МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования

“Оренбургский государственный университет”

Л.Ф. НАСЕЙКИНА

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ ПРОТОКОЛА TCP/IP

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения
высшего профессионального образования
“Оренбургский государственный университет”

Оренбург 2007

УДК 004.732(076.5)
ББК 32.973.202-02я73
Н 31

Рецензент
кандидат технических наук, доцент Синицын Ю.И.

Насейкина, Л.Ф.
Н – 31 **Разработка клиент-серверных приложений на основе протокола TCP/IP. методические указания к выполнению лабораторной работы. - Оренбург:РИК ГОУ ОГУ, 2007. – 90 с.**

В методических указаниях содержатся материалы, необходимые для самостоятельной подготовки студентов к выполнению лабораторной работы по написанию сетевых приложений с использованием протокола TCP/IP - сетевые игры, обмен текстовыми сообщениями – “Chat”.

Рассматриваются вопросы, касающиеся программирования в вычислительных сетях на основе архитектры “клиент-сервер” на базе инструментальной среды Borland Delphi 6.0.

В описание лабораторной работы включены технология пошагового создания клиент-серверного приложения, руководство по эксплуатации, код программного средства, также приведены контрольные вопросы для закрепления изученного материала, указан список использованных источников.

Методические указания предназначены для выполнения лабораторной работы по дисциплине “Сети ЭВМ и телекоммуникации” для студентов специальности 230105.65 – “Программное обеспечение вычислительной техники и автоматизированных систем” в рамках направления подготовки дипломированного специалиста 654600 - “Информатика и вычислительная техника”.

Н 1404000000 ББК 32.973.202я73
6Л9-07

Содержание

Введение

Объем и способы информирования специалистов с помощью средств компьютерных коммуникаций коренным образом изменились за последние годы. В настоящее время передача данных с помощью компьютеров, использование локальных и глобальных вычислительных сетей становится столь же распространенным, как и сами компьютеры.

На сегодняшний день широкое применение находят программы для удаленной работы с компьютерами. Такими программами могут быть, например, различные чаты, сетевые игры, сетевые базы данных. Принцип работы этих программ один. Клиентская программа (клиент) передает запрос серверу, серверная программа обрабатывает и выполняет запрос клиента. В связи с таким удобством получения и передачи данных, такие программы и получили широкое распространение во всем мире.

В рамках данных методических указаний рассматривается процесс создания сетевых программ по протоколу TCP/IP применительно к инструментальной среде Delphi 6.0. В частности, программирование с использованием компонентов ServerSocket и ClientSocket, предназначенных для написания клиент-серверных приложений.

Целью данных методических указаний является обучение студентов работе с протоколом TCP/IP на основе архитектуры “клиент-сервер”.

Методические указания состоят из введения, четырех глав, заключения, списка использованных источников и приложений. Так, в первой главе рассматриваются основы реализации межсетевого взаимодействия на основе протокола TCP/IP, дается обзор основных сетевых компонентов, подробно описываются все свойства и события компонентов ServerSocket и ClientSocket инструментальной среды Delphi 6.0.

Вторая глава посвящена рассмотрению вопроса обмена сообщениями на базе сетевых компонентов Delphi 6.0. В данной главе подробно рассматриваются процессы разработки Chat-программ с интегрированными и раздельно организованными клиентскими и серверными частями. В третьей главе рассматривается процесс создания сетевых компьютерных игр на примере игры “Шашки”, дается подробное руководство по созданию сетевого приложения данного рода, приводятся фрагменты программного кода.

В четвертой главе приведены контрольные вопросы для закрепления изученного материала. И, наконец, в приложениях приведены варианты заданий, а также правила оформления результатов лабораторной работы.

По окончании изучения представленных методических указаний студенты будут уметь писать программы по обмену текстовыми сообщениями, различные сетевые игры с использованием протокола TCP/IP в инструментальной среде Delphi 6.0.

В заключение, автор выражает признательность рецензенту за внимательное прочтение работы и замечания, способствовавшие улучшению качества предлагаемых методических указаний к выполнению лабораторной работы.

1 Основы межсетевого взаимодействия на основе протокола TCP/IP

Реализация сетевого взаимодействия по протоколу TCP/IP может быть организована на базе различных инструментальных сред программирования. Рассмотрим данный вопрос применительно к среде Borland Delphi 6.0.

В Delphi 6.0 соединение по протоколу TCP/IP может быть реализовано на базе технологии сокетов. Сокет представляет собой окончание сетевого соединения (уровень приложения) со стороны сервера или со стороны клиента /1/. Обычно при соединении приложение-сервер открывает порт с некоторым номером и переходит в состояние ожидания. Приложение-клиент устанавливает соединение с сервером. После этого сокеты можно использовать как канал передачи данных.

Socket (гнездо, разъем) – абстрактное программное понятие, используемое для обозначения в прикладной программе конечной точки канала связи с коммуникационной средой, образованной вычислительной сетью.

При использовании протоколов TCP/IP можно говорить, что socket является сетевым средством подключения прикладной программы к порту локального узла сети.

Существует три основных типа сокетов:

- *клиентские сокеты* - инициализируются со стороны клиента. Для того, чтобы открыть соединение, клиентский сокет должен “знать” имя или IP-адрес сервера и номер порта, используемого серверным сокетом. Клиент посылает серверу запрос на соединение. Сервер ставит эти запросы в очередь и обслуживает их по мере поступления;

- *серверные сокеты* - устанавливают соединение с клиентским сокетом в ответ на его запрос, полученный слушающим сокетом. При этом клиентский сокет получает описание серверного сокета, после чего соединение считается установленным;

- *слушающие сокеты* - создаются сервером и принимают сообщения после запроса на соединение.

Процесс обмена данными между сокетами может происходить в двух режимах: блокирующем и не блокирующем /1/. При блокирующем режиме программа ждет выполнения какого-либо события. При неблокирующем режиме все действия выполняются параллельно.

Рассмотрим механизм реализации сокетов в среде Borland Delphi 6.0. Для работы с сокетами в Delphi 6.0 используются компоненты TClientSocket и TServerSocket. Эти компоненты являются потомками абстрактного класса TabstractSocket, который включает методы и свойства, позволяющие прикладному приложению использовать Windows socket.

Windows socket объединяет в себе набор коммуникационных протоколов, предоставляющих возможность приложению подключаться к другим компьютерам для обмена информацией. Сокеты позволяют приложению создавать соединение с другими машинами без знания конкретного типа протокола.

Windows sockets поддерживает следующие семейства протоколов:

- TCP/IP;
- Xerox Network System (XNS);
- IPX/SPX;
- DECnet.

Для создания совета, инициирующего соединение с другими машинами, используют TClientSocket, а для создания сокета, отвечающего на запросы с других машин – TServerSocket.

Сокету для работы необходимо указать три параметра: *IP-адрес*, связанный с сокетом; *номер порта*, для которого будут выполняться операции обмена данными; *протокол*, по которому будет работать созданный сокет.

IP-адрес – это 32-битный адрес, используемый для идентификации узла в сети. Каждый узел сети должен иметь уникальный IP-адрес, состоящий из идентификаторов сети и обслуживающего компьютера. Этот адрес записывается в точечно-десятичном формате (например: 192.168.144.232).

В компьютере десятки тысяч портов из них несколько сотен используются системой, остальные - как правило, свободны и могут использоваться по желанию программистов. Они необходимы для обмена информацией между клиентом и сервером. Клиенту и серверу необходимо указать свободный порт для корректной работы. Данные на порт сервера могут приходить разными порциями от разных клиентов.

Протокол – это набор правил и соглашений для передачи данных по сети. Такие правила определяют формат, содержание, параметры времени, последовательность и проверку в сообщениях, которыми обмениваются сетевые устройства.

Существует множество протоколов: TCP/IP (Transmission Control Protocol/Internet Protocol), UDP (User Datagram Protocol), TPX/SFX (Internet work / Packet Exchange/Sequenced Packet Exchange) и другие.

Стек TCP/IP содержит набор сетевых протоколов Интернета, поддерживающих связь между объединенными сетями, состоящими из компьютеров различной архитектуры и разными операционными системами.

Также стек TCP/IP включает в себя стандарты для связи между локальными компьютерами, которым назначаются IP-адреса, и соглашения о соединении сетей и правилах маршрутизации сообщений.

В состав стека TCP/IP входит не требующий соединений транспортный протокол UDP. Он является ненадежным, но широко используется в клиент-серверных запросах и приложениях, в которых важна скорость обмена данными, например при передаче информации в интерактивном режиме.

Для решения задачи пересылки сообщений рекомендуется использовать неблокирующие сокеты. При этом, операции приема/передачи сообщений будут происходить асинхронно, и не будет останавливать выполнение программного кода. У клиентского сокета свойство *Client-Type* должно иметь значение ctNonBlocking, а у серверного сокета свойству *ServerType* нужно присвоить значение stNon-Blocking /1/.

Определение свойств *Port* и *ServerType* необходимо, чтобы к серверу могли подключаться клиенты, при этом, нужно, чтобы порт, используемый сервером, точно совпадал с портом, используемым клиентом (и наоборот). Свойство *ServerType* определяет тип подключения.

В Delphi, как правило, при построении программ по протоколу TCP/IP используют следующие стандартные функции.

1) Для передачи сообщений в Delphi можно использовать функции *SendText* и *SendBuf.*

Функция *SendText* посылает текстовое сообщение S через сокет:

function SendText **(const S*: string): integer,***

где: S – передаваемое сообщение.

Возвращаемый параметр – количество отосланных байт, в случае ошибки результат равен – 1.

Функция *SendBuf* посылает буфер через сокет. Буфером может являться любой тип, например структура (record) или integer:

function SendBuf **(var** *buf bufsize:* **integer;** *flags:* **integer): integer,**

где: *buf* - передаваемые данные;

bufsize - размер данных предназначенных для передачи;

flags - параметр передачи данных (рекомендуется приравнять этот параметр к нулю).

Возвращаемое значение - количество отосланных байт, в случае ошибки результат равен – 1.

2) Для получения сообщений используются функции *ReceiveText* и *ReceiveBuf.*

Функция *ReceiveText* получает сообщения в виде текстовой строки и вызывается без параметров:

function ReceiveText: string.

Эта функция возвращает полученную строку.

Функция *ReceiveBuf* получает сообщения через буфер. Как и у функции *SendBuf* буфером функции *ReceiveBuf* может являться любой тип:

function ReceiveBuf (var *buf, bufsize:* **integer,** *flags:* **integer):** integer,

где: *buf* – полученные данные;

bufsize – размер принятых данных;

flags - параметр передачи данных (рекомендуется принять этот параметр равным 0).

Возвращаемое значение – количество полученных байт, в случае ошибки результат равен – 1.

3) Для уточнения размера полученных данных используется функция:

function ReceiveLength: integer.

Возвращаемое значение – количество полученных байт.

Когда на прослушиваемый сокетом порт приходит информация, переданная удаленным компьютером, вызывается событие *OnClientRead,* поэтому функции, предназначенные для получения данных, необходимо располагать в обработчике этого события.

Рисунок 1 – Вкладка Internet палитры компонентов среды Delphi 6.0

На рисунке 1 представлена вкладка Internet палитры компонентов среды Delphi 6.0. Компоненты этой страницы обеспечивают средства связи программы с глобальной компьютерной сетью Internet и позволяют программистам организовывать связь между отдельными терминалами сети по протоколу TCP/IP /2/.

Эти компоненты позволяют установить соединение между двумя удаленными компьютерами сети, один из которых клиент, а другой сервер. Оба компьютера должны следовать протоколу TCP/IP, обеспечивающему логическую независимость связи от аппаратных средств компьютеров /3/.

В таблице 1 приводится обзор основных сетевых компонентов.

Таблица 1 – Компоненты страницы Internet

Название компонента	Описание назначения
1	2
ClientSocket (Сокет клиента)	Используется для установки связи с удаленным компьютером на основании его IP-адреса. Объект не визуальный.
ServerSocket (Сокет сервера)	Позволяет отслеживать возникновение соединения пользовательского компьютера с сервером TCP/IP. При возникновении связи этот компонент ответственен за ее формирование и обмен данными. Объект не визуальный.
WebDispatcher (Диспетчер)	Используется для преобразования обычных данных в формат HTML и их последующей передачи серверному приложению. Объект не визуальный.
PageProducer (Страница Web)	Используется для преобразования шаблона HTML в обычный текст HTML, который может быть интерпретирован клиентским приложением, например Web-броузером.
DataSetTableProducer (Страница набора данных)	Используется для публикации на Web-сервере таблицы базы данных. Объект не визуальный.

Продолжение таблицы 1

1	2
 DataSetPageProducer (Страница Web с данными)	Позволяет преобразовать шаблон HTML в обычный формат HTML или текстовый формат, который может быть отображен в Web-броузере. Объект не визуальный.
 QueryTableProducer (Страница набора данных SQL)	Представляет в виде страницы Web записи набора данных, полученного на основе запроса SQL. Работает вместе с Query.
 SQLQueryTable Producer (Страница набора данных SQL)	Представляет в виде страницы Web записи набора данных, полученного на основе запроса SQL. Только в Delphi 6.0.
 TcpClient (Клиент TCP)	Клиентское приложение TCP. Используется только в Delphi 6.0.
 TcpServer (Сервер TCP)	Серверное приложение TCP. Используется только в Delphi 6.0.
 UdpSocket (Сокет UDP)	Сокет клиентских и серверных приложений UDP/IP. Используется только в Delphi 6.0.
 XMLDocument (Документ XML)	Доступ к документу XML, размещенному в файле или в памяти.
 WebBrowser (Броузер Web)	Компонент используется для создания окна стандартного Web-броузера, предназначенного для выполнения функций отражения страниц HTML.

Далее представлено описание свойств и методов компонентов ServerSocket и ClientSocket.

Компонент ServerSocket. Сервер, основанный на сокетном протоколе, позволяет обслуживать сразу множество клиентов. Для каждого подключенного клиента сервер открывает отдельный сокет, по которому можно обмениваться данными с клиентом. Также возможно создание для каждого подключения отдельного процесса.

Определение свойств Port и ServerType необходимо, чтобы к серверу могли подключаться клиенты, при этом, нужно, чтобы порт, используемый сервером точно совпадал с портом, используемым клиентом (и наоборот). Свойство ServerType определяет тип подключения. На этапе открытия сокета

и указанного порта может выполняться автоматическое начало ожидания подсоединения клиентов (Listen). При отключении клиента закрывается его сокетное соединение с сервером. По команде приложения сервер завершает свою работу, закрывая все открытые сокетные каналы и прекращая ожидание подключений клиентов.

Свойство ServerType: TServerType указывает тип сервера. Оно может принимать одно из двух значений: stNonBlocking – синхронная работа с клиентскими сокетами. При таком типе сервера можно работать с клиентами через события OnClientRead и OnClientWrite. StThreadBlocking – асинхронный тип. Для каждого клиентского сокетного канала создается отдельный процесс (Thread).

Свойство Active: Boolean – показатель того, активен в данных момент сервер, или нет. Значение True указывает на то, что сервер работает и готов к приему клиентов, а False – сервер выключен. Чтобы запустить сервер, нужно просто присвоить этому свойству значение True.

Port: Integer это номер порта для установления соединений с клиентами. Значение порта у сервера и у клиентов должны быть одинаковыми. Рекомендуются значения от 1025 до 65535, т.к. от 1 до 1024 – могут быть заняты системой.

Метод Open запускает сервер. Эта команда идентична присвоению значения True свойству Active.

Метод Close останавливает сервер.

Событие OnClientConnect возникает, когда клиент установил сокетное соединение и ждет ответа сервера (OnAccept).

Событие OnClientDisconnect возникает, когда клиент отсоединился от сокетного канала.

Событие OnClientError возникает, когда текущая операция завершилась неудачно, т.е. произошла ошибка.

Событие OnClientRead возникает, когда клиент передал серверу какие-либо данные. Доступ к этим данным можно получить через передаваемый параметр Socket: TCustomWinSocket.

Событие OnClientWrite возникает, когда сервер может отправлять данные клиенту по сокету.

Событие OnAccept возникает, когда сервер принимает клиента или отказывает ему в соединении.

Событие OnListen возникает, когда сервер переходит в режим ожидания подсоединения клиентов.

Компонент ClientSocket. После назначения свойствам Host и Port соответствующих значений, можно приступить непосредственно к открытию сокета (сокет здесь рассматривается как очередь, в которой содержатся символы, передающиеся от одного компьютера к другому). Для этого можно вызвать метод Open компонента TClientSocket, либо присвоить свойству Active значение True.

Этап авторизации необходим, если сервер требует ввода логина и/или пароля. На этом этапе посылается серверу логин (имя пользователя) и пароль. Механизм авторизации зависит уже от конкретного сервера.

Свойство Host: string это строка, указывающая на хост-имя компьютера, к которому следует подключиться.

Address: string – строка, указывающая на IP-адрес компьютера, к которому следует подключиться. В отличие от Host, здесь может содержаться лишь IP-адрес. Отличие заключается в том, что при указании в Host символьного имя компьютера, IP-адрес, соответствующий этому имени, будет запрошен у DNS.

Строка Service: string, определяет службу (ftp, http, pop, и т.д.), к порту которой произойдет подключение. Это своеобразный справочник соответствия номеров портов различным стандартным протоколам.

Свойство ClientType это тип соединения. CtNonBlocking – асинхронная передача данных, т.е. посылать и принимать данные по сокету можно с помощью OnRead и OnWrite. CtBlocking – синхронная (одновременная) передача данных. События OnRead и OnWrite не работают. Этот тип соединения полезен для организации обмена данными с помощью потоков.

Событие OnConnect возникает при установлении соединения. То есть в обработчике этого события уже можно начинать авторизацию или прием/передачу данных.

Событие OnConnecting возникает при установлении соединения. Отличие от OnConnect в том, что соединение еще не установлено. Обычно такие промежуточные события используются для обновления статуса.

Событие OnDisconnect возникает при закрытии сокета.

Событие OnError возникает при ошибке в работе сокета. Следует отметить, что это событие не поможет найти ошибку в момент открытия сокета (Open). Для того, чтобы избежать выдачи сообщения об ошибке, необходимо заключить операторы открытия сокета в блок try..except (обработка исключительных ситуаций).

Событие OnLookup возникает при попытке получения от DNS IP-адреса указанного хоста.

Событие OnRead возникает, когда удаленный компьютер послал какие-либо данные. При возникновении этого события возможна обработка данных.

Событие OnWrite возникает, когда разрешена запись данных в сокет.

Метод SendBuf (var Buf; Count: Integer) используется при посылке буфера через сокет. Буфером может являться любой тип, будь то структура, либо простая переменная. Буфер указывается параметром Buf, вторым параметром необходимо указать размер пересылаемых данных в байтах (Count).

Метод SendText (const S: string) используется при посылке текстовой строки через сокет. SendStream (AStream: TStream) это посылка содержимого указанного потока через сокет. Пересылаемый поток должен быть открыт. Поток может быть любого типа – файловый или из оперативной памяти.

2 Обмен сообщениями на базе сетевых компонентов Delphi 6.0 (Chat)

Для реализации обмена сообщениями на базе сетевых компонентов среды Borland Delphi 6.0 (chat) необходимо сконструировать клиент-серверное приложение, в котором серверная и клиентская части могут быть реализованы как в одном приложении, так и в разных.

Рассмотрим создание таких приложений на конкретных примерах.

2.1 ВАРИАНТ ИНТЕГРИРОВАННОГО КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

2.1.1 Реализация серверной части

Для начала необходимо загрузить инструментальную среду Delphi 6.0. Для этого необходимо щелкнуть левой кнопкой мыши на меню “Пуск”, затем выбрать подраздел “Программы” или “Все программы”, после чего необходимо щелкнуть левой кнопкой мыши на пиктограмме “Delphi 6.0” (рисунок 2).

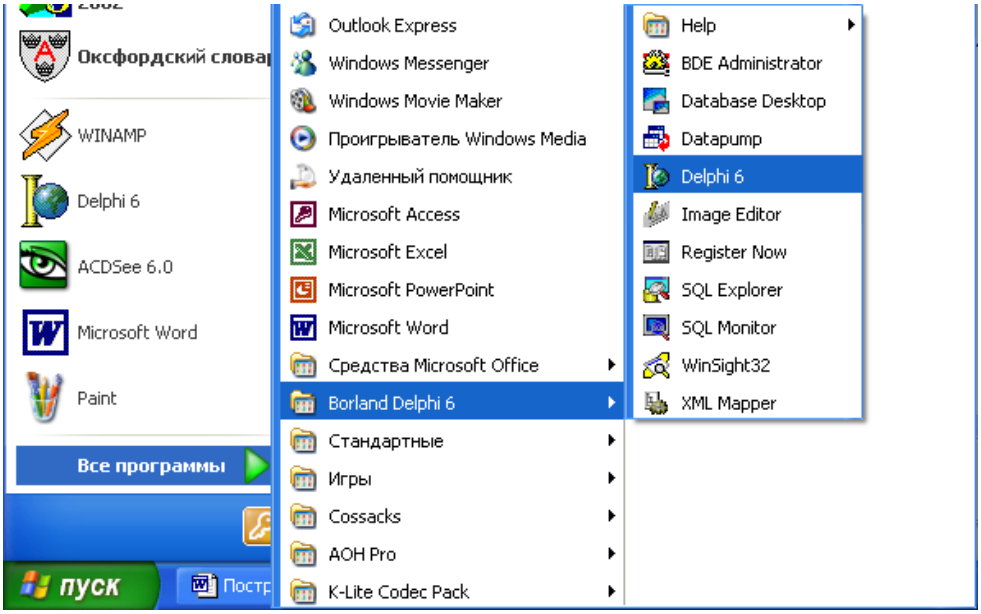


Рисунок 2 – Загрузка Delphi 6.0

После запуска программы, требуется дождаться окончательной загрузки приложения. Далее появится рабочая среда программирования. В ней будет необходимо создать одну форму и поместить на нее компоненты.

При создании сервера, использующего сокет TCP/IP для реализации сетевого соединения, необходимо выполнить следующие действия:

1) Для начала, нужно расположить на форме компонент ServerSocket со страницы Internet для реализации TCP/IP соединения со стороны сервера.
2) С помощью палитры компонентов Delphi создать на форме объекты, функционально необходимые для разрешения проблемы. Необходимо расположить на форме компонент типа TEdit для ввода передава-

емых данных и два поля типа TMemo: первое – для отображения получаемых данных, второе – для сообщений об ошибках.

3) Для того чтобы установить соединение, сервер первоначально должен находится в режиме прослушивания соединения. Для этого необходимо расположить на форме кнопку типа TButton, выполняющую открытие соединения. В обработчике события OnClick этой командной кнопки можно записать следующий программный код:

```
ServerSocket1.Active := True;
```

4) Для отображения текущего состояния соединения со стороны сервера можно использовать строку состояния. Для этого необходимо расположить на форме компонент типа TStatusBar со страницы Win32 палитры компонентов Delphi. Навести указатель мыши на добавленный компонент и дважды щелкнуть кнопкой мыши для вызова редактора панелей строки состояния. Щелчком на кнопке Add New добавить новую панель. Затем в обработчике события OnClick созданной кнопки ввести следующий код программы:

```
StatusBar1.Panels[0].Text := 'Прослушивание…';
```

5) Выполнить настройку TCP/IP соединения со стороны сервера. Для этого установить в Инспекторе Объектов для компонента ServerSocket1 значение свойства Port равным любому значению, допустимому для номера порта сервера (например, 1024).

