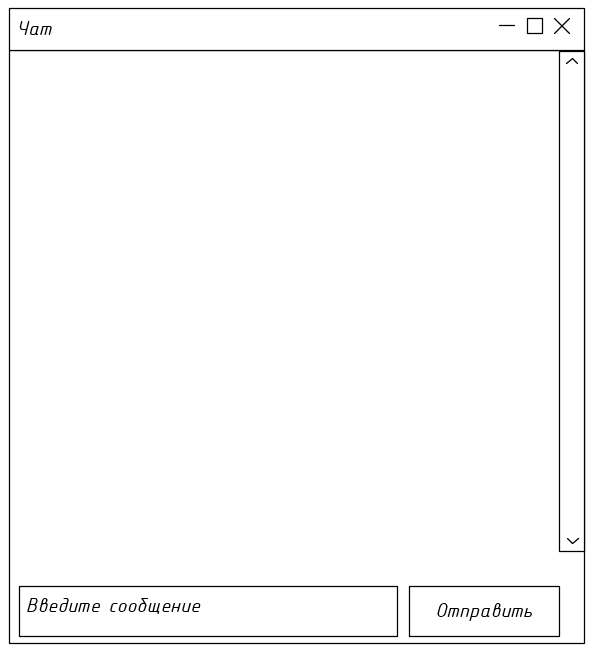


Рисунок 16 - Макет окна чата

# 4. ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

**4.1 Разработка руководства пользователя**

Данный программный продукт служит для мгновенного обмена сообщений между пользователями. Основными его преимуществами являются минимальный размер, нетребовательность в ресурсах.

Запуск программы осуществляется по открытию файла PeerToPeer.exe , после запуска открывается окно приложения. Вводим всю необходимую информацию, чтобы войти в чат.

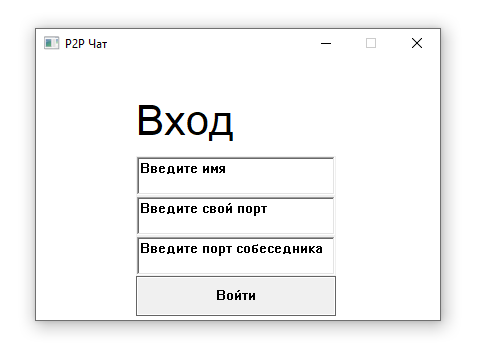


Рисунок 17 - Первоначальный вид окна

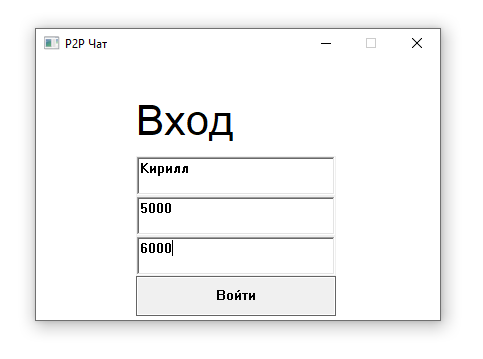


Рисунок 18 - Заполненное окно клиента

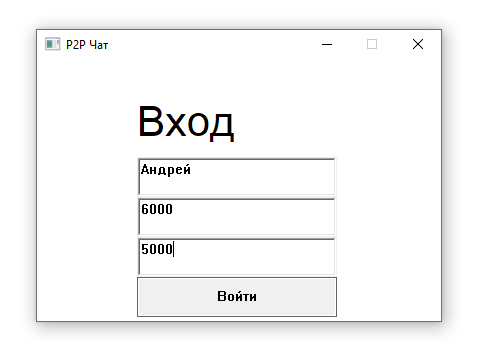


Рисунок 19 - Окно собеседника (второго клиента)