6) Создать для компонента ServerSocket1 обработчик события OnClientConnect. Это событие будет инициализировано при подключении клиента. Ввести в созданный обработчик события программный код, информирующий об установлении соединения. Например:

```
Memo2.Lines.Add('К серверу подключается клиент');
```

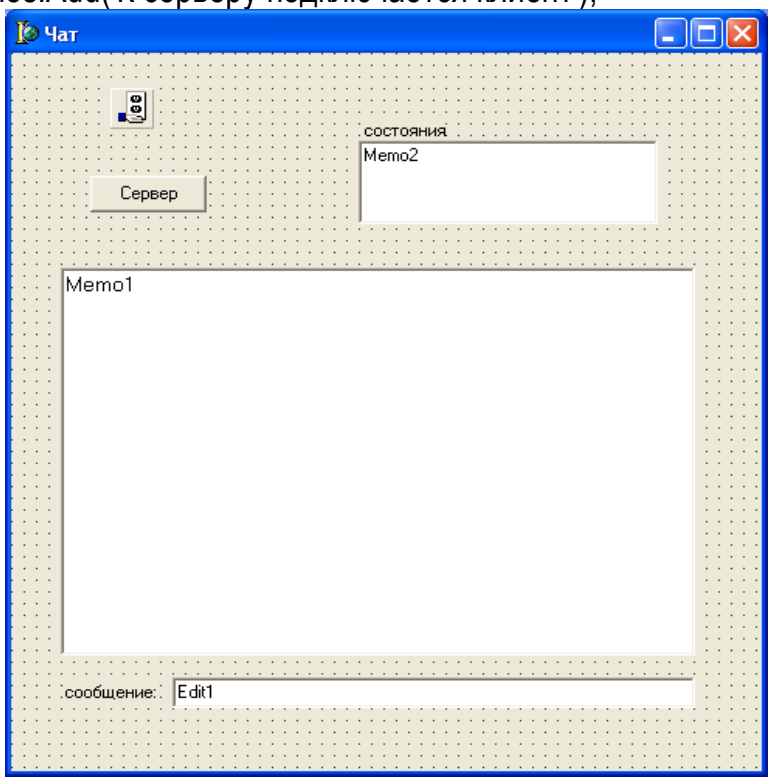


Рисунок 3 – Форма с некоторыми компонентами

7) Необходимо создать для компонента ServerSocket1 обработчик события OnAccept и ввести в него код, информирующий об установлении соединения с локальным портом (рисунок 5). Свойство LocalAddress определяет IP-адрес при установке локального соединения. Для получения IP-адреса удаленного соединения следует использовать свойство RemoteAddress. Например:

```
Memo2.Lines.Add('Соединение установлено');
StatusBar1.Panels[0].Text := 'Соединено с :' + Socket.LocalAddress;
```

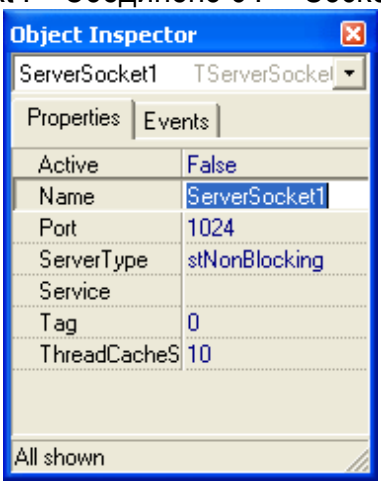


Рисунок 4 – Настройка TCP/IP соединения со стороны сервера

8) Создать для компонента ServerSocket1 обработчик события OnClientDisconnect (рисунок 5) и ввести в него программный код, информирующий о состоянии соединения. Например:

```
StatusBar1.Panels[0].Text := 'Прослушивание соединения...';
```

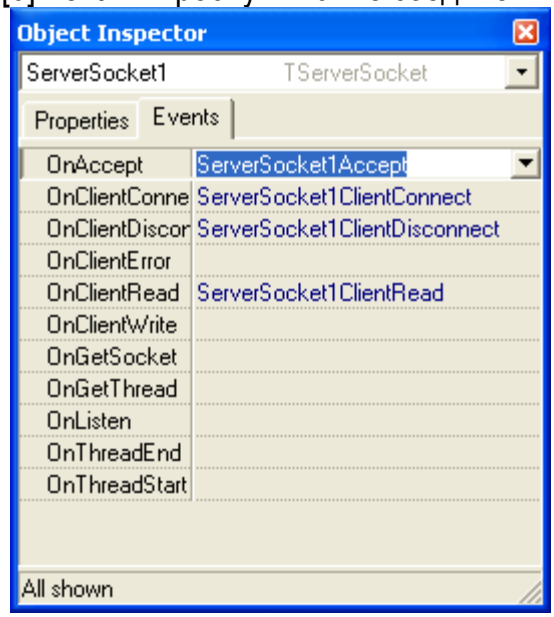


Рисунок 5 – События для компонента ServerSocket1

Следующие пункты предназначены для реализации приема и передачи данных по установленному TCP/IP соединению.

9) Для передачи строки по установленному TCP/IP соединению используется метод SendText объекта типа TCustomWinSocket. Нужно

создать кнопку типа TButton. В обработчике события OnClick ввести программный код:

```
if ServerSocket1.Active then
ServerSocket1.Socket.Connections[0].SendText(edit2.Text+' : '+edit1.Text);
```

10) После выполнения этих действий для приложения-клиента будет инициировано событие OnRead.
11) Используя объект типа TServerSocket, необходимо реализовать получение данных на стороне сервера. Для этого нужно создать для компонента ServerSocket1 обработчик события OnClientRead (рисунок 4). Это событие будет инициализировано при передаче по установленному соединению данных с клиента. Ввести в созданный обработчик события код, принимающий передаваемые данные. Метод ReceiveText принимает данные, переданные объекту Socket. Например, следующий код отображает принимаемые данные в компоненте типа TMemo:

```
Memo1.Lines.Add(Socket.ReceiveText);
```

Таким образом, создание сокета со стороны сервера завершено.

2.1.2 Реализация клиентской части

Для того чтобы создать клиента, использующего сокет TCP/IP для реализации сетевого соединения необходимо выполнить следующие действия:

1) На форме расположить компонент ClientSocket  со страницы Internet палитры компонентов. Этот компонент используется для реализации TCP/IP соединения со стороны клиента.
2) Написать код, выполняющий установление соединения с сервером. Сначала следует определить, открыто ли соединение, и если соединение открыто, то закрыть его. Открыть соединение можно, установив свойство Active равным True, для закрытия необходимо установить False. Затем следует расположить на форме кнопку типа TButton, и в обработчике события OnClick этой кнопки следует записать:

```
Var Server: String; {объявите как глобальную переменную модуля}
Begin
if ClientSocket1.Active
then
  ClientSocket1.Active := False;
```

3) Далее следует установить IP-адрес подключаемого сервера. Например, ввести для запроса IP-адреса следующий программный код:

```
if InputQuery('Установить связь с','Псевдоним IP-адреса:',Server) then
if Length(Server) > 0 then
```

4) Установить значения свойств компонента ClientSocket1. Свойство Host должно содержать псевдоним для IP-адреса. Если значение этого свойства установлено, то оно преобладает над

значением свойства Address, определяющего IP-адрес. Например:

```
with ClientSocket1 do
Begin
  Host := Server;
  Active := True;
end;
```

5) Создать код, выполняемый для закрытия соединения. Например, расположить на форме кнопку типа TButton и в обработчик события OnClick для этого компонента ввести:

```
ClientSocket1.Active := False;
```

6) Выполнить настройку TCP/IP соединения со стороны клиента. Для этого необходимо установить в Инспекторе Объектов для сетевого компонента ClientSocket1 значение свойства Port равным номеру порта, используемого сервером (1024) (рисунок 7).

7) Создать для компонента ClientSocket1 обработчик события OnConnect (рисунок 8). Это событие будет инициализировано при подключении клиента. Ввести в созданный обработчик события код, информирующий об установлении соединения. Свойство LocalHost содержит информацию о псевдониме IP-адреса, с которым установлено соединение. Например:

```
StatusBar1.Panels[0].Text := 'Соединено с адресом: '+Socket.LocalHost;
```

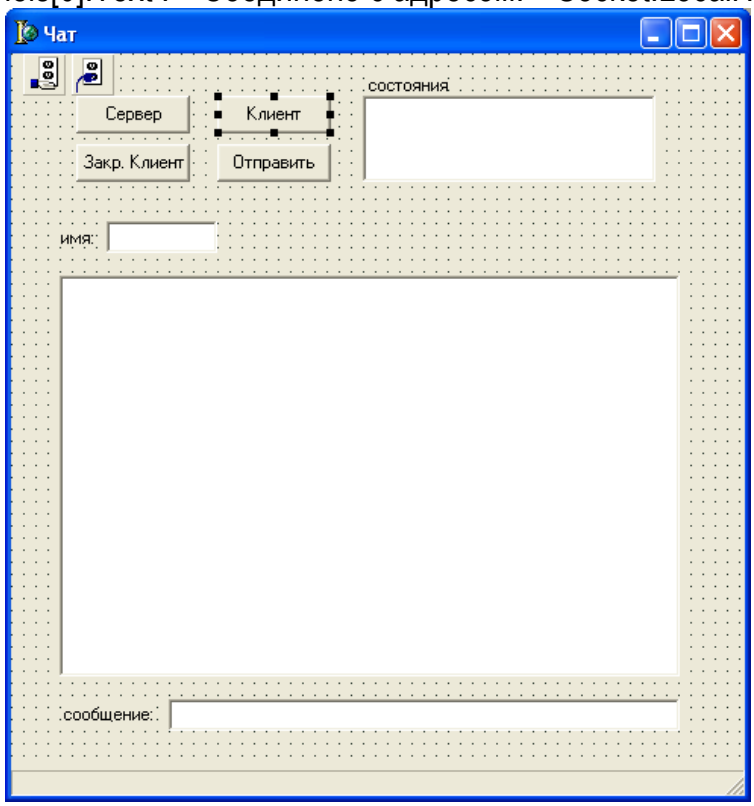


Рисунок 6 – Готовая форма программы в режиме Конструктор

8) Создать для компонента ClientSocket1 обработчик события OnDisconnect (рисунок 8). Ввести в созданный обработчик со-

бытия код, информирующий о состоянии соединения. Например:

```
StatusBar1.Panels[0].Text := 'Соединение закрыто';
```

9) Создать для компонента ClientSocket1 обработчик события OnError (рисунок 8). Это событие будет инициализировано при неудаче соединения с сервером. Ввести в созданный обработчик события код, отображающий сообщение об ошибке соединения:

```
Memo2.Lines.Add('Ошибка соединения с сервером : '+Server);
ErrorCode := 0;
```

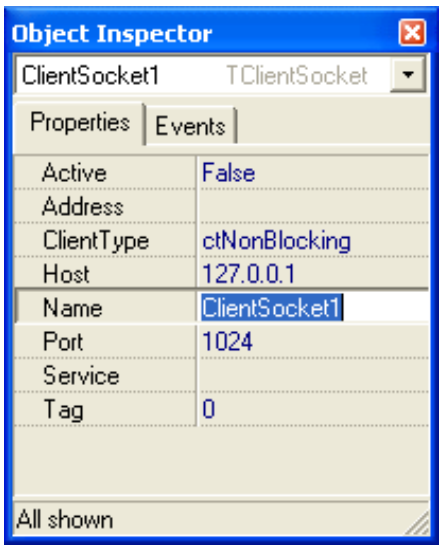


Рисунок 7 - Настройка TCP/IP соединения со стороны клиента

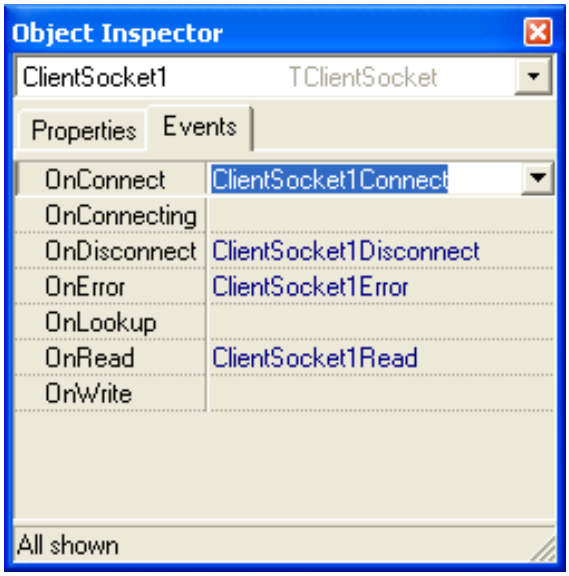


Рисунок 8 – События для компонента ClientSocket1

Следующие пункты предназначены для реализации приема и передачи данных по установленному TCP/IP соединению со стороны клиента.

10) Создать код, выполняющий передачу данных серверу. Наряду с методом SendText объекта типа TCustomWinSocket для передачи данных по установленному TCP\IP соединению можно использовать метод SendBuf. Например, для созданного ранее

объекта TButton, осуществляющего передачу данных в обработчике события OnClick необходимо ввести:

```
if ClientSocket1.Active
then
ClientSocket1.Socket.SendText(Edit2.Text+' : '+Edit1.Text);
```

11) После выполнения этих действий для приложения сервера будет инициировано событие OnClientRead.

12) Создать код, выполняющий получение данных на стороне клиента. Для этого необходимо создать для компонента ClientSocket1 обработчик события OnRead (рисунок 8). Метод ReceiveText принимает данные, переданные объекту Socket. Например, нужно ввести в созданный обработчик события код, принимающий передаваемые данные и отображающий их в компоненте Memo1:

```
Memo1.Lines.Add(Socket.ReceiveText);
```

Создание сокета со стороны клиента закончено. В завершение необходимо сохранить проект и можно запускать программу.

Сохранить проект можно при помощи пиктографических кнопок: “Save” , “Save all” , расположенных на панели инструментов Delphi 6.0, либо через пункт меню “File”.

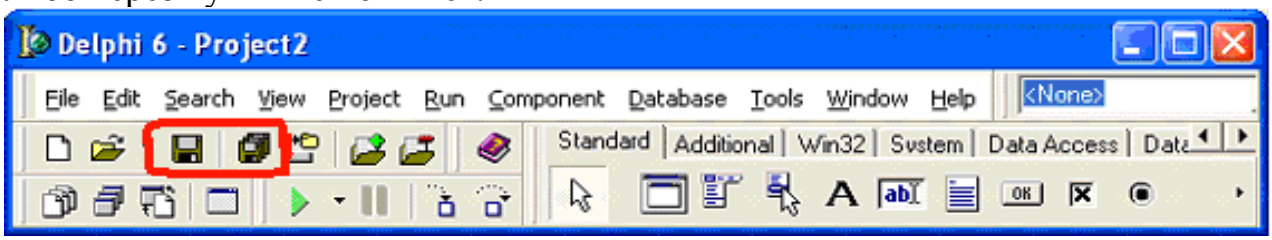


Рисунок 9 – Панель Инструментов Delphi 6.0

В Delphi необходимо выполнить команду “File” => “Save As…”, в результате станет активным окно для сохранения текущего программного модуля (рисунок 10). Необходимо указать свое имя, или согласиться с предложенным, затем требуется нажать на кнопку “Сохранить”.

Выполнить команду “File” => “Save Project As…” для сохранения проекта. Необходимо указать свое имя проекта, или согласиться с предложенным, затем требуется нажать на кнопку “Сохранить”.

Рекомендуется сохранить проект, рабочие модули и остальные файлы, необходимые для нормальной работы программы в отдельной папке, к примеру, в папке с именем – Чат.

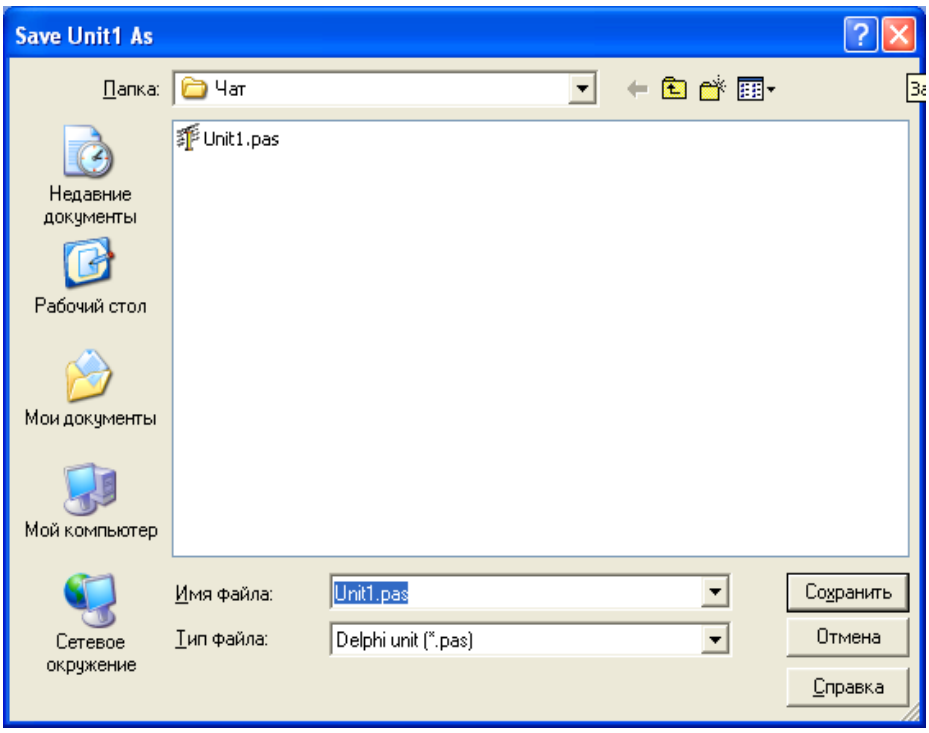


Рисунок 10 – Сохранение модуля в Delphi

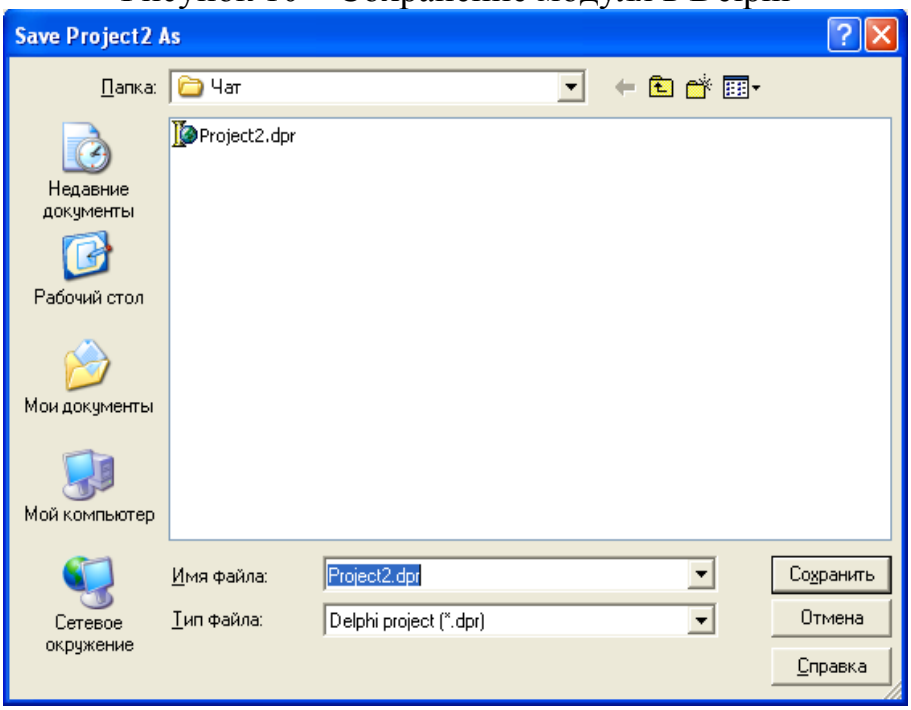


Рисунок 11 – Сохранение проекта в Delphi

2.1.3 Руководство по эксплуатации программного средства

Ознакомимся с руководством по эксплуатации данного программного средства. Данная программа реализует обмен текстовыми сообщениями (Chat) на основе протокола TCP/IP. После запуска программы на экране появится форма, представленная на рисунке 12.

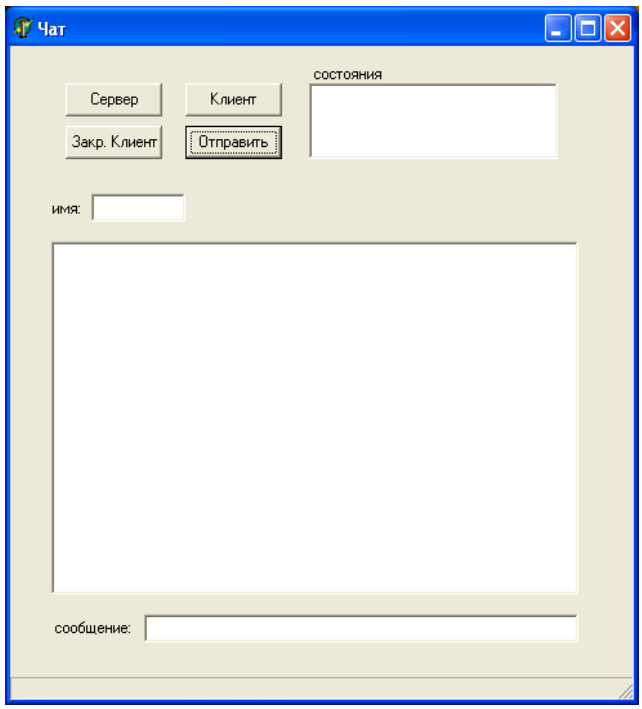


Рисунок 12 – Форма после запуска программы

Существует множество вариантов вызова программы на выполнение, приведем некоторые из них:

1) Программу можно запустить на выполнение из инструментальной среды Delphi 6.0 для этого необходимо нажать F9.

2) Возможен другой способ запуска программы:

- в проводнике Windows необходимо, при помощи указателя мыши перейти к той папке, в которой сохранен проект. Затем дважды щелкнуть на папку, которую следует открыть;

- после необходимо отыскать .exe файл с указанным ранее именем. Необходимо использовать горизонтальную и вертикальную полосы прокрутки, в том случае, если в текущей папке много файлов и каталогов;

- дважды щелкнуть на данном файле левой кнопкой мыши.

Если в папке “Мой компьютер” или ее подпапках нет файла или папки, которые требуется открыть, найти этот файл или папку, нажав кнопку “Поиск”. Чтобы запустить средство поиска, нажать кнопку “Пуск” и выбрать команду “Найти”.

Выполнить выше описанные действия еще раз. Причем, важно понимать что, выполнять это приложение можно как на одном, так и на разных компьютерах локальной вычислительной сети.

Для использования TCP/IP соединения для передачи или приема данных сначала следует запустить приложение-сервер и перевести его в режим ожидания соединения с клиентом. Для этого необходимо руководствоваться приведенными ниже описаниями действий.

В одном приложении необходимо щелкнуть указателем мыши на кнопке “Сервер”. В результате текущее состояние соединения со стороны сервера изменится, о чем свидетельствует надпись в строке состояния: “Прослушивание…”.

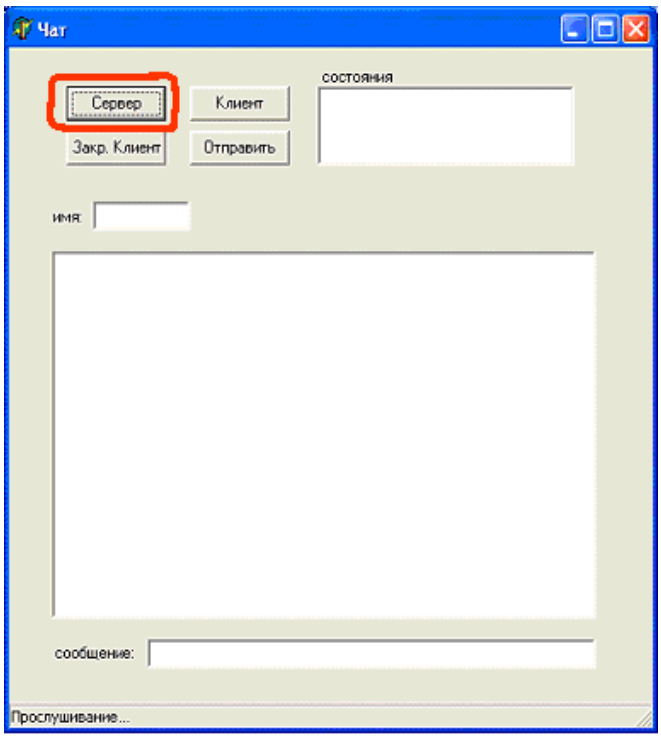


Рисунок 13 – Прослушивание сервером

Далее следует запустить приложение-клиент и указать IP-адрес сервера. Для этого в другом приложении необходимо щелкнуть по кнопке “Клиент”.

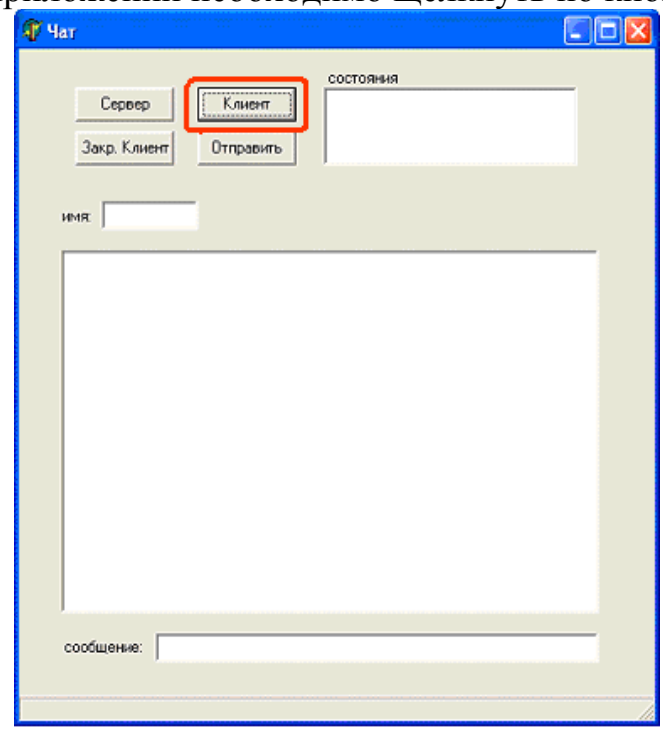


Рисунок 14 – Подключение клиента

В результате откроется окно “Установить связь с”. При выполнении сервера на локальном компьютере следует вводить адрес 127.0.0.1.

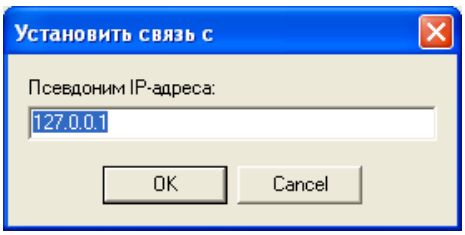


Рисунок 15 – Прописывание IP-адреса

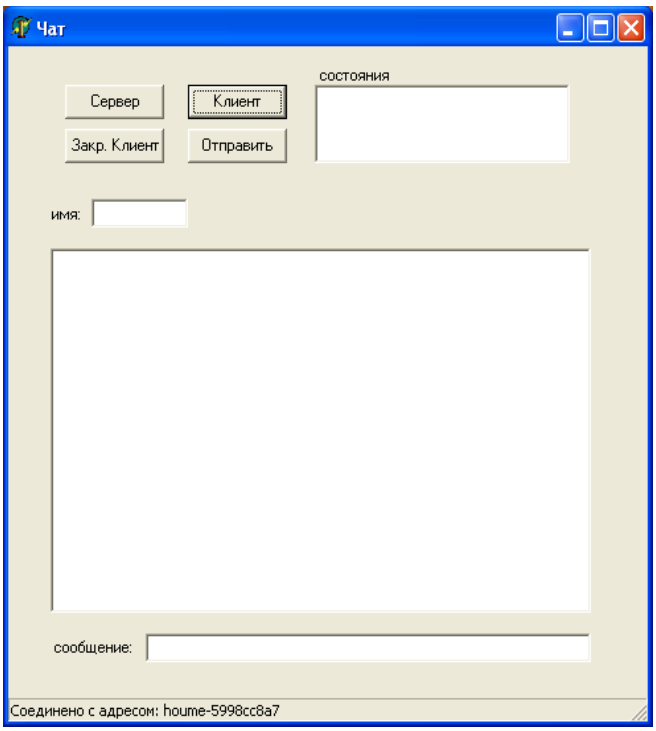


Рисунок 16 – Окно клиента после установки подключения

После установки связи в строке состояния, расположенной в нижней части окна, появляется информация: для приложения-сервера информация, свидетельствующая об установлении соединения с локальным портом; для приложения-клиента, содержит информацию о IP-адресе, с которым установлено соединение.

В поле “Состояния” для сервера должны появится сообщения: “К серверу подключается клиент”, а через некоторое время: “Соединение установлено”.

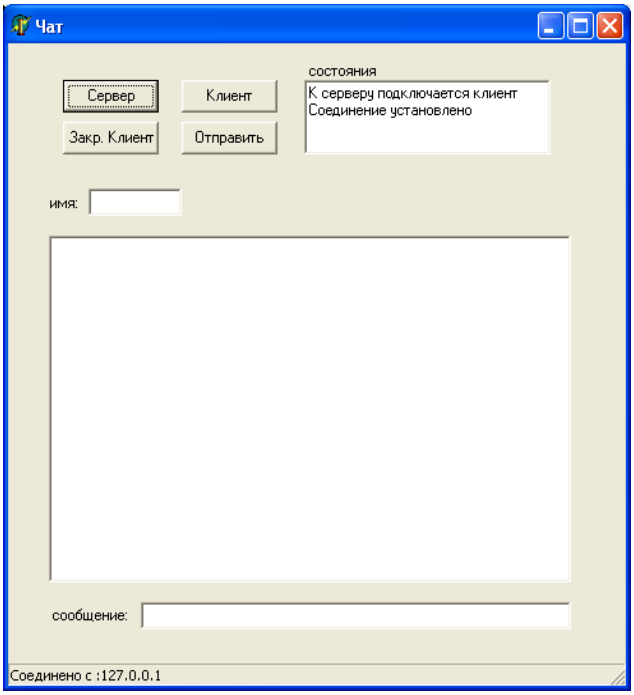


Рисунок 17 – Окно сервера после установки подключения

Здесь необходимо указать имя, если оно не указано программа выдаст ошибку.

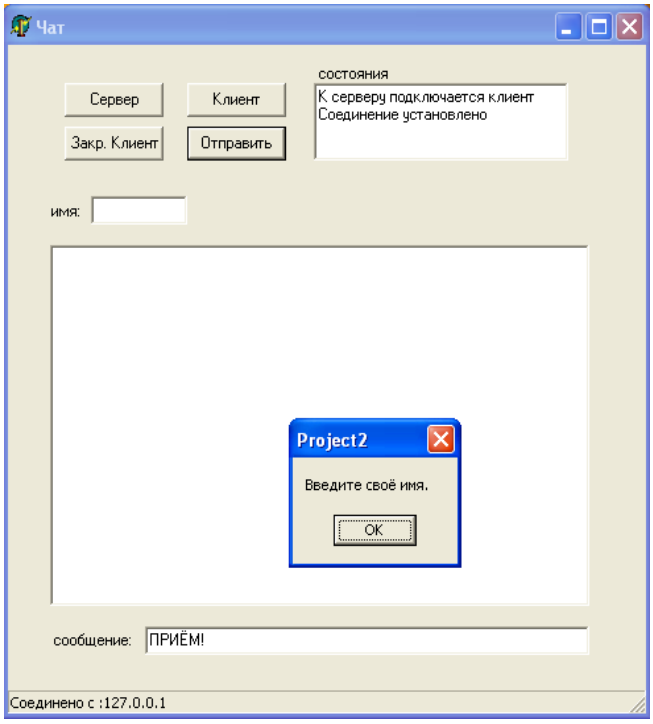


Рисунок 18 – Сообщение об ошибке

Затем можно ввести текст своего сообщения. Ниже приводится пример, в котором под именем Саша зарегистрировано приложение-сервер, а под именем Даша – приложение-клиент.

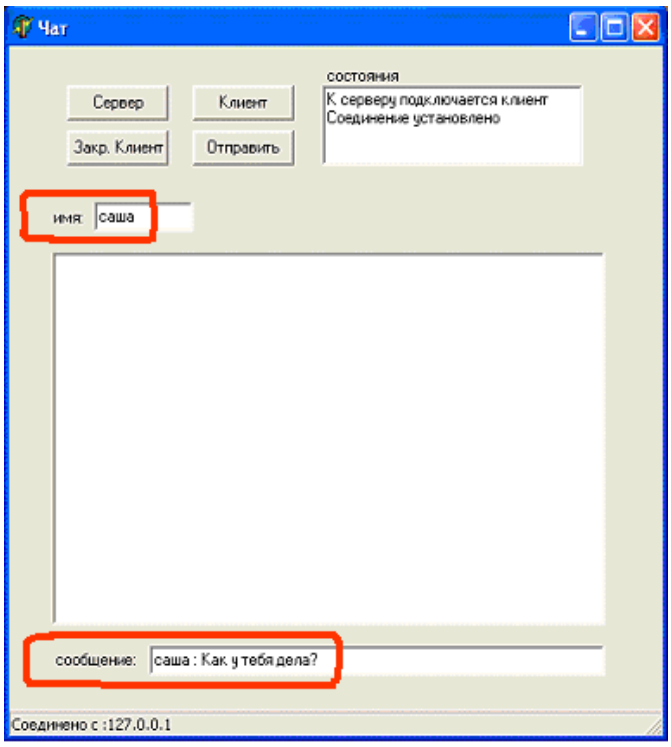


Рисунок 19 – Ввод сообщения

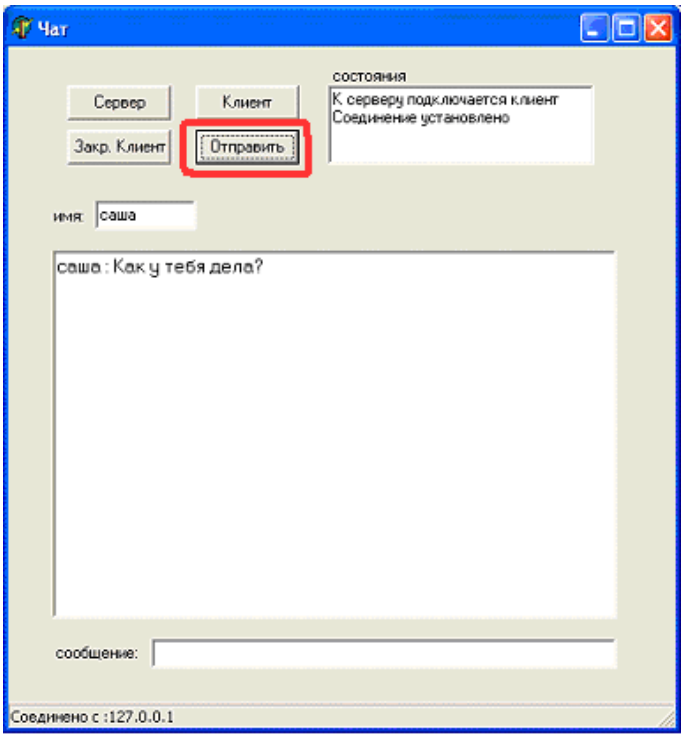


Рисунок 20 – Отправка сообщения

Для посылки сообщения необходимо щелкнуть на кнопке “Отправить”. После того, как нажата кнопка “Отправить”, поле “Сообщение” автоматически будет чистым – готовым к вводу нового сообщения. Посланное сообщение будет отображено в текстовом поле, расположенном в центральной части окна. Причем, в этом поле содержится информация об имени отправителя и отправленном сообщении. Эти атрибуты разделены двоеточием с пробелом.

На рисунках 21-22 приводится подробный пример работы программы в случае взаимодействия двух приложений – клиента и сервера.

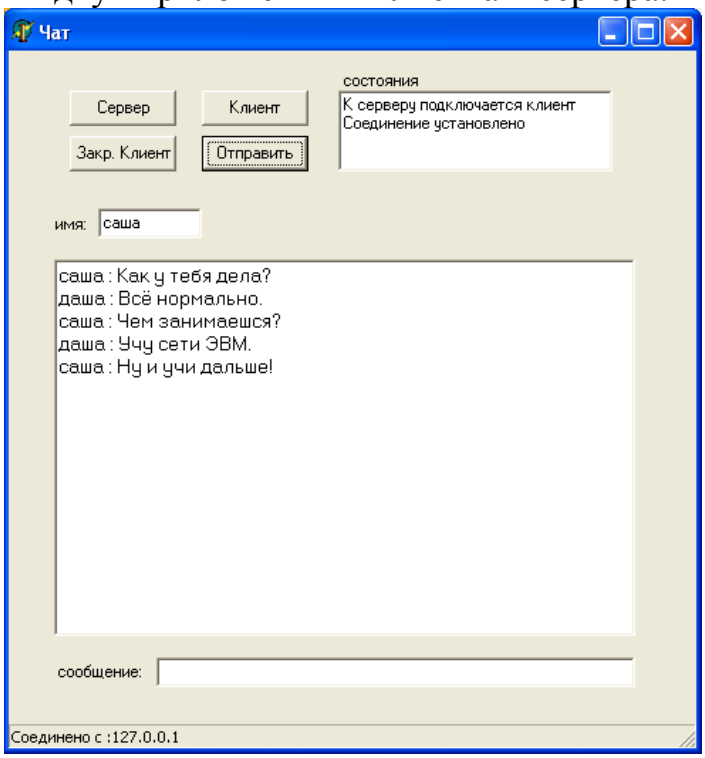


Рисунок 21 – Приложение сервер

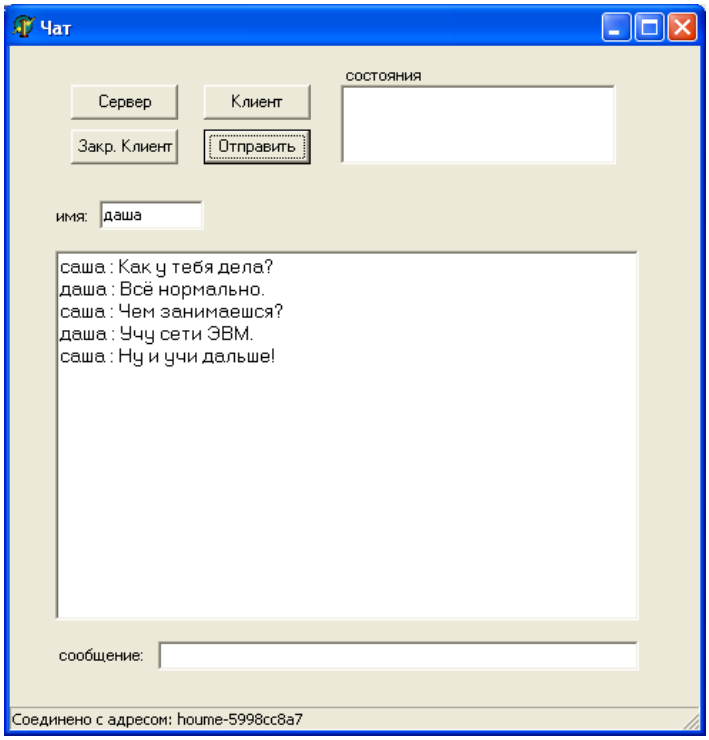


Рисунок 22 – Приложение клиент

В программе предусмотрена возможность закрытия клиента, для этого необходимо щелкнуть левой кнопкой мыши на кнопке “Закр. Клиент.” Данная ситуация наглядно отображена на рисунке 23.

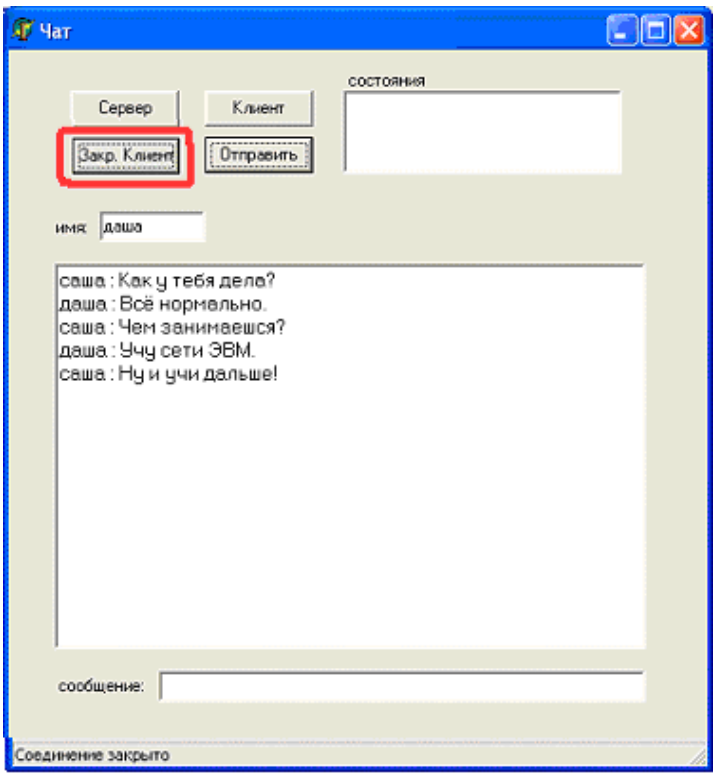


Рисунок 23 – Соединение закрыто

Для выхода из программы необходимо нажать кнопку “Закрыть” в правом верхнем углу строки заголовка.

2.1.4 Программный код

```
{Текст программы}
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ScktComp, ComCtrls;
type
  TForm1 = class(TForm)
    ClientSocket1: TClientSocket;
    ServerSocket1: TServerSocket;
    Button1: TButton;
    Edit1: TEdit;
    Memo1: TMemo;
    Button2: TButton;
    Button3: TButton;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    StatusBar1: TStatusBar;
    Memo2: TMemo;
    Button4: TButton;
```

```
    Label3: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure ServerSocket1ClientRead(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure ClientSocket1Read(Sender: TObject; Socket: TCustomWinSocket);
    procedure ServerSocket1ClientConnect(Sender: TObject;Socket:TCustomWinSocket);
    procedure ServerSocket1Accept(Sender: TObject;Socket: TCustomWinSocket);
    procedure ServerSocket1ClientDisconnect(Sender: TObject;Socket:
TCustomWinSocket);
    procedure Button4Click(Sender: TObject);
    procedure ClientSocket1Connect(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure ClientSocket1Disconnect(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure ClientSocket1Error(Sender: TObject; Socket: TCustomWinSocket;
    ErrorEvent: TErrorEvent; var ErrorCode: Integer);
  private
   { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  Server: String;
implementation
{$R *.dfm}
{Кнопка Отправить}
procedure TForm1.Button1Click(Sender: TObject);
begin{Проверка}
if Edit2.Text='' then ShowMessage('Введите свое имя.') else
if ClientSocket1.Active
then begin {Отправка сообщения на сервер}
     clientsocket1.Socket.SendText(Edit2.Text+' : '+Edit1.Text);
     {Отображение того же сообщения в поле Memo1}
     memo1.Lines.Add(edit2.Text+' : '+edit1.Text);
     Edit1.Text:='';
     end else
if ServerSocket1.Active
then begin{Отправка сообщения клиенту}
ServerSocket1.Socket.Connections[0].SendText(edit2.Text+' : '+edit1.Text);
{Отображение того же сообщения в поле Memo1}
memo1.Lines.Add(edit2.Text+' : '+edit1.Text);
Edit1.Text:='';
```

```
end; end;
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
Socket: TCustomWinSocket);
Begin {Прием сообщения}
memo1.Lines.Add(Socket.ReceiveText);
end;
{Кнопка Клиент}
procedure TForm1.Button2Click(Sender: TObject);
Begin
if ClientSocket1.Active then ClientSocket1.Active := False; {Ввод IP-адреса}
if InputQuery('Установить связь с','Псевдоним IP-адреса:',Server) then
{Проверка введенного значения}
if Length(Server) > 0 then with ClientSocket1 do
Begin {Присвоение ранее введенного значения}
Host := Server;
Active := True; end; end;
{Кнопка Сервер}
procedure TForm1.Button3Click(Sender: TObject);
begin{Активизация Сервера}
ServerSocket1.Active:=true;
StatusBar1.Panels[0].Text := 'Прослушивание…'; end;
procedure TForm1.ClientSocket1Read(Sender: TObject;
Socket: TCustomWinSocket);
Begin {Получение данных на стороне клиента}
memo1.Lines.Add(Socket.ReceiveText); end;
procedure TForm1.ServerSocket1ClientConnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
Memo2.Lines.Add('К серверу подключается клиент');
end;
procedure TForm1.ServerSocket1Accept(Sender: TObject;
Socket: TCustomWinSocket);
begin
Memo2.Lines.Add('Соединение установлено');
StatusBar1.Panels[0].Text := 'Соединено с :' + Socket.LocalAddress;
end;
procedure TForm1.ServerSocket1ClientDisconnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
StatusBar1.Panels[0].Text := 'Прослушивание соединения…';
end;
{Кнопка Закр. Клиент}
procedure TForm1.Button4Click(Sender: TObject);
begin
ClientSocket1.Active := False;
```

```
end;
procedure TForm1.ClientSocket1Connect(Sender: TObject;
Socket: TCustomWinSocket);
Begin {Информирование о соединении}
StatusBar1.Panels[0].Text := 'Соединено с адресом: '+Socket.LocalHost;
end;
procedure TForm1.ClientSocket1Disconnect(Sender: TObject;
Socket: TCustomWinSocket);
begin{Информирование о соединении}
StatusBar1.Panels[0].Text := 'Соединение закрыто';
end;
procedure TForm1.ClientSocket1Error(Sender: TObject;
Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
var ErrorCode: Integer);
begin
Memo2.Lines.Add('Ошибка соединения с сервером : '+Server);
ErrorCode := 0;
end;end.
```

2.2 Вариант раздельной организации клиентского и серверного приложений

2.2.1 Реализация серверного приложения

В среде программирования Delphi 6.0, при первом запуске автоматически создается пустая форма. А для ее создания используется пиктографическая кнопка *"New Form"*  , расположенная в левой верхней части среды Delphi.

После создания формы на нее помещаются компоненты, такие как ListBox1 (рисунок 24), Panel1 (рисунок 25), Button1 и Button2 (рисунок 26). Далее на форме необходимо поместить компонент ServerSocket1 (рисунок 27).