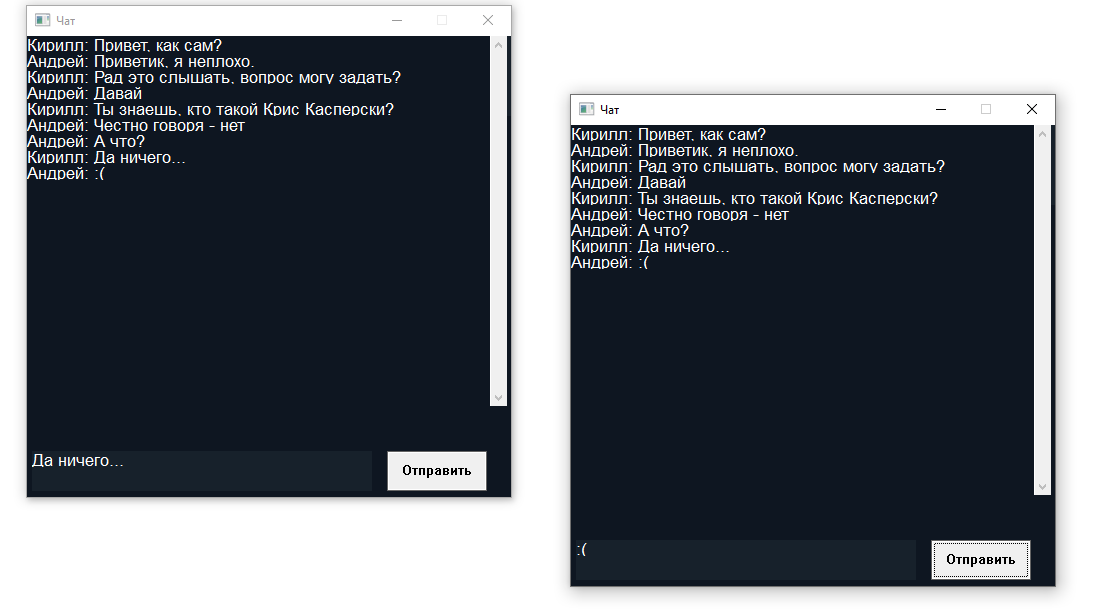


Рисунок 20 - Работающий чат

**Установка и запуск.**

Программа не требует никакой предварительной установки. Разархивируйте архивный файл, содержащий скомпилированные файлы, в удобное место.



Рисунок 21 - Заархивированное приложение

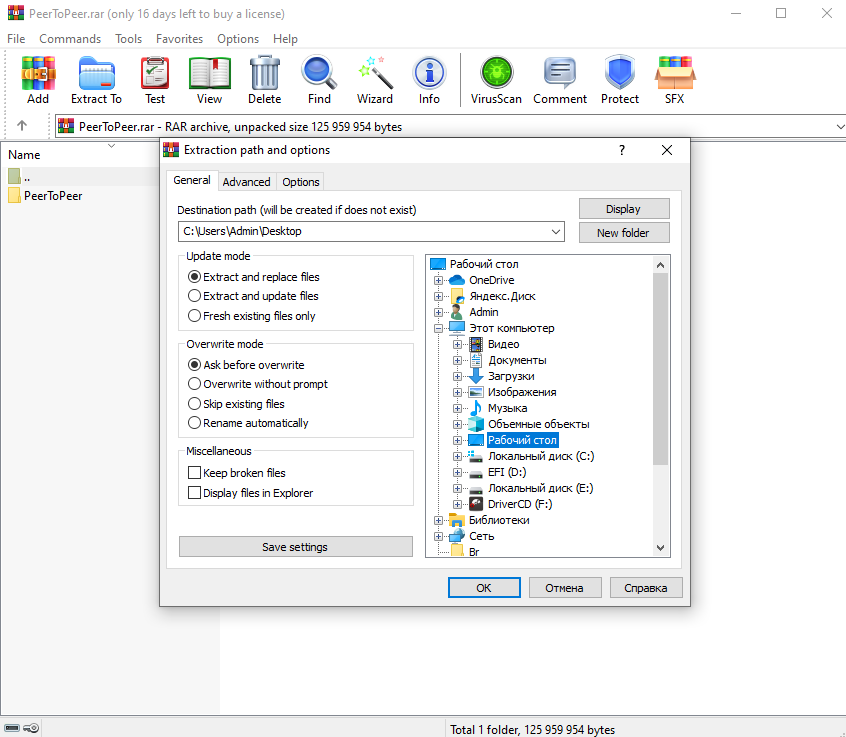


Рисунок 22 - Выбор места для разархивации проекта

Для того, чтобы запустить чат, вам необходимо запустить этот исполняемый файл PeerToPeer.exe по пути PeerToPeer\x64\Debug .

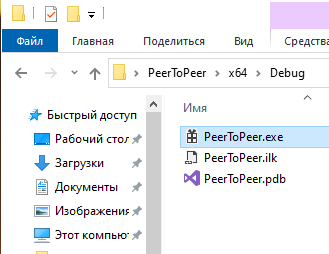


Рисунок 23 - Приложения находится в указанном месте

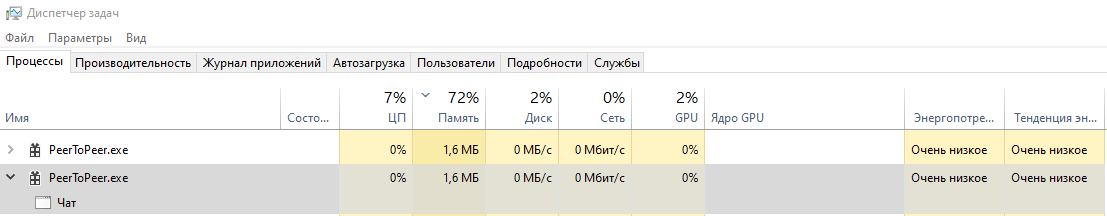
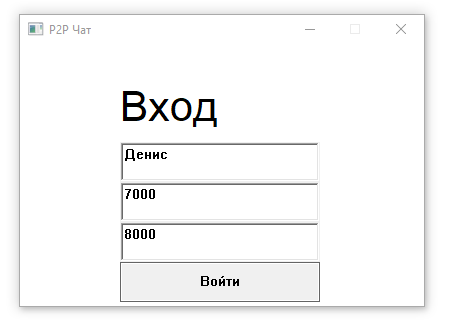