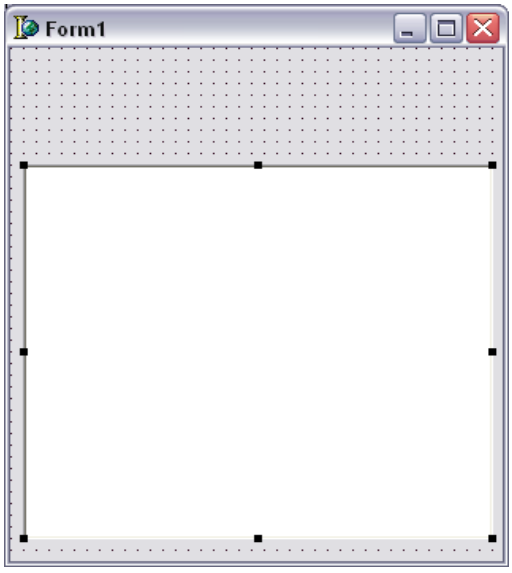


Рисунок 24 – Компонент ListBox1

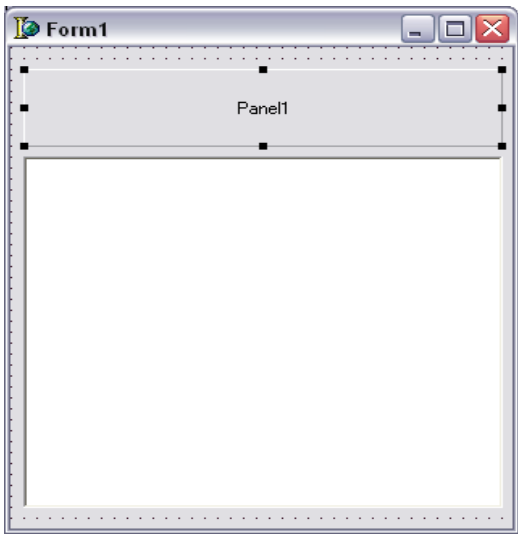


Рисунок 25 – Компонент Panel1

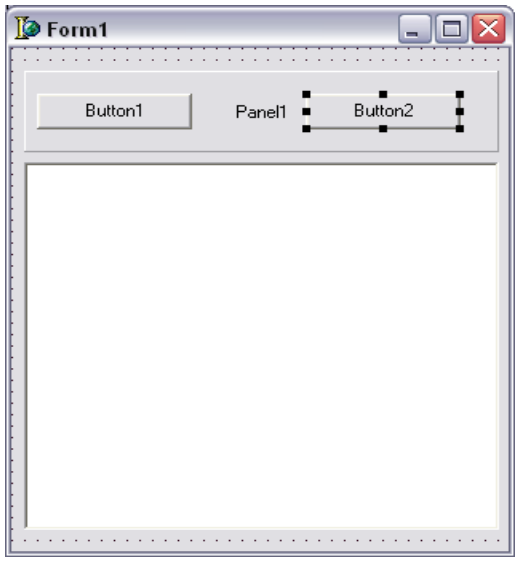


Рисунок 26 – Компоненты Button1 и Button2

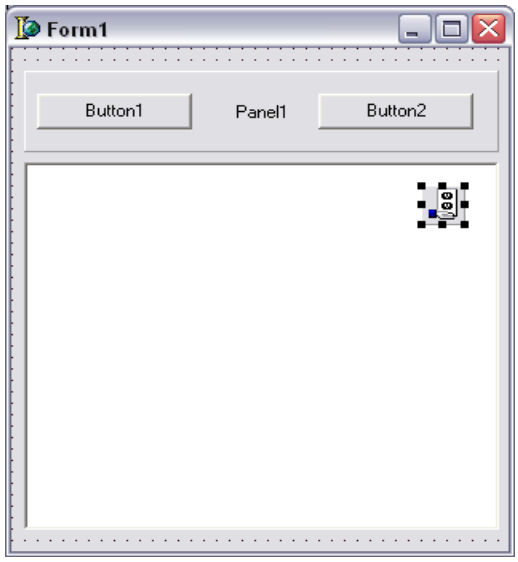


Рисунок 27 – Компонент ServerSocket1

Эти компоненты находятся на страницах “Standard” и “Internet” палитры компонентов Delphi 6.0 (рисунок 28).

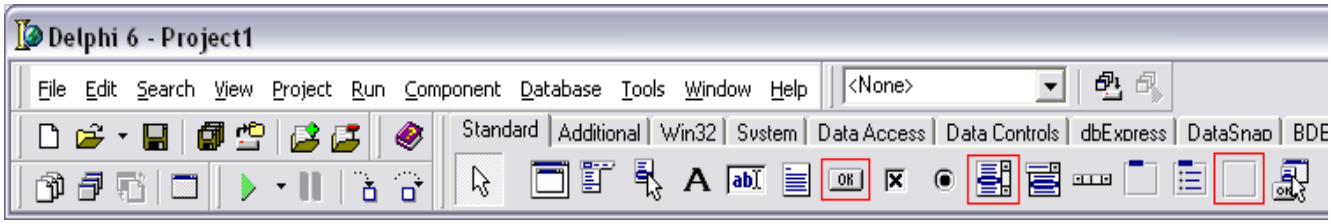


Рисунок 28 – Выбор компонентов на вкладке Standard

После того, как все компоненты расположены согласно описанию, для некоторых из них задаются определенные свойства.

В настройках (Properties) Инспектора Объектов (Object Inspector) для компонента ListBox1 задается свойство: “Align” => “alClient” (рисунок 29), после чего ListBox1 растянется на всю форму.

Далее, таким же образом выделяется компонент Panel1 и задается свойство “Align” => “alTop” (рисунок 30), после чего панель примет верхнее положение на рабочей форме.

Во всплывающем меню Инспектора указывается “Form1”, и в настройках (Properties) вручную изменяется имя Form1 в “Caption” на “Chat Server Example” (рисунок 31).

Аналогичным образом изменяются имена кнопок на панели (Panel1) для Button1 – Start и для Button2 – Stop.

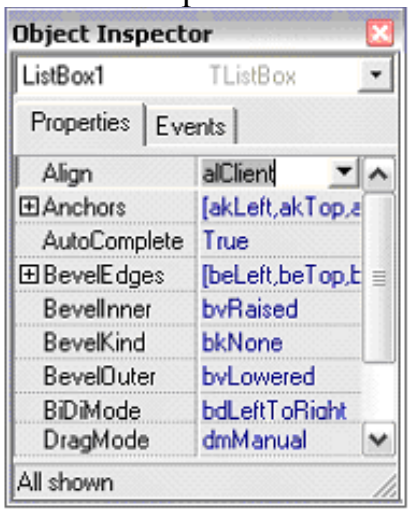


Рисунок 29 – Окно Инспектора Объектов. “Align” => “alClient”

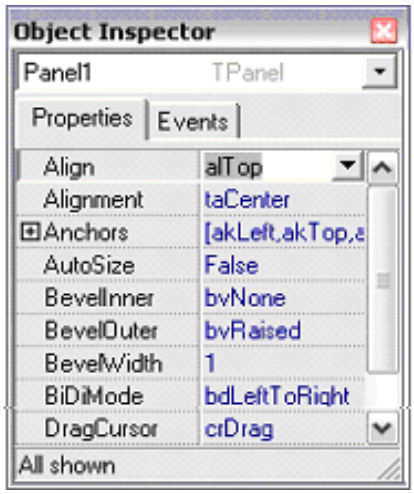


Рисунок 30 – Окно Инспектора Объектов. “Align” => “alTop”

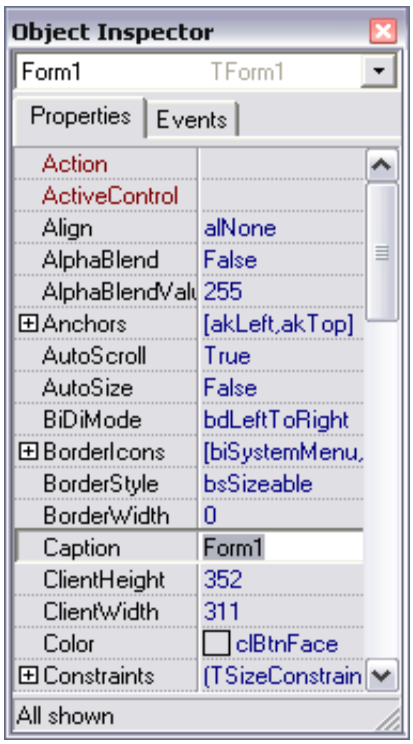


Рисунок 31 – Окно Инспектора Объектов. “Caption” => “Form1”

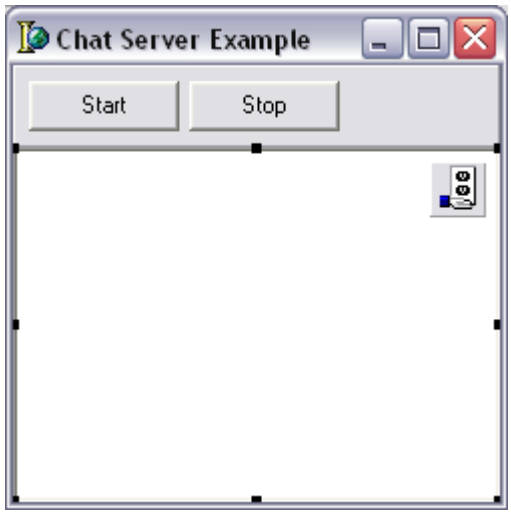


Рисунок 32 – Главное окно программы

Для реализации сервера программным кодом в начале вызывается событие кнопки “Start”. После чего в окне кода программы добавляется новая заготовочная процедура Button1Click и в нее вписывается код:

```
procedure TForm1.Button1Click(Sender: TObject);
 var s: string;
begin
 {Запрашиваем порт}
 s := InputBox('Start chat server','Enter port:','1001');
 if s = '' then
  Exit;
 {Очищаем пользовательский лист}
 ListBox1.Items.Clear;
```

```
  {Устанавливаем порт}
  ServerSocket1.Port := StrToInt(s);
  {Запускаем сервер}
  ServerSocket1.Open;
end;
```

Этот код служит для запуска серверного приложения в состояние ожидания, где определяется и устанавливается номер порта и обновляется список пользователей, далее осуществляется запуск сервера.

Аналогично выполняются все действия для кнопки “Stop”, и вписывается в него новый код:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  {Очищаем пользовательский лист и останавливаем сервер}
  ListBox1.Items.Clear;
  if ServerSocket1.Active then
   ServerSocket1.Close;
end;
```

Данный программный код служит для очищения списка пользователей, и приостановки сервера.

Далее будет необходимо написать алгоритм, который принимает сообщения с клиентских программ, обновляет список пользователей и рассылает все сообщения другим клиентским программам. Для этого необходимо воспользоваться зарезервированной процедурой в Инспекторе Объектов OnClientRead, во вкладке События (рисунок 33).

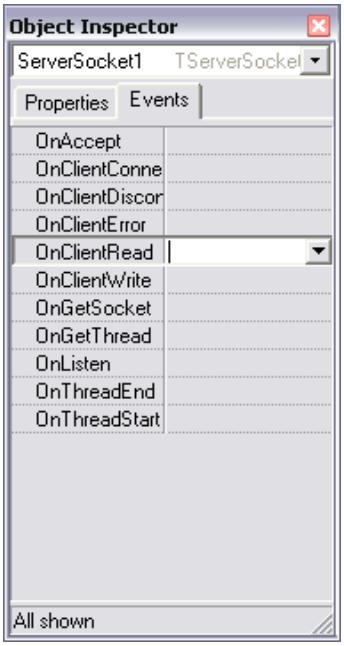


Рисунок 33 – Окно Инспектора Объектов. Events

Необходимо дважды щелкнуть левой клавишей мыши на пункте OnClientRead и в появившемся окне требуется ввести алгоритмическую процедуру, которая может иметь следующий вид:

```
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
 var s: string;
     i: Integer;
begin
  {сохраняем в s присланную нам строку}
  s := Socket.ReceiveText;
  {Если кто-то прислал нам свое имя}
  if Copy(s,1,2) = '#N' then begin Delete(s,1,2);
   {Добавляем его в пользовательский лист}
   ListBox1.Items.Add(s);
   {Записываем в s команду для посылки нового списка пользователей}
   s := '#U';
   for i := 0 to ListBox1.Items.Count-1 do
    s := s+ListBox1.Items[i]+';';
   {...и рассылаем этот список всем клиентам}
   for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
    ServerSocket1.Socket.Connections[i].SendText(s);
   Exit;
  end;
    {Если кто-то кинул сообщение - рассылаем его всем клиентам}
  if (Copy(s,1,2) = '#M')or(Copy(s,1,2) = '#P') then begin
   for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
    ServerSocket1.Socket.Connections[i].SendText(s);
   Exit;
  end;
end;
```

Для отсоединения от сервера, создается еще одна процедура, которая также служит для идентификации, и проверки всех подсоединенных пользователей (клиентов). Такая процедура создается аналогично Инспектору событий, и будет иметь вид:

```
procedure TForm1.ServerSocket1ClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
 var i: Integer;
begin
  {Кто-то присоединился или отсоединился? Запрашиваем у всех
   пользователей их имена}
  ListBox1.Items.Clear;
  for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
   ServerSocket1.Socket.Connections[i].SendText('#N');
end;
```

На данном этапе серверная часть программы полностью реализована. Далее необходимо сохранить проект. Для этого необходимо в верхней части среды Delphi выбрать кнопку “Save All” или нажать на горячие клавиши Shift+Ctrl+S.

2.2.2 Реализация клиентского приложения

Для создания клиентской программы необходимо запустить Delphi 6.0. Затем создается форма, и на ней размещаются необходимые компоненты: Panel1 (рисунок 34), для которого необходимо присвоить свойство “Align” => “alTop”, компонент ClientSocket1 (рисунок 35).

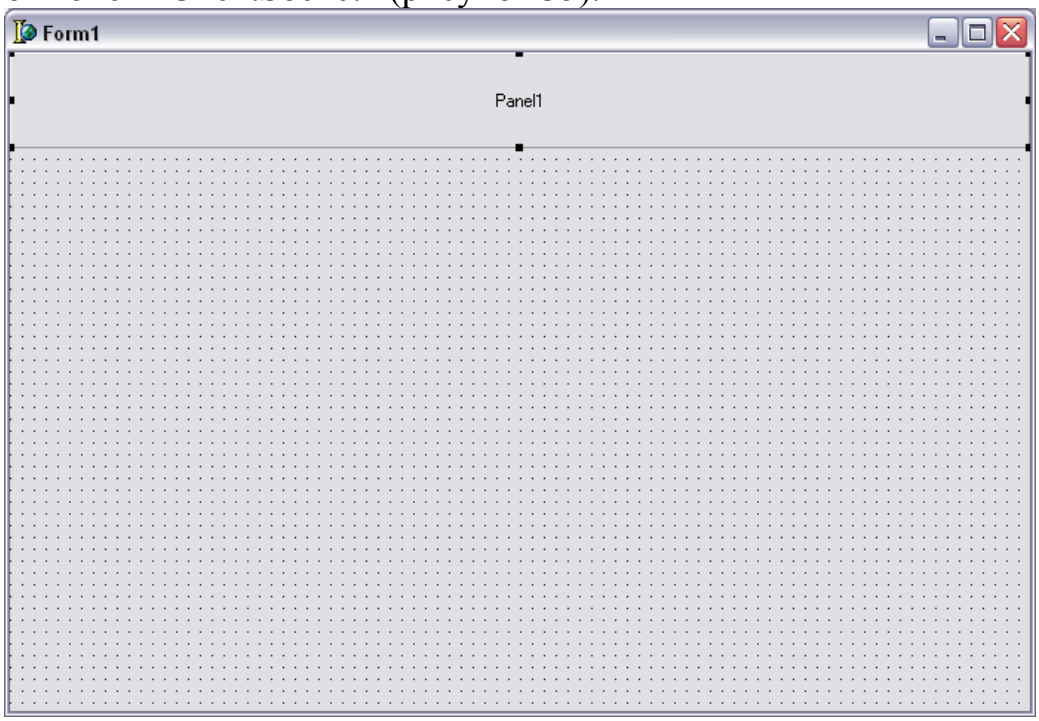


Рисунок 34 – Компонент Panel1

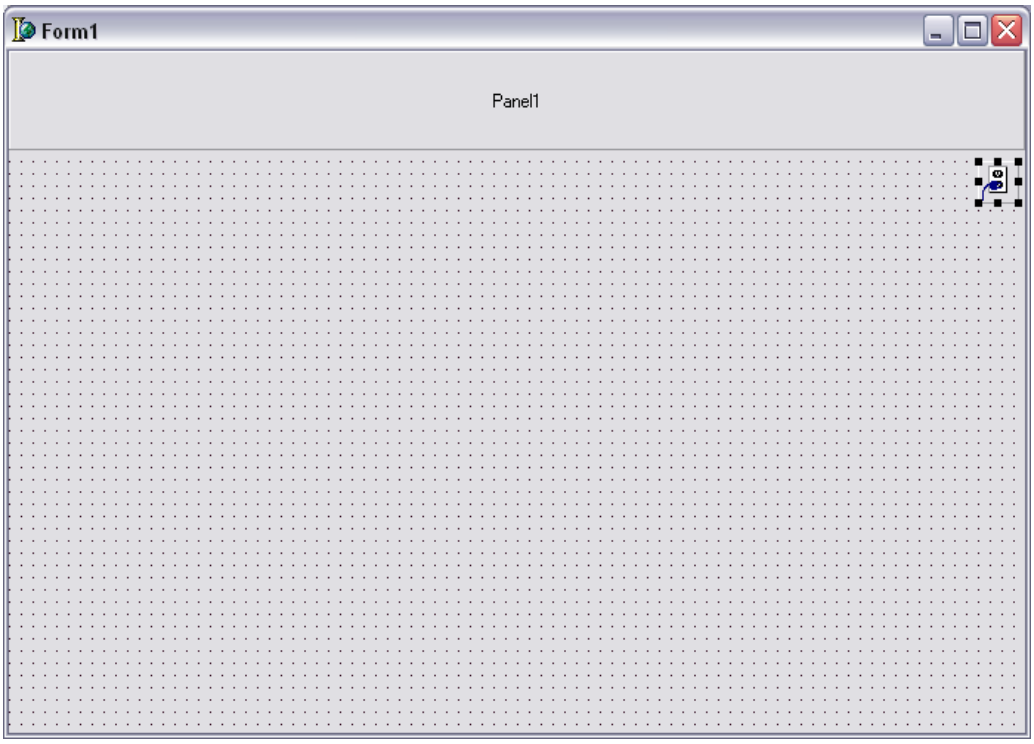


Рисунок 35 – Компонент ClientSocket1

Далее на форму необходимо поместить три кнопки типа TButton, один CheckBox1 и еще один Edit1 (рисунок 36).

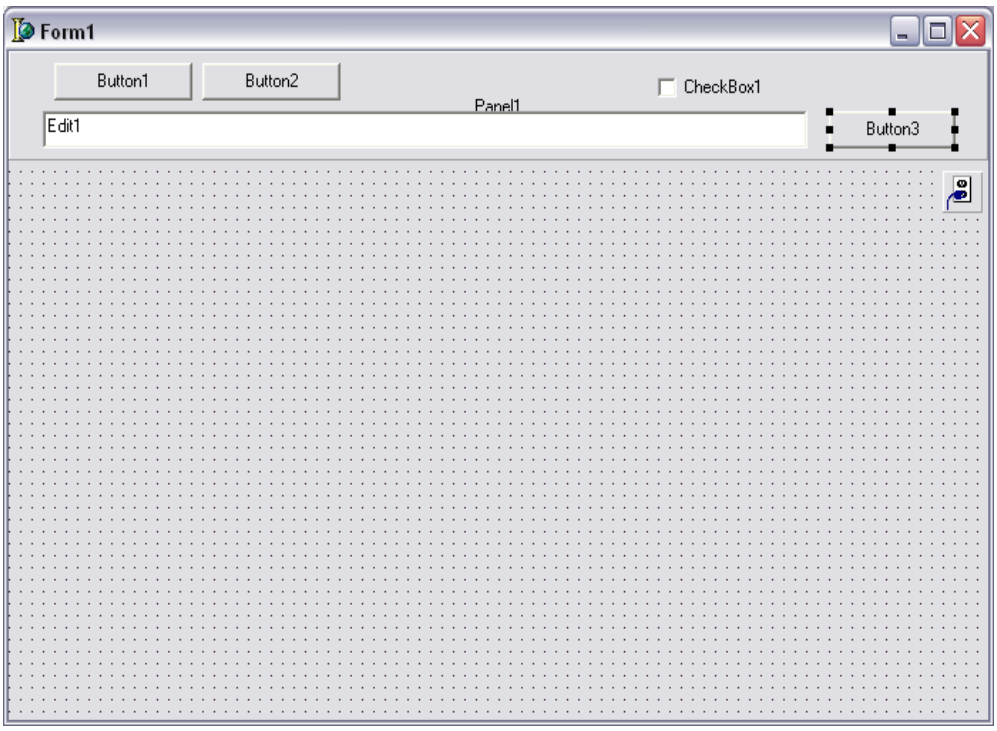


Рисунок 36 – Компоненты на Panel1

После этого на форму необходимо поместить еще одну Panel2 со свойствами “Align” => “alBottom”, слева компонент Memo1 и справа ListBox1.

Последним двум компонентам Memo1 и ListBox1 необходимо задать свойство “Align” => “alClient” (рисунок 37).

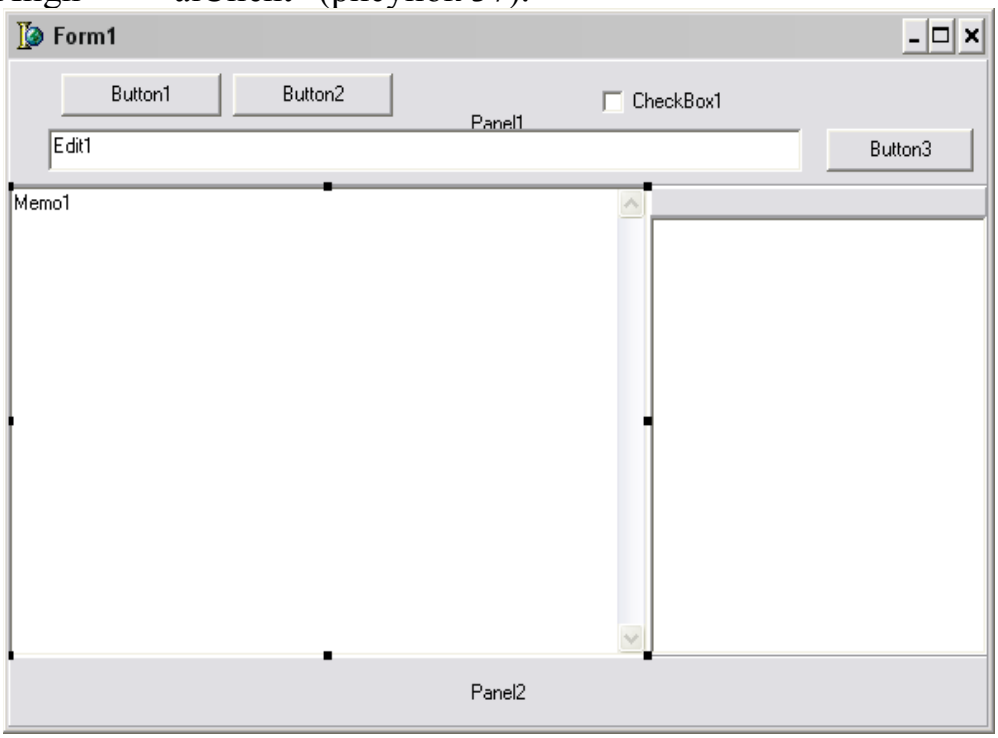


Рисунок 37 – Компоненты на Form1

Аналогичным способом, описанным ранее при реализации серверного приложения, в Инспекторе Объектов (Object Inspector) необходимо изменить имена для компонентов, после чего главное окно программы примет вид как представлено на рисунке 38.

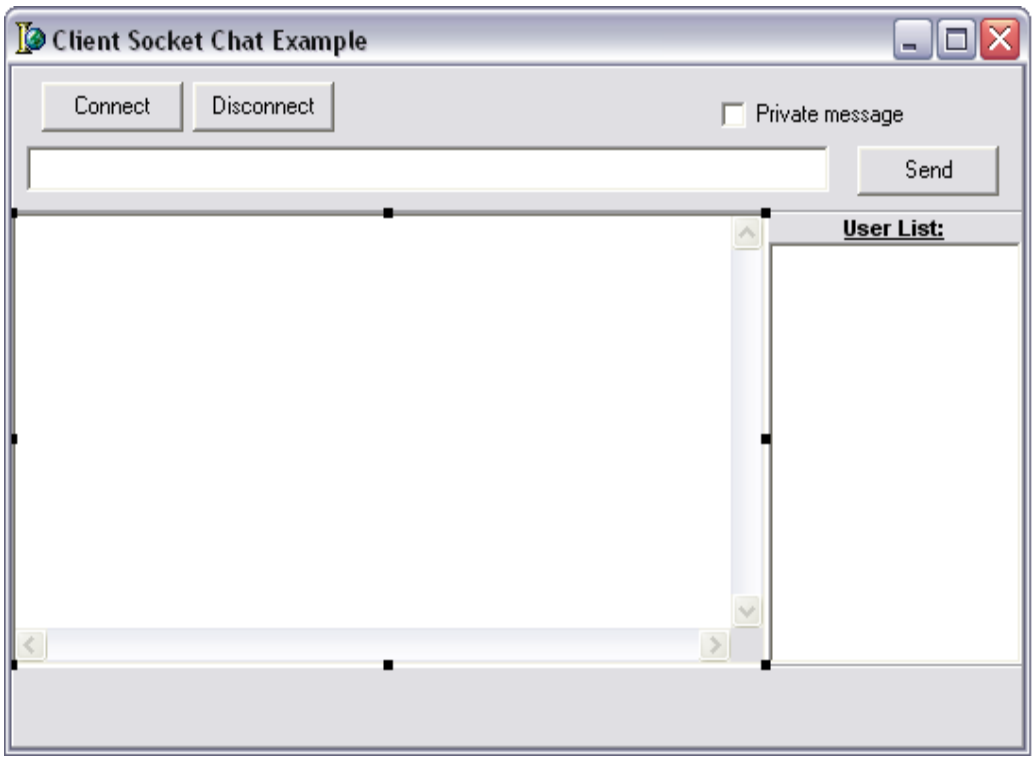


Рисунок 38 – Главное окно программы для (Client)

Для реализации клиента программным кодированием сначала вызывается событие кнопки “Connect”. После чего в окне кода программы добавляется новая заготовочная процедура, в которую вписывается код программы:

```
procedure TForm1.Button2Click(Sender: TObject);
 var do_connect: Boolean;
    host,port: string;
begin
 {Показываем окно установки соединения с сервером}
 Form2 := TForm2.Create(Application);
 {do_connect = True, если была нажата кнопка Connect}
 do_connect := (Form2.ShowModal = mrOk);
 {заполнение переменных до того, как мы уничтожим форму}
 host := Form2.Edit1.Text;
 port := Form2.Edit2.Text;
 nickname := Form2.Edit3.Text;
 {Уничтожаем форму}
 Form2.Free;
 {Если была нажата кнопка Cancel, то уходим отсюда}
 if not do_connect then
  Exit;
 {Если соединение уже установлено, то обрываем его}
 if ClientSocket1.Active then
  ClientSocket1.Close;
 {Устанавливаем свойства Host и Port}
 ClientSocket1.Host := host;
```

```
  ClientSocket1.Port := StrToInt(port);
  {Пытаемся соединиться}
  ClientSocket1.Open;
end;
```

Данная процедура служит для подключения к серверу, т.е. вводится Ник, порт, IP-адрес или HostName (рисунок 39), затем происходит ожидание ответа сервера, и если все было введено правильно и такого больше подключения не существует, то клиент подсоединяется к серверу.

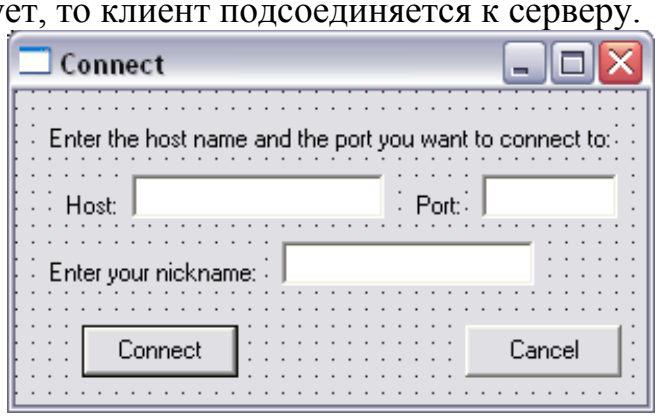


Рисунок 39 – Окно для соединения с сервером

Затем в кнопке “Disconnect”, аналогично создается процедура, ее описание приведено ниже:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  {Закрываем соединение (если оно установлено)}
  if ClientSocket1.Active then
    ClientSocket1.Close;
end;
```

После описания процедуры “Disconnect”, описывается следующая процедура, для выполнения кнопки “Send”:

```
procedure TForm1.Button1Click(Sender: TObject);
 var s: string;
begin
  {Если мы хотим послать приватное сообщение, но не выбрали адресата -
    выведется замечание и завершится обработка}
  if (CheckBox1.Checked)and(ListBox1.ItemIndex < 0) then begin
    ShowMessage('At first you should select the user in the User List!');
    Exit;
  end;
  {Если это приватное сообщение}
  if CheckBox1.Checked then
     s := '#P'+ListBox1.Items[ListBox1.ItemIndex]+';' {добавляем спец.команду и адресат}
  else {А если не приватное}
    s := '#M'; {Просто спец.команду}
  {Добавляем наше имя (от кого) и само сообщение}
```

```
  s := s+nickname+';'+Edit1.Text;
  {Посылаем все сообщение по сокету}
  ClientSocket1.Socket.SendText(s);
  {И снова ждем ввода в уже чистом TEdit-е}
  Edit1.Text := '';
  ActiveControl := Edit1;
end;
```

Это процедура служит для отправки сообщения, через серверную программу всем клиентам, либо определенному клиенту, через “Private message”.

Далее описывается еще несколько процедур влияющих на правильную работу сервер-сокета. Для их создания в Инспекторе Объектов (Object Inspector), необходимо использовать события компонента ServerSocket1. Описания этих процедур, приведены ниже:

```
procedure TForm1.ClientSocket1Error(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  {Если произошла ошибка, выводим ее код в Memo1}
  {Insert вставляет строку в указанную позицию (в данном случае - 0 - в начало)}
  Memo1.Lines.Insert(0,'Socket error ('+IntToStr(ErrorCode)+')');
end;
procedure TForm1.ClientSocket1Lookup(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Сообщаем о том, что идет поиск хоста}
  Memo1.Lines.Insert(0,'Looking up for server...');
end;
procedure TForm1.ClientSocket1Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {соединяемся...}
  Memo1.Lines.Insert(0,'connecting...');
end;
procedure TForm1.ClientSocket1Connect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {соединились!}
  Memo1.Lines.Insert(0,'connected!');
end;
procedure TForm1.ClientSocket1Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {отсоединились :(}
  Memo1.Lines.Insert(0,'disconnected');
end;
```

```
procedure TForm1.ClientSocket1Read(Sender: TObject;
  Socket: TCustomWinSocket);
 var s,from_,to_: string;
begin
  {присваиваем s полученную от сервера строку}
  s := Socket.ReceiveText;
  {Если сервер посылает нам User List}
  if Copy(s,1,2) = '#U' then begin
   Delete(s,1,2);
   {Чистим ListBox1}
   ListBox1.Items.Clear;
    {Добавляем по одному пользователю в список. Имена пользователей разде-
лены знаком ";"}
   while Pos(';',s) > 0 do begin
    ListBox1.Items.Add(Copy(s,1,Pos(';',s)-1));
    Delete(s,1,Pos(';',s));
   end;
   Exit;
  end;
    {Если нам прислали общее сообщение (видимое для всех пользователей)}
  if Copy(s,1,2) = '#M' then begin
   Delete(s,1,2);
   {Добавляем его в Memo1}
   Memo1.Lines.Insert(0,Copy(s,1,Pos(';',s)-1)+'> '+
                      Copy(s,Pos(';',s)+1,Length(s)-Pos(';',s)));
   Exit;
  end;
  {Если нам прислали запрос на наше имя пользователя}
  if Copy(s,1,2) = '#N' then begin
   {Посылаем ответ}
   Socket.SendText('#N'+nickname);
   Exit; end;
    {Если нам прислали приватное сообщение (или не нам)}
  if Copy(s,1,2) = '#P' then begin
   Delete(s,1,2);
   {Выделяем в to_ - кому оно предназначено}
   to_ := Copy(s,1,Pos(';',s)-1);
   Delete(s,1,Pos(';',s));
   {Выделяем в from_ - кем отправлено}
   from_ := Copy(s,1,Pos(';',s)-1);
   Delete(s,1,Pos(';',s));
   {Если оно для нас, или написано нами - добавляем в Memo1}
   if (to_ = nickname)or(from_ = nickname) then
    Memo1.Lines.Insert(0,from_+' (private) > '+s); Exit; end;
end;
```

Эти все процедуры служат для соединения, отсоединения, чтения, возврата ошибки, при отправке сообщения к клиентам или клиенту. В них описан алгоритм обработки запросов и отправки сообщений адресату.

После выполнения всех действий проект необходимо сохранить на диске.

2.2.3 Руководство по эксплуатации программного средства

Перед тем, как начать работу с программынм средсвом “Chat”, необходимо запустить серверную программу (рисунок 40).

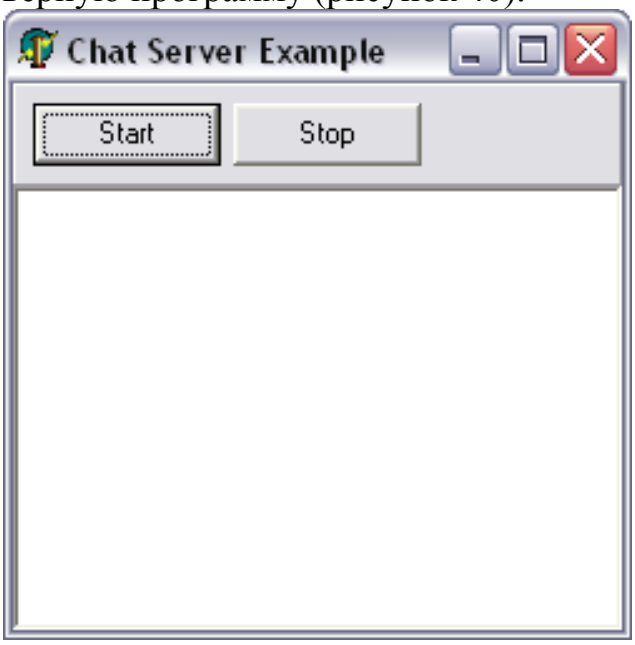


Рисунок 40 – Главное окно серверной программы

В появившемся главном окне необходимо щелкнуть левой кнопкой мыши по кнопке “Start”, затем появляется новое диалоговое окно “Start chat server”, в котором необходимо указать порт (рисунок 41).

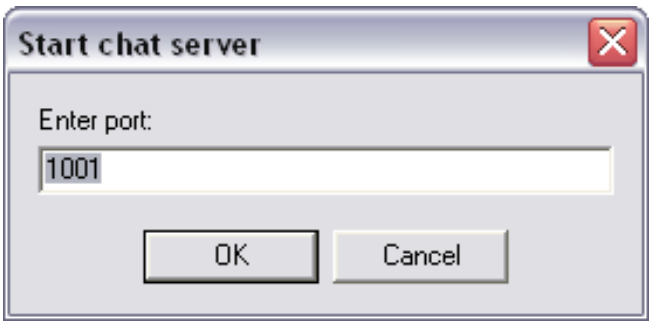


Рисунок 41 – Диалоговое окно для указания порта

Здесь необходимо щелкнуть по кнопке “OK” и подождать подключения клиентов. Если необходимо приостановить сервер, аналогично нужно щелкнуть по кнопке “Stop”.

После нажатия кнопки “Cancel” очищается список пользователей, и соединение между клиентами (пользователями) прекращается. Далее запускается клиентская программа (рисунок 42).

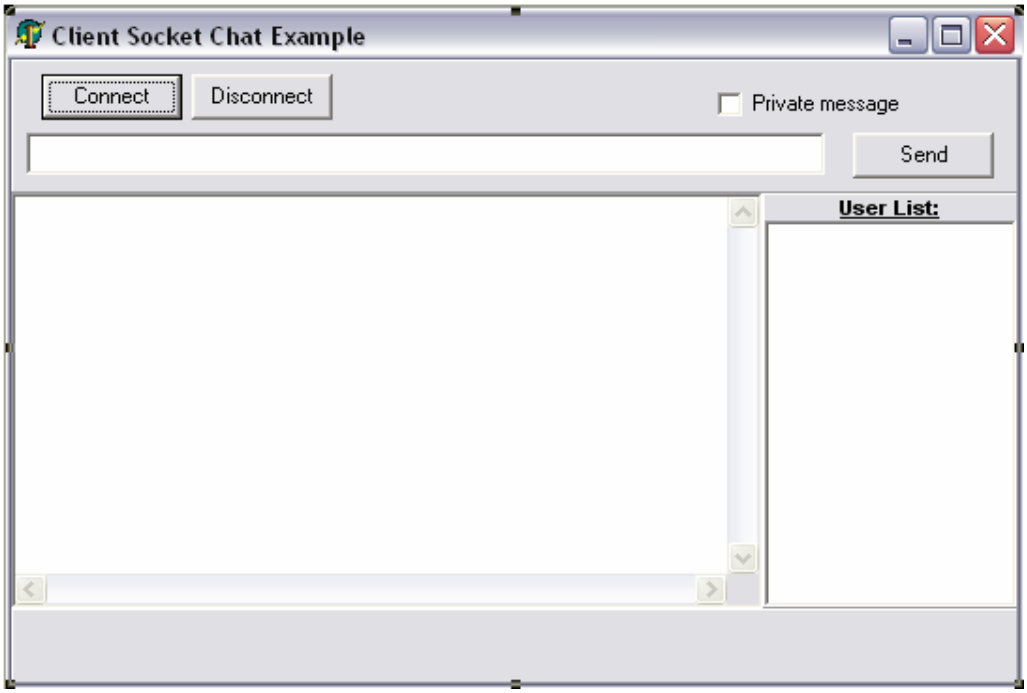


Рисунок 42 - Главное окно клиентской программы

В появившемся главном окне, необходимо щелкнуть по кнопке “Connect”, после чего появляется новое диалоговое окно (рисунок 43), в котором нужно указать имя пользователя (Ник - Nick), IP-адрес или HostName того узла, где запущена серверная программа, например 192.168.0.1 или OSU-Server, или LocalHost. Далее указывается порт, который соответствует забронированному в серверной программе, и происходит подсоединение по щелчку по кнопке “Connect” (рисунок 44).

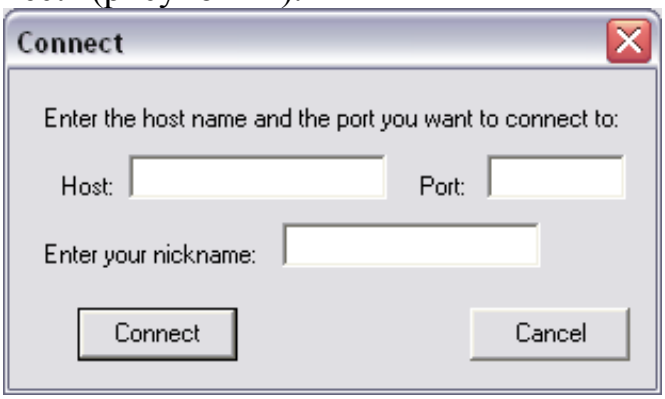


Рисунок 43 – Диалоговое окно для соединения пользователя с сервером

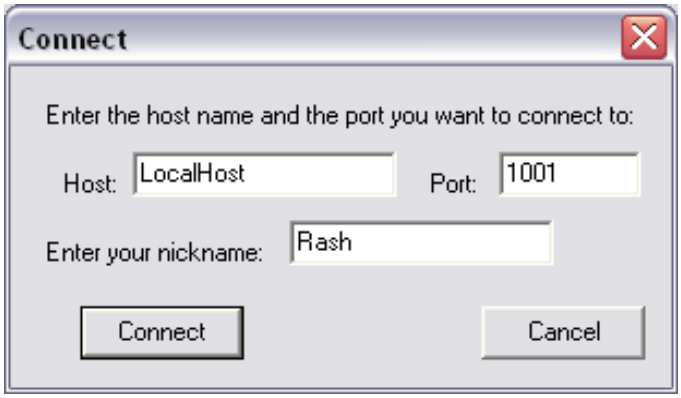


Рисунок 44 – Диалоговое окно для соединения пользователя с сервером

После подсоединения, имя пользователя (NickName), отобразится в списке пользователей серверной программы (рисунок 45).

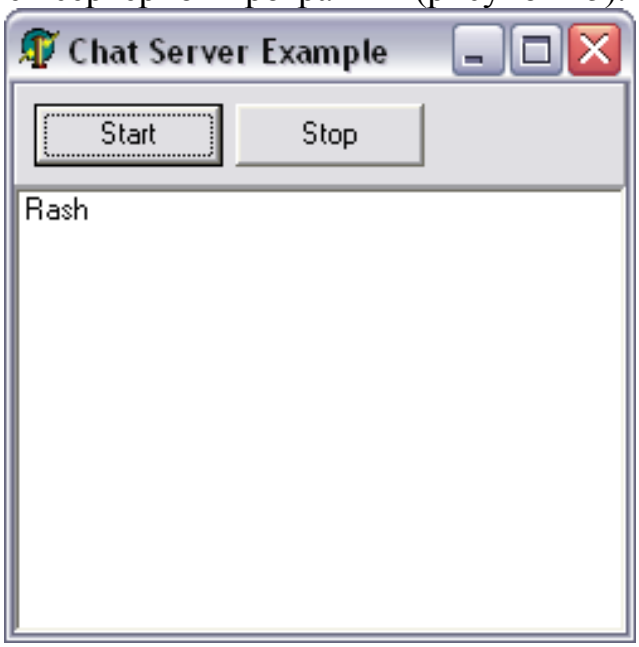


Рисунок 45 - Главное окно серверной программы

Также и в текстовом поле (Memo1) клиентской программы, отобразится, что клиентская программа соединилась, и ждет подсоединения других пользователей через сервер (рисунок 46).

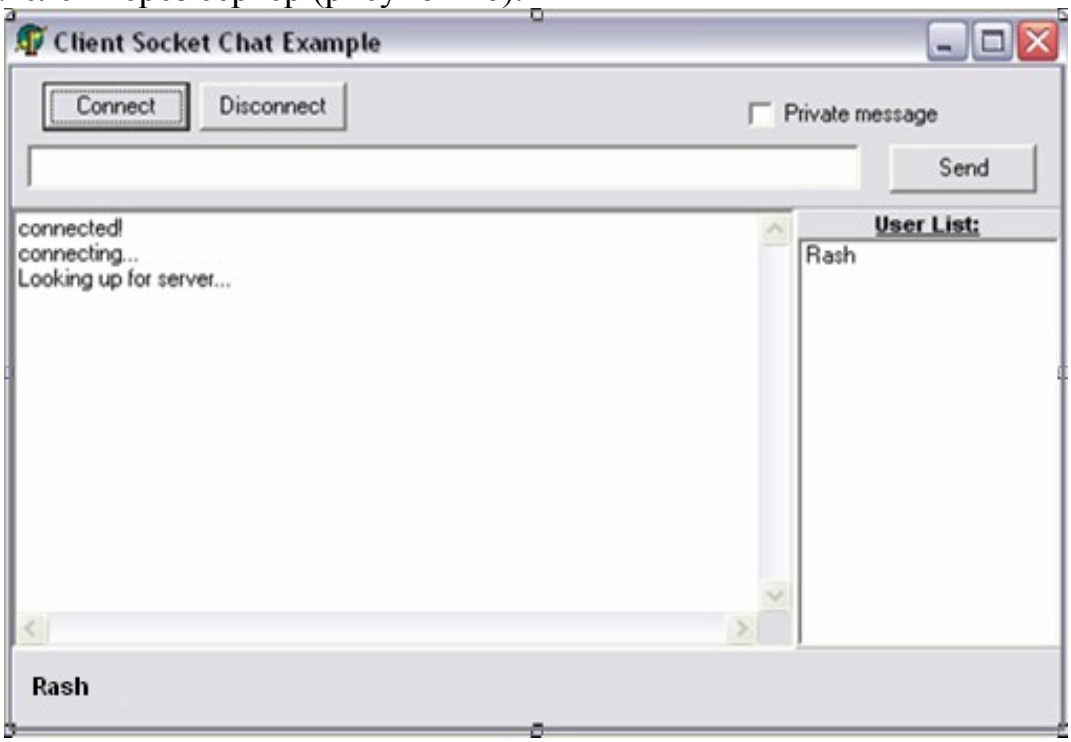


Рисунок 46 - Главное окно клиентской программы

Чтобы увидеть, как передаются сообщения между пользователями, необходимо подсоединить еще пользователей к серверу. Для этого необходимо произвести дополнительный запуск клиентских программ, и аналогично соединить их с сервером (рисунки 47, 48).

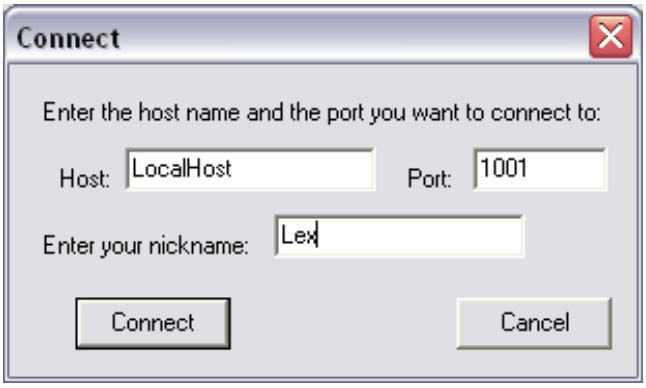


Рисунок 47 – Диалоговое окно для соединения пользователя (Lex) с сервером

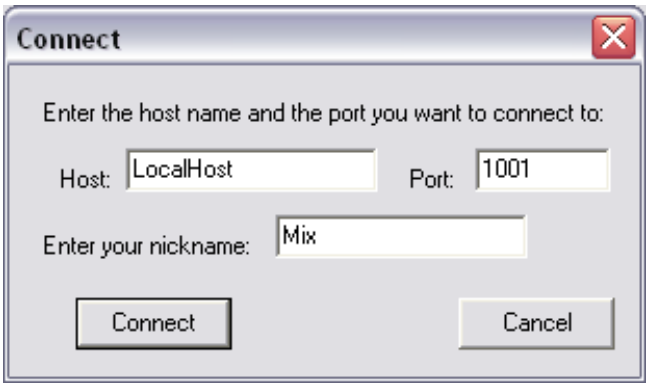


Рисунок 48 – Диалоговое окно для соединения пользователя (Mix) с сервером

После их соединения список пользователей в серверной программе пополнится (рисунок 49).