Рисунок 24 - Потребление ресурсов приложением

Использование клиента. После запуска Чат-клиента вы увидите небольшое окно. Оно необходимо для ввода псевдонима, от имени которого будут отправляться сообщения в дальнейшем, а также для установки соединения с собеседником. Ввод псевдонима является необходимым, чтобы не создавать путаницы между участниками чата. Ввод портов является обязательным, так как наличия неадресуемого пользователя в чате недопустимо. После этого вы увидите основной интерфейс программы – большое окно с набором определённых функций в нижней части. Новые сообщения будут появляться вверху. Для отправления сообщения наберите его в окне набора и нажмите кнопку «Отправить». После этого сообщение будет отправлено и отображено у каждого в окне чата.

# 



# Рисунок 24 – заполнение данных для входа

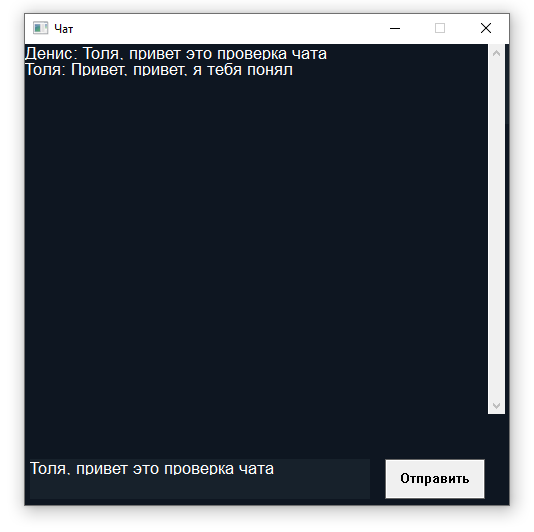


Рисунок 25 – отображение сообщений у клиента «Денис»

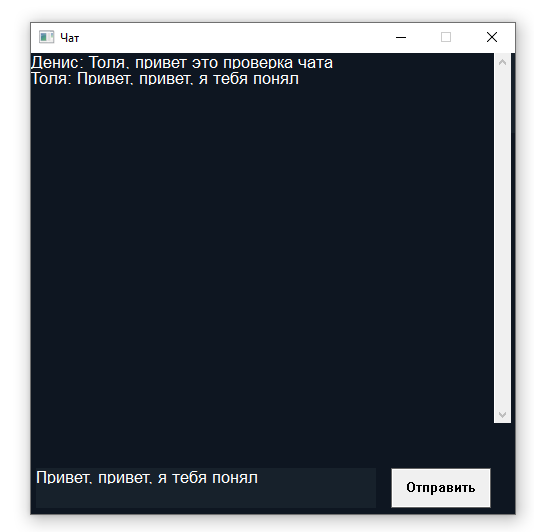


Рисунок 26 - Отображение сообщений у клиента «Толя»

Вход в приложение был выполнен успешно. Клиенты подключились друг к другу. Сообщения корректно отображаются в каждом чате. По завершении общения, приложение можно закрыть.

# 5. ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы был разработан пиринговый чат на основе WinAPI.

Работа выполнялась в несколько этапов: была разработана структура системы и реализовано прикладное оконное приложение на основе WinAPI.

Были сформированы навыки по разработке и реализации программного приложения с использованием интерфейса прикладного программирования (АРI) операционных систем.

Спроектированное приложение подойдет для задач, в ходе которых необходимо запустить файл в заданное пользователем время.

В будущем можно усовершенствовать разработанное приложение путем добавления дополнительного функционала (например, организовав передачу файлов от одного пользователя другому).

**6. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТУРАТУРЫ**

1. Павловская Т.А. C/C++ . Процедурное и объектно-ориентированное программирование. Стандарт 3-го поколения. СПб.: Питер, 2015.

2. Платонов В.В. Программно-аппаратные средства защиты информации М.: Академия, 2014.

3. Бречка Д.М. Операционные системы: в 3 ч. Ч. 1. Пакетные файлы и управление компьютером: учебно-методическое пособие. Омск: Омский государственный университет им. Ф.М. Достоевского, 2012 [https://e.lanbook.com/book/75382?category\_pk=1554#book\_name](https://e.lanbook.com/book/75382?category_pk=1554%23book_name)

4. Вирт Н., Гуткнехт Ю. Разработка операционной системы и компилятора. Проект Оберон. М.: ДМК Пресс, 2012 [https://e.lanbook.com/book/39992?category\_pk=1554#book\_name](https://e.lanbook.com/book/39992?category_pk=1554%23book_name)

5. Сидоров В.Н., Сломинская Е.Н., Полникова Т.В., Макарова О.Ю. Оформление графической части выпускной квалификационной работы. Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2016.