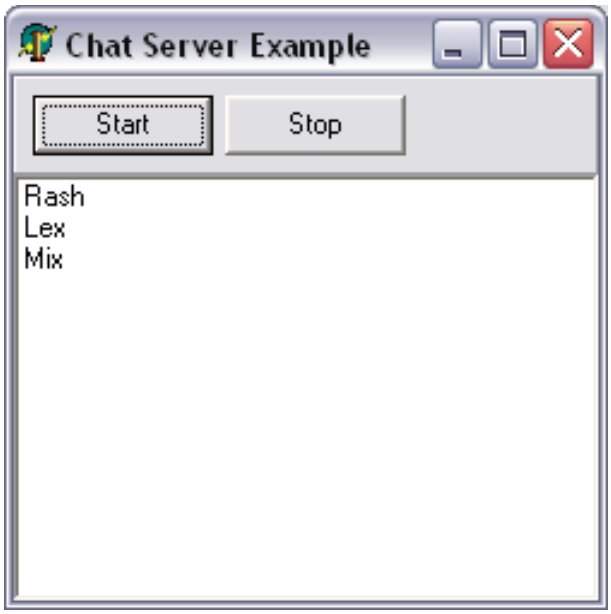


Рисунок 49 - Главное окно серверной программы

Для того, чтобы отправить сообщение от одного клиента (пользователя) к другому, необходимо выполнить следующие действия: в длинной пустой строке (Edit1), написать текст, затем щелкнуть по кнопке отправить “Send”, или нажать ввод (клавишу Enter) (рисунок 50).

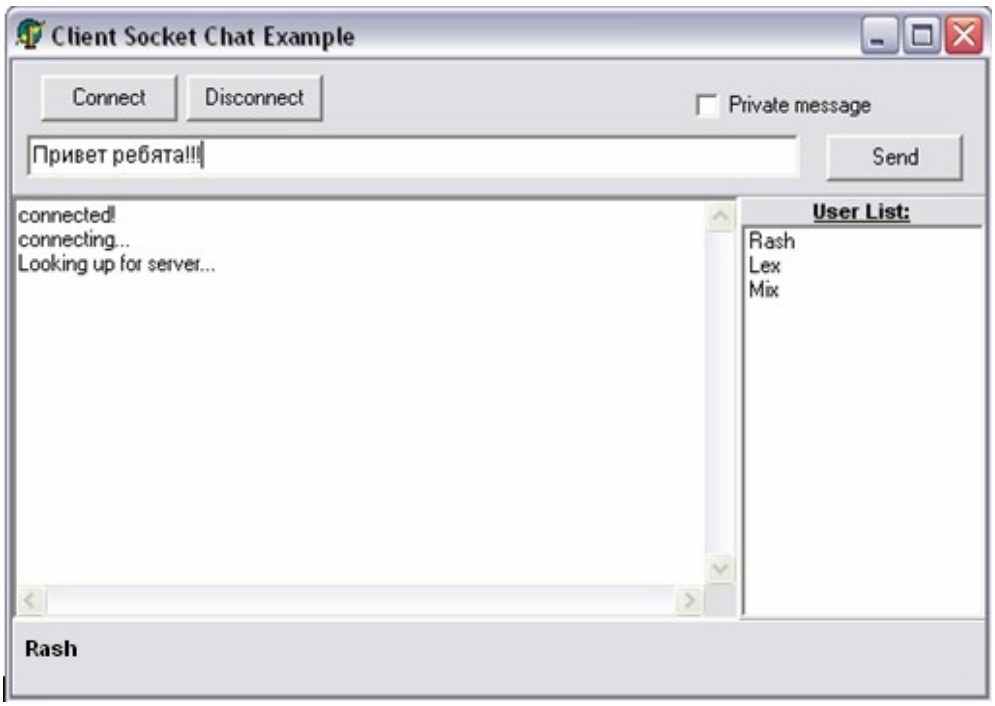


Рисунок 50 - Главное окно клиентской программы

После этого действия сообщение отправится всем клиентам (рисунок 51).

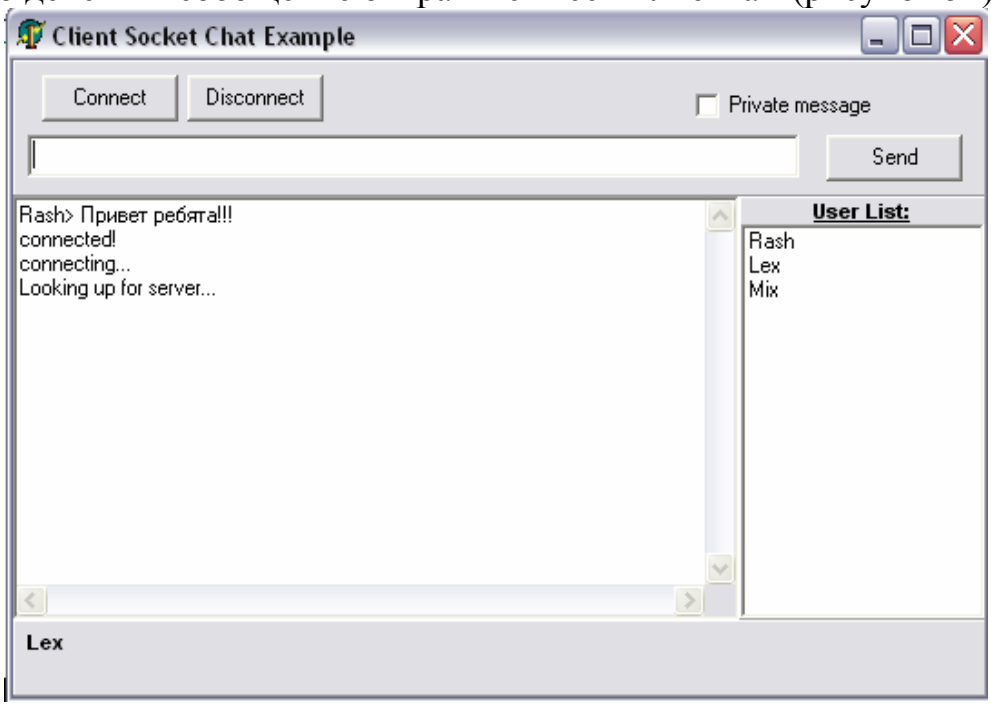


Рисунок 51 - Главное окно клиентской программы

Если же нужно отправить приватное сообщение определенному пользователю, можно воспользоваться скрытой отправкой для этого необходимо:

- необходимо поставить галочку в “Private message”;
- затем требуется выделить Ник в списке всех пользователей клиентской программы;
- щелкнуть по кнопке отправить “Send”.

Данная ситуация представлена на рисунках 52, 53.

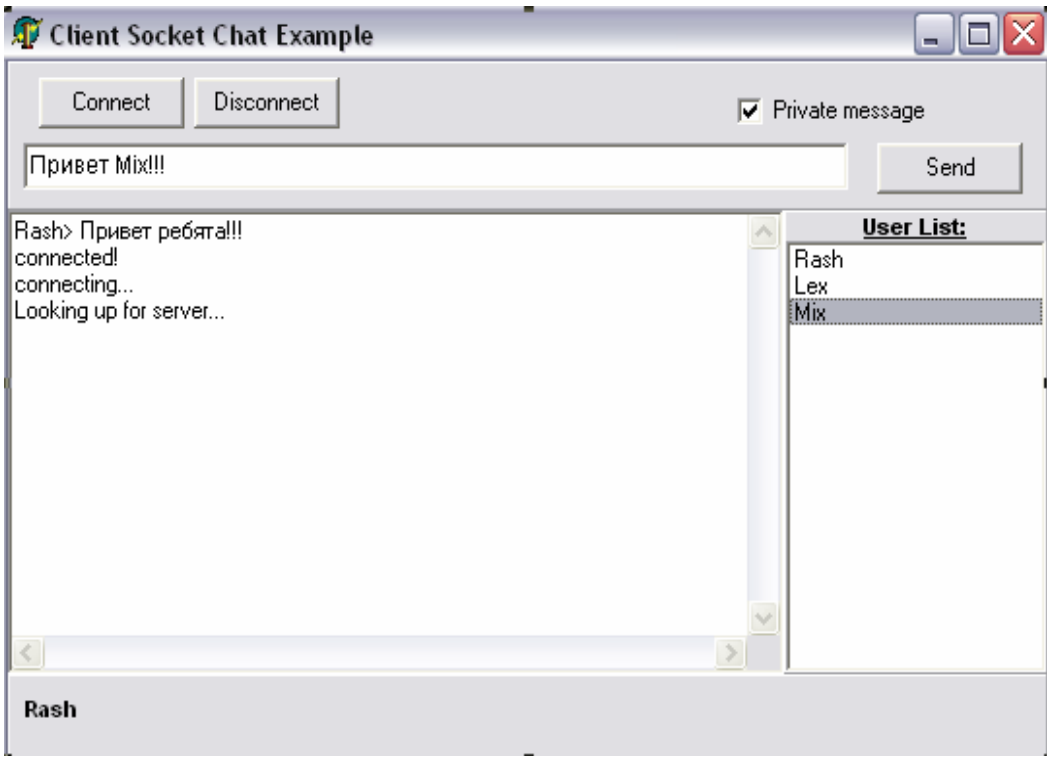


Рисунок 52 - Главное окно клиентской программы

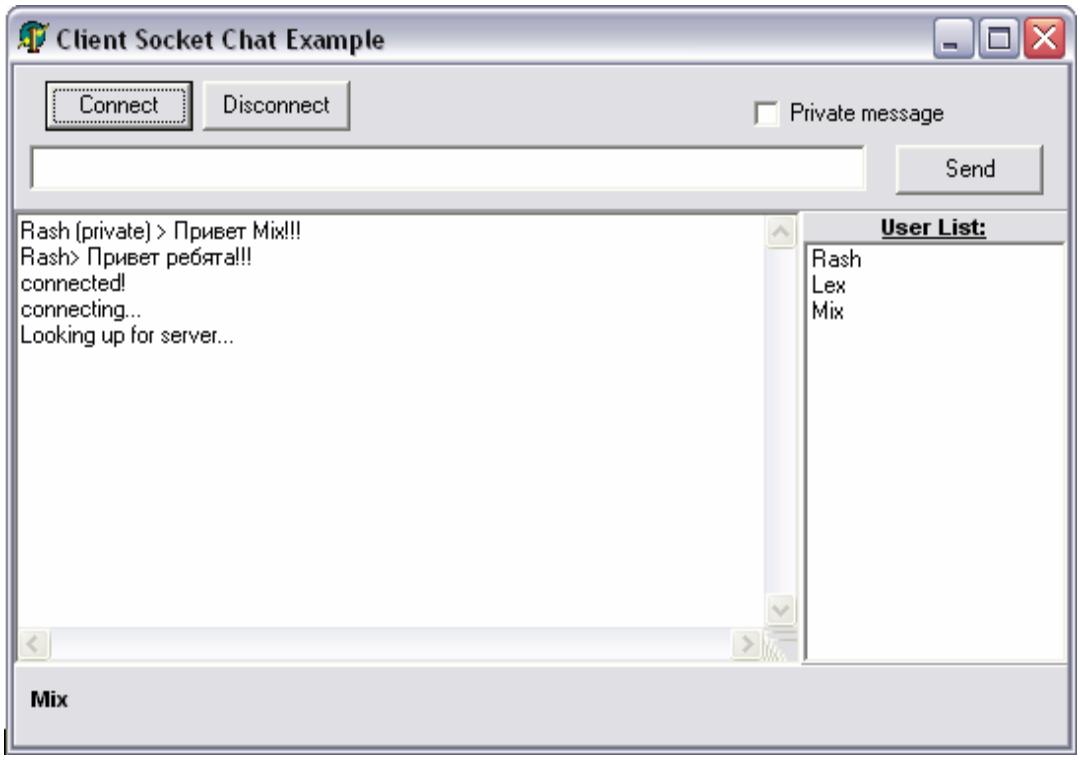


Рисунок 53 - Главное окно клиентской программы

Все отправляемые и получаемые сообщения будут выводиться в большой текстовой области, расположенной в центральной части рабочей формы, в левой ее части (Memo1).

Если же необходимо отсоединится, нужно навести указатель мыши на кнопку “Disconnect” и нажать на ней левой клавишей мыши или закрыть программу, воспользовавшись стандартной кнопкой “Закрыть”.

2.2.4 Программный код

{Код серверной программы:}

```
 program srv_ex;
uses
  Forms,
  srvmain in 'srvmain.pas' {Form1};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
unit srvmain;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, ScktComp;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Button1: TButton;
    Button2: TButton;
    ListBox1: TListBox;
    ServerSocket1: TServerSocket;
    procedure Button1Click(Sender: TObject); procedure Button2Click(Sender: TObject);
    procedure ServerSocket1ClientRead(Sender: TObject; Socket: TCustomWinSocket);
     procedure ServerSocket1ClientDisconnect(Sender: TObject; Socket: TCustomWinSock-
et);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
 var s: string;
begin
  {Запрашиваем порт}
  s := InputBox('Start chat server','Enter port:','1001');
  if s = '' then
```

```
   Exit;
  {Чистим пользовательский лист}
  ListBox1.Items.Clear;
  {Устанавливаем порт}
  ServerSocket1.Port := StrToInt(s);
  {Запускаем сервер}
  ServerSocket1.Open;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  {Чистим пользовательский лист и останавливаем сервер}
  ListBox1.Items.Clear;
  if ServerSocket1.Active then
   ServerSocket1.Close;
end;
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
 var s: string;  i: Integer;
begin
  {сохраняем в s присланную нам строку}
  s := Socket.ReceiveText;
  {Если кто-то прислал нам свое имя}
  if Copy(s,1,2) = '#N' then begin
   Delete(s,1,2);
   {Добавляем его в пользовательский лист}
   ListBox1.Items.Add(s);
   {Записываем в s команду для посылки нового списка пользователей}
   s := '#U';
   for i := 0 to ListBox1.Items.Count-1 do
    s := s+ListBox1.Items[i]+';';
   {...и рассылаем этот список всем клиентам}
   for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
    ServerSocket1.Socket.Connections[i].SendText(s);
   Exit;
  end;
  {Если кто-то кинул сообщение - рассылаем его всем клиентам}
  if (Copy(s,1,2) = '#M')or(Copy(s,1,2) = '#P') then begin
   for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
    ServerSocket1.Socket.Connections[i].SendText(s);
   Exit;   end; end;
procedure TForm1.ServerSocket1ClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
 var i: Integer;
begin
  {Кто-то присоединился или отсоединился? Запрашиваем у всех
```

```
   пользователей их имена}
  ListBox1.Items.Clear;
  for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
ServerSocket1.Socket.Connections[i].SendText('#N');
end;
end.

{Код клиентской программы:}

program chat_ex;
uses
  Forms,
  main in 'main.pas' {Form1},
  conn in 'conn.pas' {Form2};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
unit conn;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    Edit2: TEdit;
    Label4: TLabel;
    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
Form2: TForm2;
implementation
{$R *.DFM}
```

```
end.
unit main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, ScktComp;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Button1: TButton;
    Edit1: TEdit;
    Panel3: TPanel;
    Label2: TLabel;
    Label3: TLabel;
    ListBox1: TListBox;
    Button2: TButton;
    Button3: TButton;
    CheckBox1: TCheckBox;
    Memo1: TMemo;
    ClientSocket1: TClientSocket;
    procedure Button2Click(Sender: TObject); procedure Button3Click(Sender: TObject);
    procedure ClientSocket1Error(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
    procedure ClientSocket1Lookup(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket1Connecting(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket1Connect(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket1Disconnect(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket1Read(Sender: TObject; Socket: TCustomWinSocket);
      procedure Button1Click(Sender: TObject); procedure Edit1KeyDown(Sender: TObject;
var Key: Word;
     Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  nickname: string;
implementation
uses conn;
{$R *.DFM}
procedure TForm1.Button2Click(Sender: TObject);
```

```
 var do_connect: Boolean;
     host,port: string;
begin
  {Показываем окно установки соединения с сервером}
  Form2 := TForm2.Create(Application);
  {do_connect = True, если была нажата кнопка Connect}
  do_connect := (Form2.ShowModal = mrOk);
  {заполнение переменных до того, как мы уничтожим форму}
  host := Form2.Edit1.Text; port := Form2.Edit2.Text; nickname := Form2.Edit3.Text;
  {Уничтожаем форму}
  Form2.Free;
  {Если была нажата кнопка Cancel, то уходим отсюда}
  if not do_connect then
   Exit;
  {Если соединение уже установлено, то обрываем его}
  if ClientSocket1.Active then
   ClientSocket1.Close;
  {Устанавливаем свойства Host и Port}
  ClientSocket1.Host := host;
  ClientSocket1.Port := StrToInt(port);
  {Пытаемся соединиться}
  ClientSocket1.Open;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  {Закрываем соединение (если оно установлено)}
  if ClientSocket1.Active then
   ClientSocket1.Close;
end;
procedure TForm1.ClientSocket1Error(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  {Если произошла ошибка, выводим ее код в Memo1}
  {Insert вставляет строку в указанную позицию (в данном случае - 0 - в начало)}
  Memo1.Lines.Insert(0,'Socket error ('+IntToStr(ErrorCode)+')');
end;
procedure TForm1.ClientSocket1Lookup(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Сообщаем о том, что идет поиск хоста}
  Memo1.Lines.Insert(0,'Looking up for server...');
end;
procedure TForm1.ClientSocket1Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
```

```
begin
  {соединяемся...}
  Memo1.Lines.Insert(0,'connecting...');
end;
procedure TForm1.ClientSocket1Connect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {соединились!}
  Memo1.Lines.Insert(0,'connected!');
end;
procedure TForm1.ClientSocket1Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {отсоединились}
  Memo1.Lines.Insert(0,'disconnected');
end;
procedure TForm1.ClientSocket1Read(Sender: TObject;
  Socket: TCustomWinSocket);
 var s,from_,to_: string;
begin
  {присваиваем s полученную от сервера строку}
  s := Socket.ReceiveText;
  {Если сервер посылает нам User List}
  if Copy(s,1,2) = '#U' then begin
   Delete(s,1,2);
   {Чистим ListBox1}
   ListBox1.Items.Clear;
    {Добавляем по одному пользователю в список. Имена пользователей разделены
знаком ";"}
   while Pos(';',s) > 0 do begin
    ListBox1.Items.Add(Copy(s,1,Pos(';',s)-1));
    Delete(s,1,Pos(';',s));
   end;
   Exit;
  end;
  {Если нам прислали общее сообщение (видимое для всех пользователей)}
  if Copy(s,1,2) = '#M' then begin
   Delete(s,1,2);
   {Добавляем его в Memo1}
   Memo1.Lines.Insert(0,Copy(s,1,Pos(';',s)-1)+'> '+
                      Copy(s,Pos(';',s)+1,Length(s)-Pos(';',s)));
   Exit;
  end;
  {Если нам прислали запрос на наше имя пользователя}
  if Copy(s,1,2) = '#N' then begin
```

```
   {Посылаем ответ}
   Socket.SendText('#N'+nickname);
   Exit;
  end;
  {Если нам прислали приватное сообщение (или не нам )}
  if Copy(s,1,2) = '#P' then begin Delete(s,1,2);
   {Выделяем в to_ - кому оно предназначено}
   to_ := Copy(s,1,Pos(';',s)-1);
   Delete(s,1,Pos(';',s));
   {Выделяем в from_ - кем отправлено}
   from_ := Copy(s,1,Pos(';',s)-1); Delete(s,1,Pos(';',s));
   {Если оно для нас, или написано нами - добавляем в Memo1}
   if (to_ = nickname)or(from_ = nickname) then
    Memo1.Lines.Insert(0,from_+' (private) > '+s);
   Exit;
  end;
end;
procedure TForm1.Button1Click(Sender: TObject);
 var s: string;
begin
  {Если мы хотим послать приватное сообщение, но не выбрали адресата -
   выведется замечание}
  if (CheckBox1.Checked)and(ListBox1.ItemIndex < 0) then begin
   ShowMessage('At first you should select the user in the User List!');
   Exit;  end;
  {Если это приватное сообщение}
  if CheckBox1.Checked then
   s := '#P'+ListBox1.Items[ListBox1.ItemIndex]+';' {добавляем спец.команду и адресат}
  else {А если не очень приватное?}
   s := '#M'; {Просто спец.команду}
  {Добавляем наше имя (от кого) и само сообщение}
  s := s+nickname+';'+Edit1.Text;
  {Посылаем все это сообщение по сокету}
  ClientSocket1.Socket.SendText(s);
  {И снова ждем ввода в уже чистом TEdit-е}
  Edit1.Text := '';
  ActiveControl := Edit1; end;
procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  {Если была нажата Enter - тоже посылается сообщение}
  if Key = VK_RETURN then
   Button1.Click;
end;
end.
```

3 Программирование сетевых игр средствами Borland Delphi 6.0

Рассмотрим основы построения сетевых компьютерных игр средствами Delphi 6.0 на примере игры “Шашки”.

“Шашки” – игра белыми и черными кружками на 64-клеточной доске (русские шашки) или 100-клеточной доске (международные шашки) для двух партнеров, у каждого по 12 фишек (русские шашки) или по 20 фишек (международные шашки), также называемых шашками.

Цель игры – уничтожение шашек соперника или создание положения, при котором у него не буде ходов.

3.1 Базовые принципы построения программы

При создании сервера, использующего сокет TCP/IP для реализации сетевого соединения, необходимо выполнить настройку сетевых компонентов.

1) Требуется разместить на форме компонент ServerSocket  со страницы Internet для реализации TCP/IP соединения со стороны сервера.
2) Необходимо выполнить настройку компонента ServerSocket в соответствии с рисунком 54.

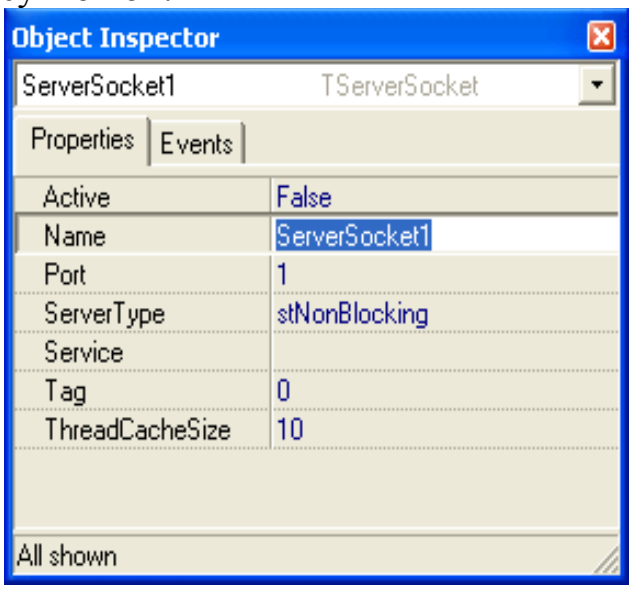


Рисунок 54 – Настройка свойств сервера

3) Требуется создать для компонента ServerSocket1 обработчик события OnClientConnect. Это событие будет инициализировано системой при подключении клиента.

4) Как уже известно, для передачи строки по установленному TCP/IP соединению используется метод SendText объекта типа TCustomWinSocket.

5) Используя объект типа TServerSocket можно реализовать получение данных на стороне сервера. Для этого необходимо создать для компонента ServerSocket1 обработчик события OnClientRead. Это событие будет инициализировано при передаче по установленному соединению данных с клиента.

Необходимо ввести в созданный обработчик события код, принимающий передаваемые данные. Метод ReceiveText принимает данные, переданные объекту Socket.

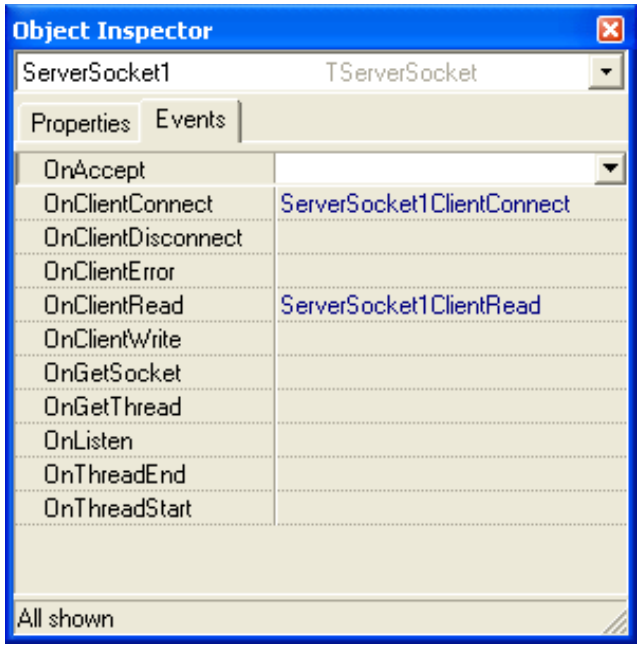


Рисунок 55 – Настройка событий сервера

6) Необходимо разместить на форме некоторый компонент, который будет использоваться в программе для создания сервера, например, компонент типа TButton.

7) Требуется сгенерировать для данного компонента, какое либо событие, например OnClick и в окне кода программы написать следующий код:

```
procedure TForm2.Button2Click(Sender: TObject);
begin
Form1.ServerSocket1.Active:=true;
Form1.SetGame(1);
Form1.Stroi(1);
Form1.Label1.Caption:='Имя: '+edit1.Text;
g.name:=edit1.Text;
form1.Label4.Caption:='Ожидание';
Form1.Label4.Font.Color:=clgreen;
Form1.Cursor:=crhourGlass;
Form1.Image26.Picture.LoadFromFile('Шашка11.bmp');
Close;
end;
```

Создание сервера можно завершено.

Для создания клиента, использующего сокет TCP/IP для реализации сетевого соединения необходимо выполнить следующие действия:

1) На форме расположить компонент ClientSocket со страницы Internet палитры компонентов, который используется для реализации TCP/IP соединения со стороны клиента.

2) Требуется выполнить настройку компонента ClientSocket в соответствии с рисунком 56.

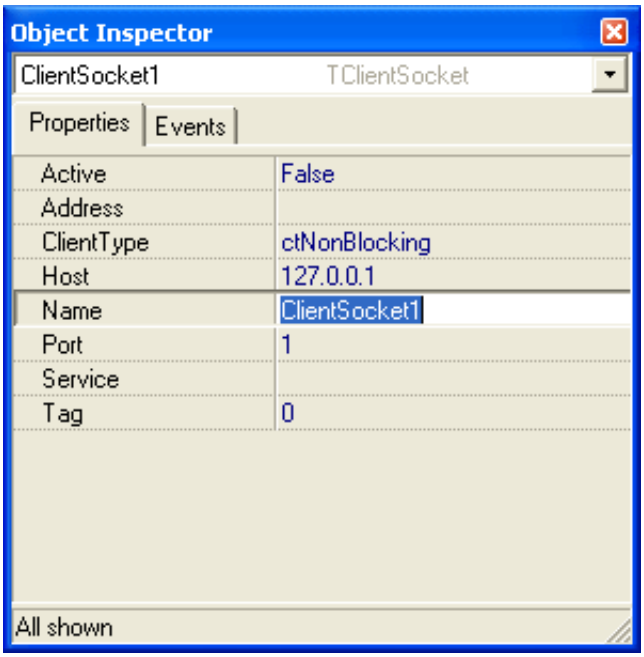


Рисунок 56 – Настройка свойств клиента

3) Необходимо создать для компонента ClientSocket1 обработчик события OnConnect. Это событие будет инициализировано при подключении клиента. Необходимо ввести в созданный обработчик события код, информирующий об установлении соединения. Свойство LocalHost содержит информацию о псевдониме IP-адреса, с которым установлено соединение.

4) Требуется создать код, выполняющий получение данных на стороне клиента. Для этого требуется создать для компонента ClientSocket1 обработчик события OnRead. Метод ReceiveText принимает данные, переданные объекту Socket.

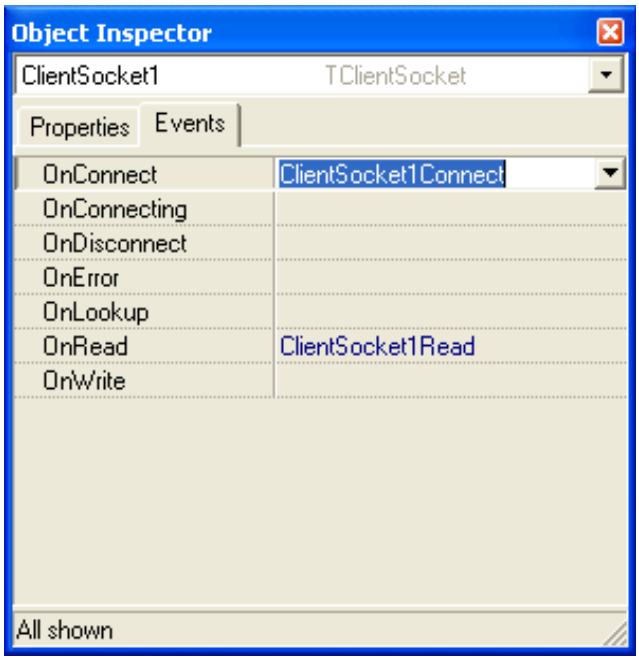


Рисунок 57 – Настройка событий клиента

5) Необходимо разместить на форме некоторый компонент, который будет использоваться в программе для создания клиента, например, компонент типа TButton. Необходимо выполнить для данного компонента событие OnClick и в окне кода программы написать следующий программный код:

```
procedure TForm3.Button2Click(Sender: TObject);
begin
form1.ClientSocket1.Host:=MaskEdit1.text;
form1.ClientSocket1.Active:=true;
form1.SetGame(2);
form1.Stroi(1);
Form1.Label1.Caption:='Имя: '+edit1.Text;
g.name:=edit1.Text;
Form1.Cursor:=crarrow;
Form1.Label4.Caption:='Ваш ход';
Form1.Label4.Font.Color:=clred;
Form1.Image26.Picture.LoadFromFile('Шашка21.bmp');
Form1.mou(sender);
Close;
end;
```

Требуется разместить на форме компоненты функционально необходимые для решения задачи, например компоненты типа:

- TImage – для отображения шашек в окне программы;
- TShape –для отображения имени игрока;
- TButton – для управления ходом программы;
- TMainMenu – компонент главного меню программы.

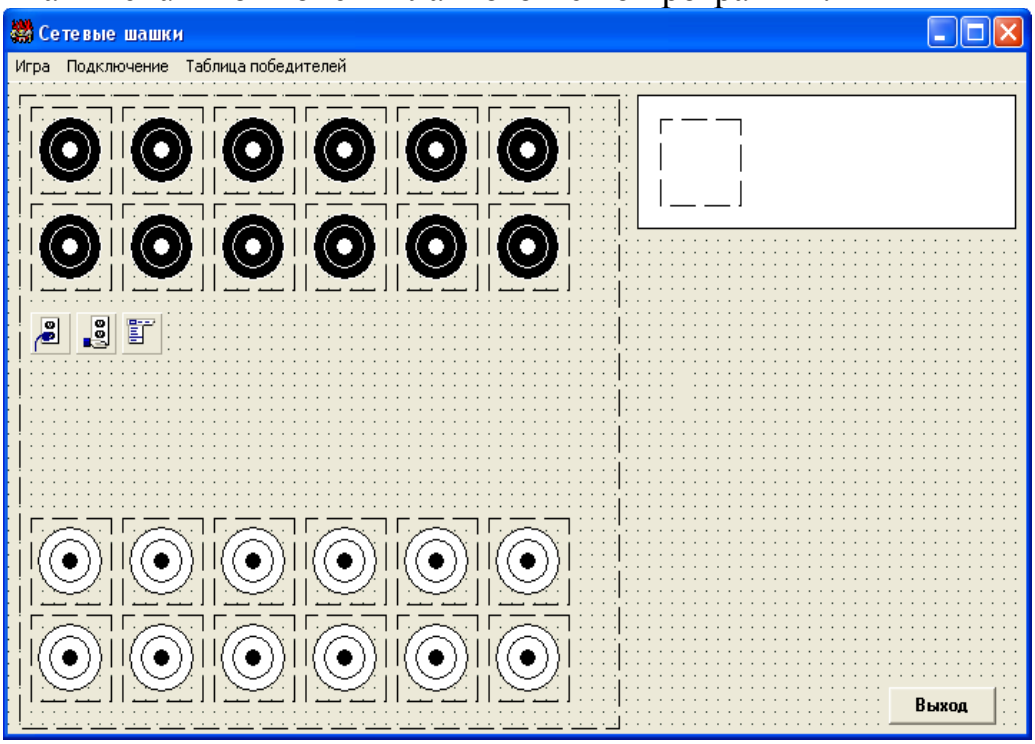


Рисунок 58 – Окно программы в режиме конструктора

Левой клавишей мыши требуется щелкнуть два раза на компоненте MainMenu1. Ввести текст для создания стандартного Windows меню. Пункт меню “Игра” включает в себя пункты: “Начать заново”, “Выход”. Пункт меню Подключение состоит из: “Создать сервер”, “Подключиться к серверу”, “Отсоединение”.

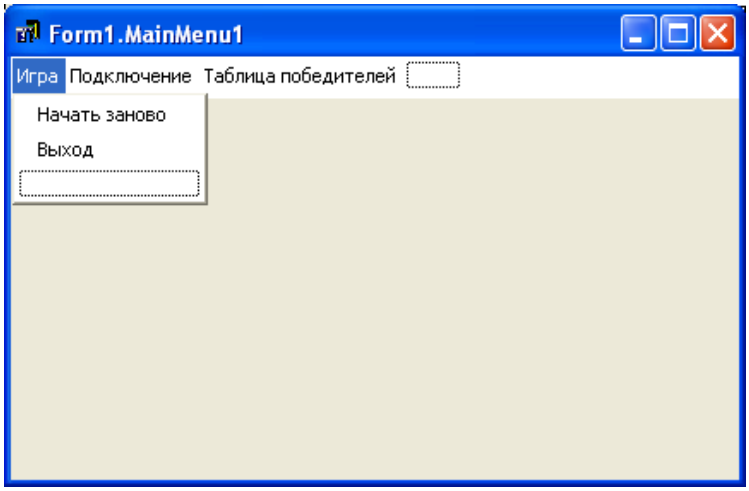


Рисунок 59 – Режим создания меню. Пункт Игра

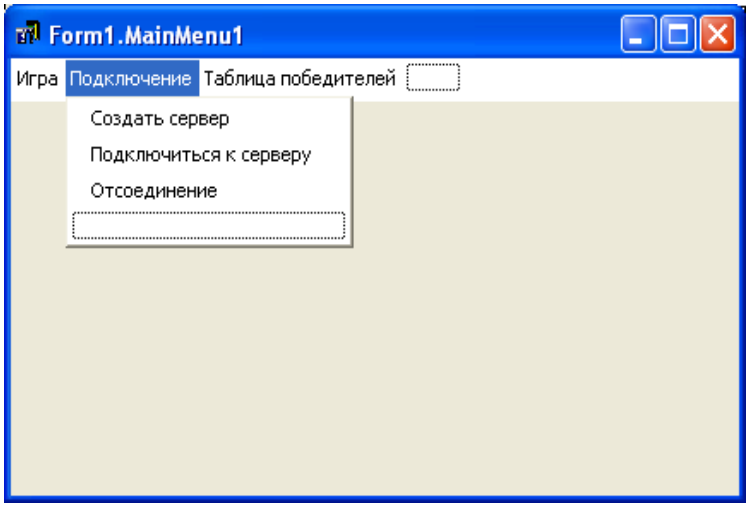


Рисунок 60 – Режим создания меню. Пункт Подключение

В процессе работы клиентское и серверное приложения должны взаимодействовать между собой, передавать данные о совершенных ходах. Процедура обмена данными может быть организована, например, при нажатии на игровом поле указателем мыши и может быть представлена в виде:

```
procedure TForm1.PlaceMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var a,b,s1:integer;
    s:rec;
    F:file of rec;
    op:boolean;
    weri,werj:integer;
    serwer:string;
begin
```

```
if (plc.t) and (g.n=kn[plc.kn])
then begin
    if  (x>=10) and (x<=410)
    and (y>=10) and (y<=410)
    then begin
        a:=((x-10) div 50)+1;
        b:=((y-10) div 50)+1;
        if ((a+b)mod 2)<>0
        then begin
            case err(a,b,s1) of
             0:begin end;
            1:begin
               plc.mp[plc.i,plc.j]:=0;
               msg:=inttostr(plc.i)+inttostr(plc.j);
               plc.i:=a;plc.j:=b;
               msg:=msg+inttostr(a)+inttostr(b);
               plc.mp[a,b]:=plc.k;
               mas[plc.k].Setij(a,b);
               if ((g.ud=1) and (b=1))
               or ((g.ud=2) and (b=8))
               then mas[plc.k].SetT(2);
               plc.t:=false;
               plc.k:=0;
               revers;
               msg:=msg+inttostr(plc.kn);
               if ServerSocket1.Active
               then ServerSocket1.Socket.Connections[0].SendText(msg)
               else ClientSocket1.Socket.SendText(msg);
               if Victoria then begin
                 messagedlg('Вы победили',mtinformation,[mbyes],0);
                 Form4.ShowModal;
                 op:=false;
                 Form4.FormShow(Sender);
                 AssignFile(f,'Victor.txt');
                 rewrite(f);
                 For werj:=1 to 2 do begin
                   if serwer=name2 then serwer:=g.name else serwer:=name2;
                   for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                     if Form4.StringGrid1.Cells[0,weri]=serwer then begin
      Form4.StringGrid1.Cells[werj,weri]:=inttostr(strtoint(Form4.StringGrid1.Cells[werj,weri])
+1);
                        op:=true;
                   end;
                 end;
                 if op=false then begin
```

```
            Form4.StringGrid1.RowCount:=Form4.StringGrid1.RowCount+1;
            Form4.StringGrid1.Cells[0,Form4.StringGrid1.RowCount-2]:=serwer;
            Form4.StringGrid1.Cells[1,Form4.StringGrid1.RowCount-2]:=inttostr(1);
            Form4.StringGrid1.Cells[2,Form4.StringGrid1.RowCount-2]:=inttostr(0);
          end;
        end;
        for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
          s.s1:=Form4.StringGrid1.Cells[0,weri];
          s.n1:=strtoint(Form4.StringGrid1.Cells[1,weri]);
          s.n2:=strtoint(Form4.StringGrid1.Cells[2,weri]);
          write(f,s);
        end;
        msg:='y';
        if ServerSocket1.Active
        then ServerSocket1.Socket.Connections[0].SendText(msg)
        else ClientSocket1.Socket.SendText(msg);
        Form4.ShowModal;
      end;
    end;
  2:begin
      plc.s:=true;
      plc.mp[plc.i,plc.j]:=0;
      msg:=inttostr(plc.i)+inttostr(plc.j);
      plc.i:=a;plc.j:=b;
      msg:=msg+inttostr(a)+inttostr(b);
      plc.mp[a,b]:=plc.k;
      mas[plc.k].Setij(a,b);
      if ((g.ud=1) and (b=1))
      or ((g.ud=2) and (b=8))
      then mas[plc.k].SetT(2);
      plc.t:=YesStep(plc.i,plc.j);
      if not(plc.t) then begin plc.k:=0;revers; end;
      msg:=msg+inttostr(plc.kn);
      if s1>9 then msg:=msg+inttostr(s1)
            else msg:=msg+'0'+inttostr(s1);
      if ServerSocket1.Active
      then ServerSocket1.Socket.Connections[0].SendText(msg)
      else ClientSocket1.Socket.SendText(msg);
      if Victoria then begin
        messagedlg('Вы победили',mtinformation,[mbyes],0);
        op:=false;
        Form4.FormShow(Sender);
        AssignFile(f,'Victor.txt');
        rewrite(f);
        For werj:=1 to 2 do begin
```

```
                if serwer=name2 then serwer:=g.name else serwer:=name2;
                for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                  if Form4.StringGrid1.Cells[0,weri]=serwer then begin
    Form4.StringGrid1.Cells[werj,weri]:=inttostr(strtoint(Form4.StringGrid1.Cells[werj,weri])+
1);
                    op:=true;
                  end;
                end;
                if op=false then begin
                  Form4.StringGrid1.RowCount:=Form4.StringGrid1.RowCount+1;
                  Form4.StringGrid1.Cells[0,Form4.StringGrid1.RowCount-2]:=serwer;
                  Form4.StringGrid1.Cells[1,Form4.StringGrid1.RowCount-2]:=inttostr(1);
                  Form4.StringGrid1.Cells[2,Form4.StringGrid1.RowCount-2]:=inttostr(0);
                end;
              end;
              for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                s.s1:=Form4.StringGrid1.Cells[0,weri];
                s.n1:=strtoint(Form4.StringGrid1.Cells[1,weri]);
                s.n2:=strtoint(Form4.StringGrid1.Cells[2,weri]);
                write(f,s);
              end;
              msg:='y';
              if ServerSocket1.Active
              then ServerSocket1.Socket.Connections[0].SendText(msg)
              else ClientSocket1.Socket.SendText(msg);
              Form4.ShowModal;
            end;
          end;
        end;
      end;
    end;
end; end;
```

В завершение необходимо сохранить проект и можно запускать программу. В Delphi необходимо выполнить команду “File” => “Save As…”, в результате станет активным окно “Save As”. Рекомендуется сохранить проект в отдельной папке, к примеру, в папке с именем Шашки.

3.2 Руководство по эксплуатации программного средства

Программу можно запустить на выполнение из среды Delphi для этого необходимо нажать клавишу F9. Однако, возможен другой способ: в проводнике Windows необходимо, при помощи курсора мыши перейти к той папке, в которой сохранен проект, затем необходимо отыскать .exe файл с указанным ранее именем, щелкнуть на данном файле левой кнопкой мыши два раза.

Выполнить выше описанные действия еще раз. Причем, важно понимать что, выполнять это приложение можно как на одном, так и на разных компьютерах локальной вычислительной сети.

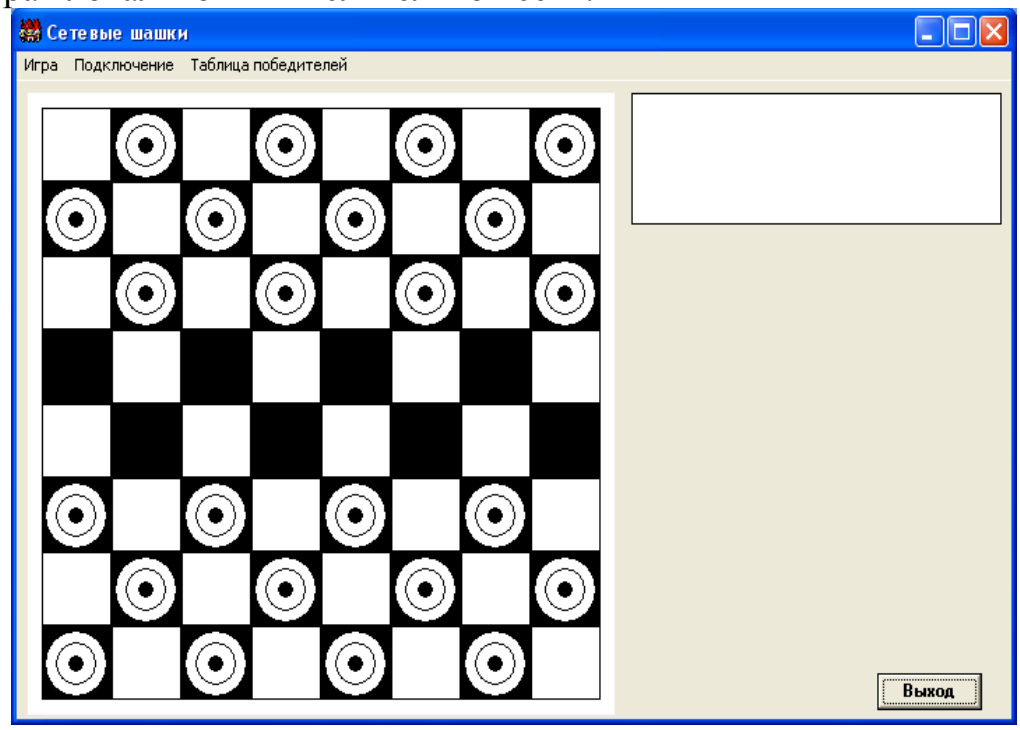


Рисунок 61 – Окно программы после запуска

Для начала, необходимо выполнить подключение со стороны сервера. Для этого требуется выбрать пункт меню “Подключение”, после чего необходимо выбрать пункт “Создать сервер”.

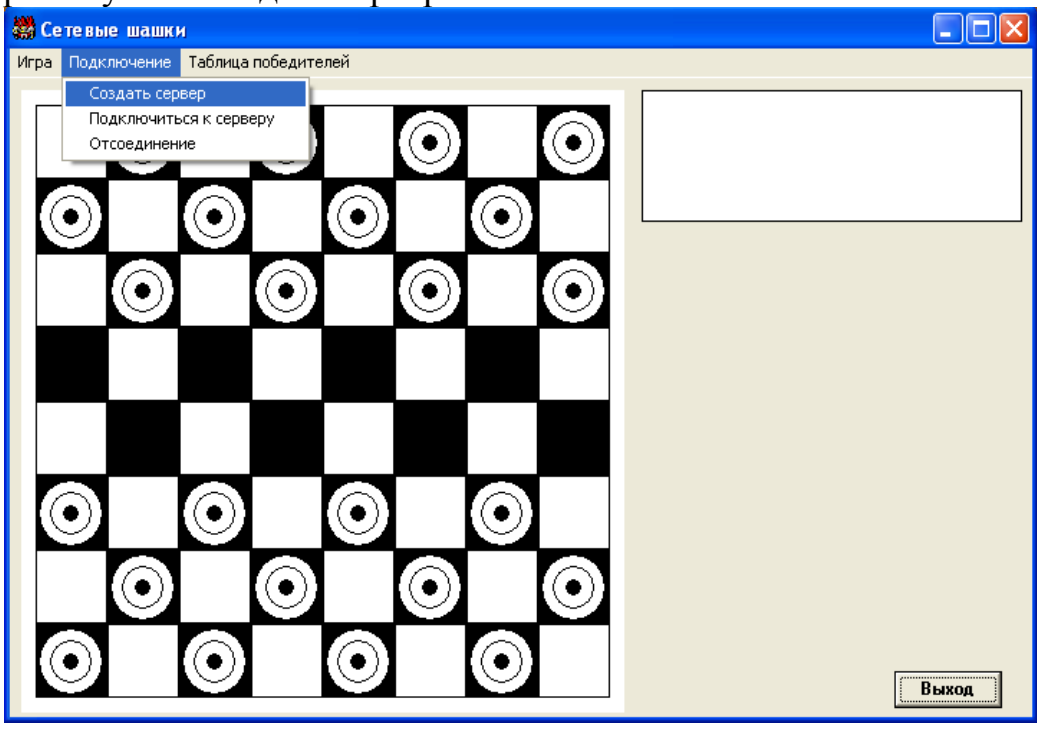


Рисунок 62 – Пункт меню “Создать сервер”

В результате выполнения меню “Подключение” и подменю “Создать сервер” станет активным окно “Создание сервера”.

В появившемся окне необходимо ввести имя игрока. Приложение, инициированное с указанным именем, будет работать в режиме Сервер. Данная ситуация представлена на рисунке 63.

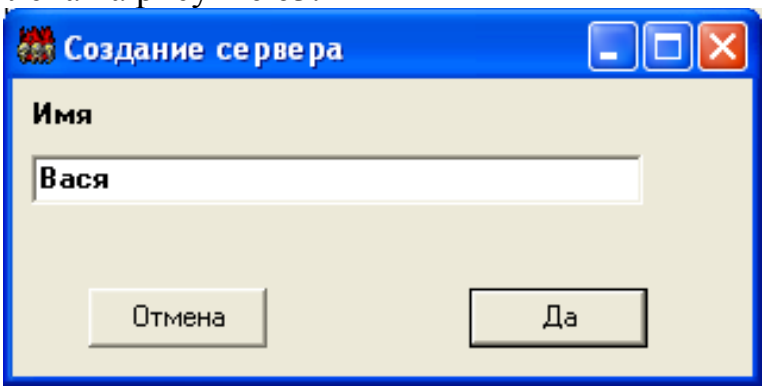


Рисунок 63 – Регистрация со стороны сервера

В результате проведенных действий вид окна рабочей программы должен измениться в соответствии с рисунком 64.

По рисунку 64 видно, что сервер находится в режиме ожидание, а имя игрока – Вася, однако, отсутствует информация о противнике, который пока не подключился к серверу.

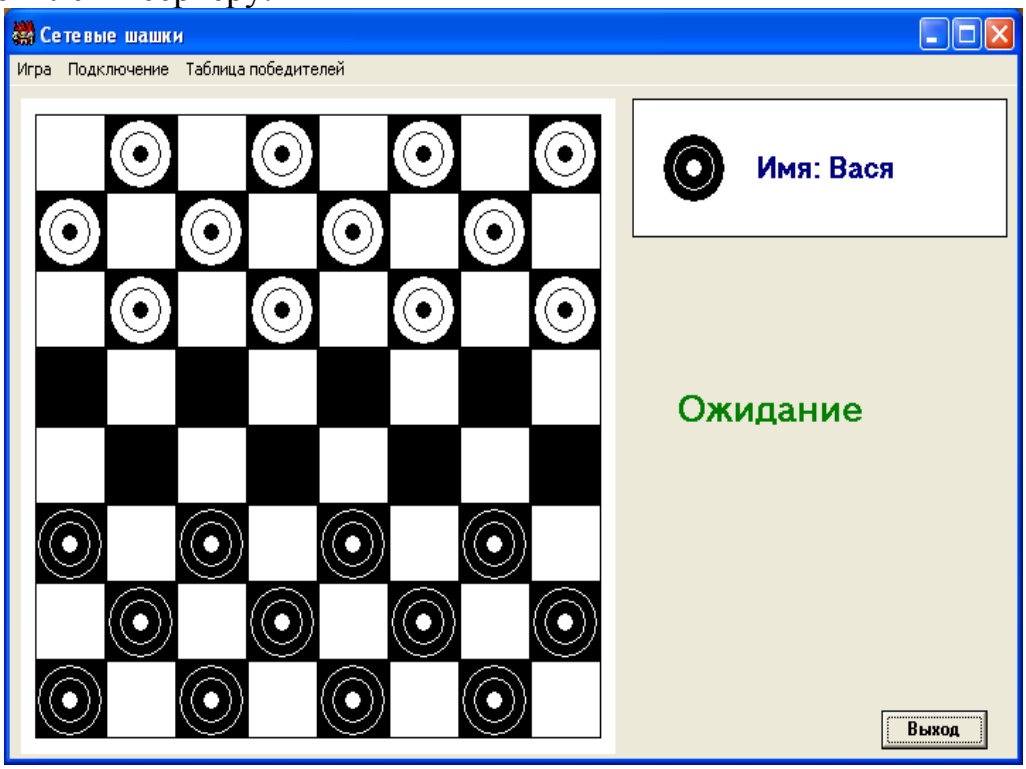


Рисунок 64 – Окно приложения-сервера

Для создания клиента, использующего сокет TCP/IP для реализации сетевого соединения на уровне приложения необходимо выполнить следующую последовательность действий:

1) В окне программы требуется навести указатель мыши на пункт меню “Подключение”.

2) Необходимо щелкнуть на нем левой кнопкой мыши.

3) В раскрывшемся подменю второго уровня нужно найти пункт “Подключиться к серверу”.

4) Необходимо щелкнуть левой кнопкой мыши на пункте меню “Подключиться к серверу”.

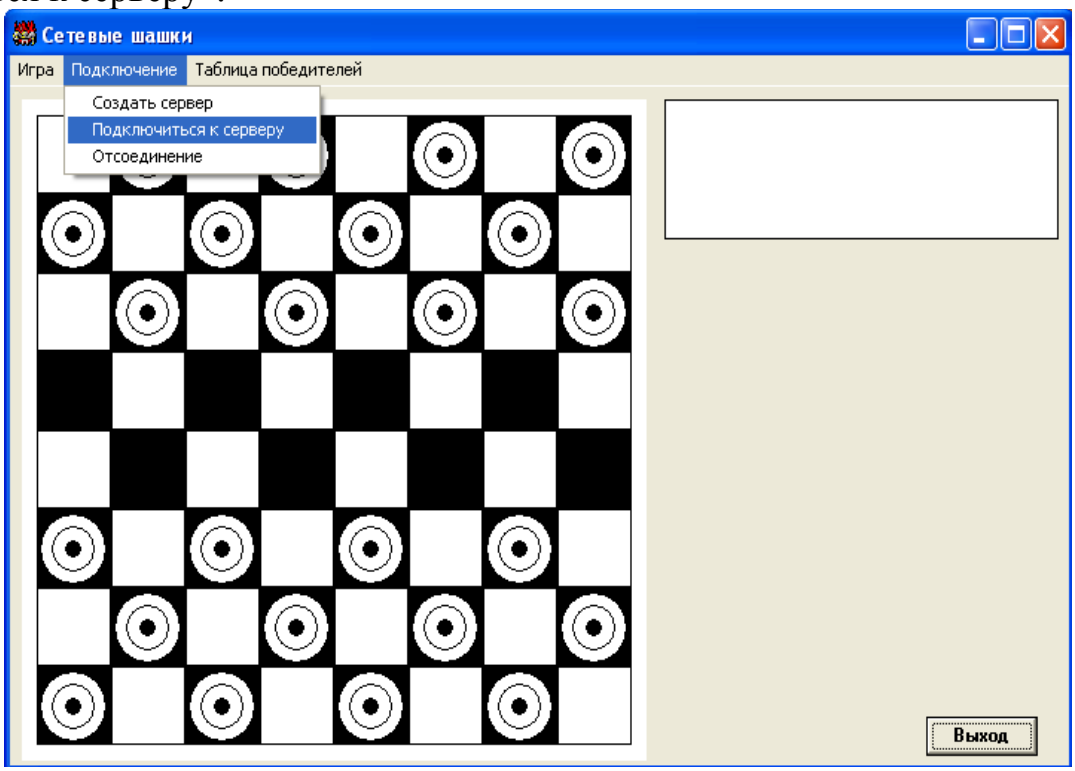


Рисунок 65 – Подключение к серверу

В результате выше описанных действий по созданию клиента активизируется окно “Подключение к серверу”, в котором необходимо ввести имя игрока и указать IP-адрес сервера, к которому производится подключение.

Приложение, инициированное с указанным именем, будет работать в режиме – Клиент. Ситуация представлена на рисунке 66.

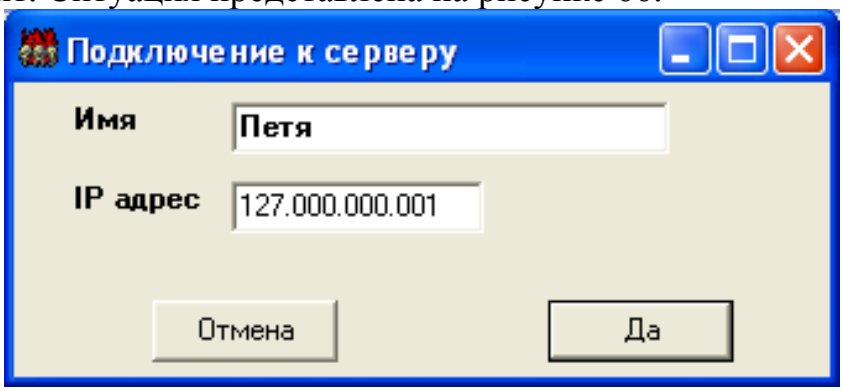


Рисунок 66 – Регистрация со стороны клиента

Таким образом, было выполнено инициирование приложения Сервера – игрок под именем Вася и приложение Клиент – игрок под именем Петя.

В результате проведенных действий вид окна рабочих программ приложения Сервера и приложения Клиента должен измениться в соответствии с рисунками 67-68.

На рисунках видно:

- приложение Сервер находится в режиме ожидание, имя игрока – Вася, соперник – Петя;

- приложение Клиент находится в активном режиме, имя игрока – Петя, соперник – Вася.

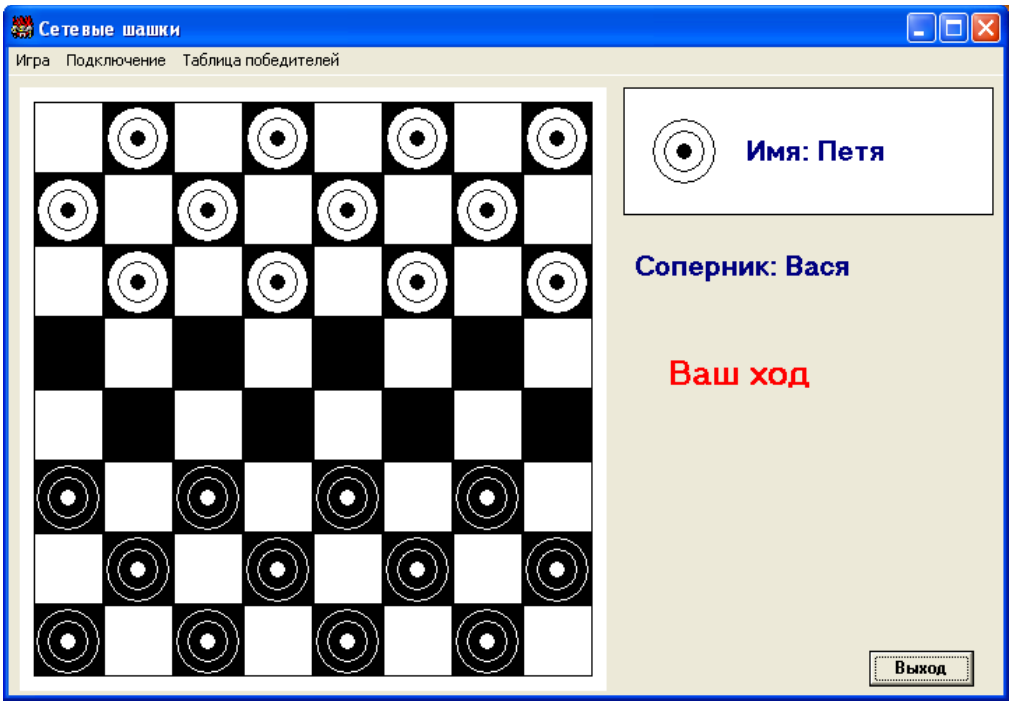


Рисунок 67 – Окно клиента

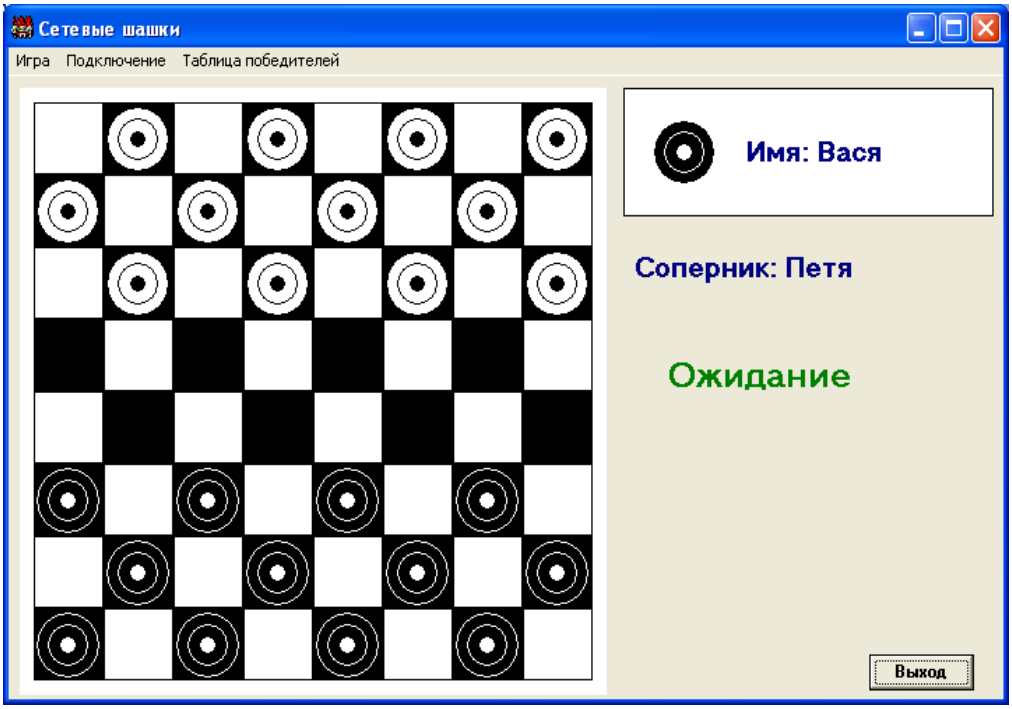


Рисунок 68 – Окно сервера

После выполнения всех мероприятий по настройки подключения со стороны сервера и со стороны клиента можно приступать к игре.

Право на первый ход имеет игрок, играющий белыми шашками, в нашем случае это Петя.

Для выполнения хода необходимо:

- выбрать необходимую шашку, нажав на нее левой кнопкой мыши;
- выбрать черную клетку игровой области;
- щелкнуть левой кнопкой мыши по выбранной клетке, таким образом, шашка будет перемещена на новую клетку.

Рисунок 69 – Начало игры

Для того чтобы срубить шашку противника необходимо:
- выбрать необходимую шашку, нажав на ней левой кнопкой мыши;
- щелкнуть левой кнопкой мыши на клетке, следующей за шашкой противника, эта клетка должна быть свободной;
- в результате проведенных действий противник лишится шашки, а может быть и не одной.

На рисунках 70-71 приведена ситуация появления дамки в игре. На первом контрольном примере изображена ситуация появление дамки у “черных”, на втором – у “белых”.

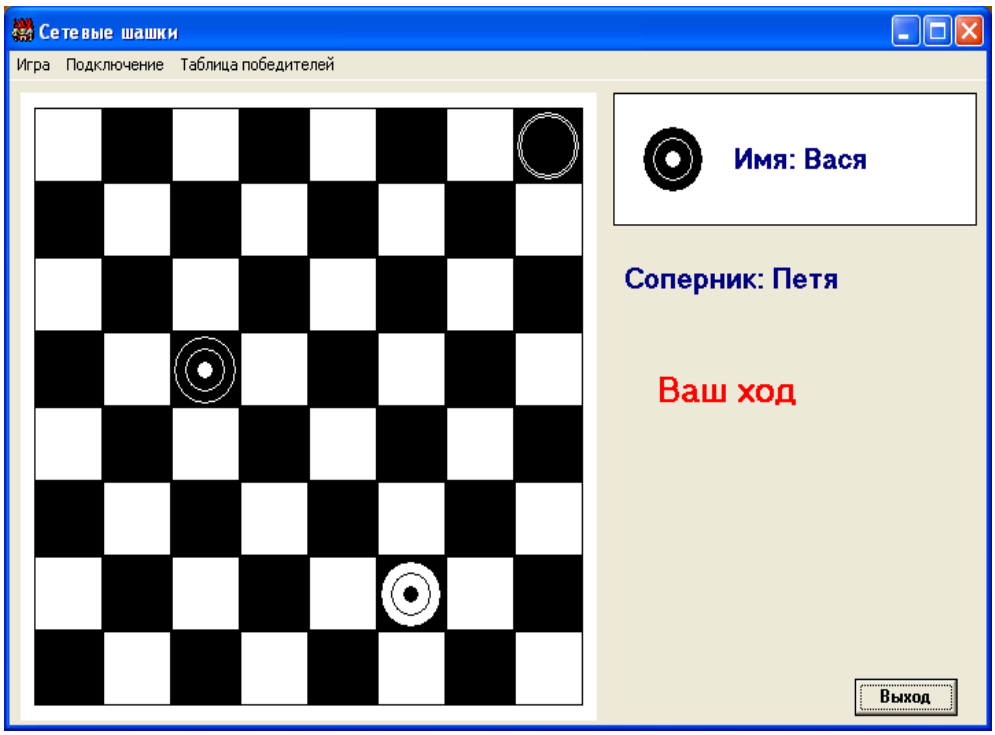


Рисунок 70 – Появление дамки

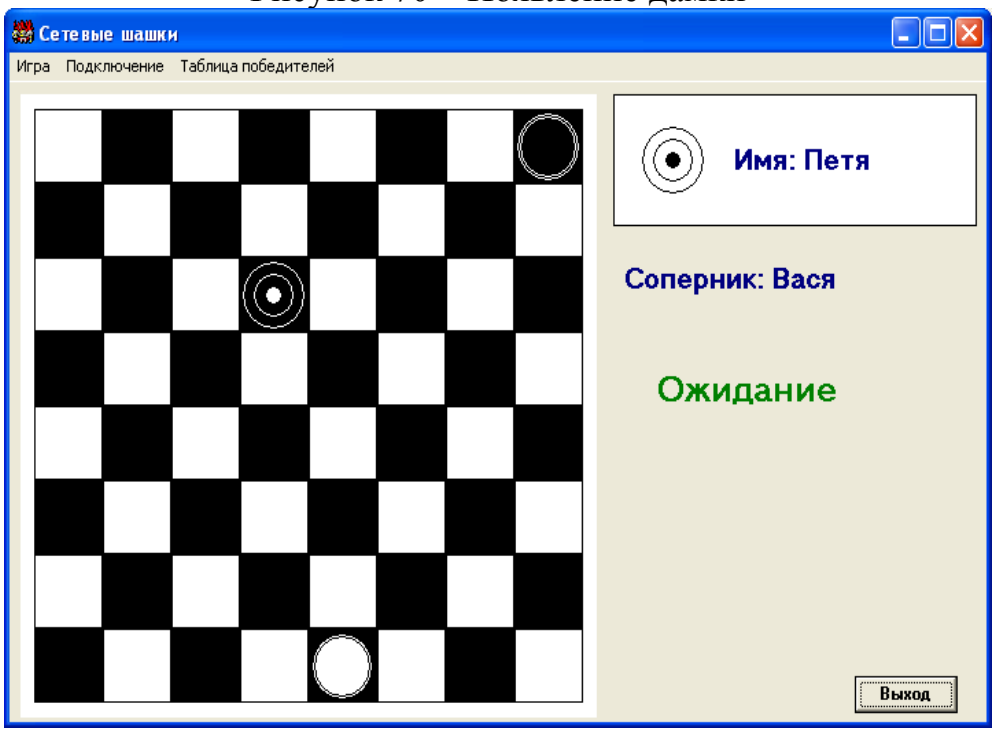


Рисунок 71 – Появление дамки

В данном тестовом примере игрок, зарегистрировавшийся под именем Вася, партию выиграл. По завершении игровой партии на экран монитора выдается сообщение о результатах игры. В появившемся информационном окне содержится информация: “Вы победили!” (рисунок 72).

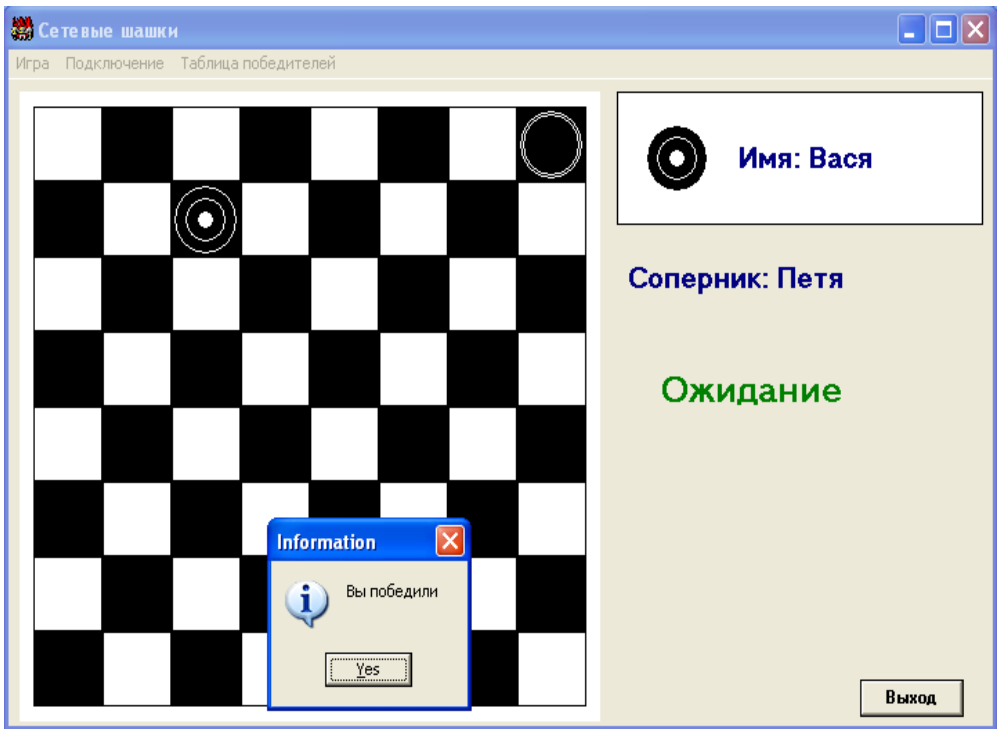


Рисунок 72 – Сообщение о победе

Игрок, зарегистрировавшийся под именем Петя, данную игру проиграл. По завершении игровой партии на экран выдается сообщение о результатах игры. Таким образом, в появившемся информационном окне содержится информация: “Вы проиграли!” (рисунок 73).

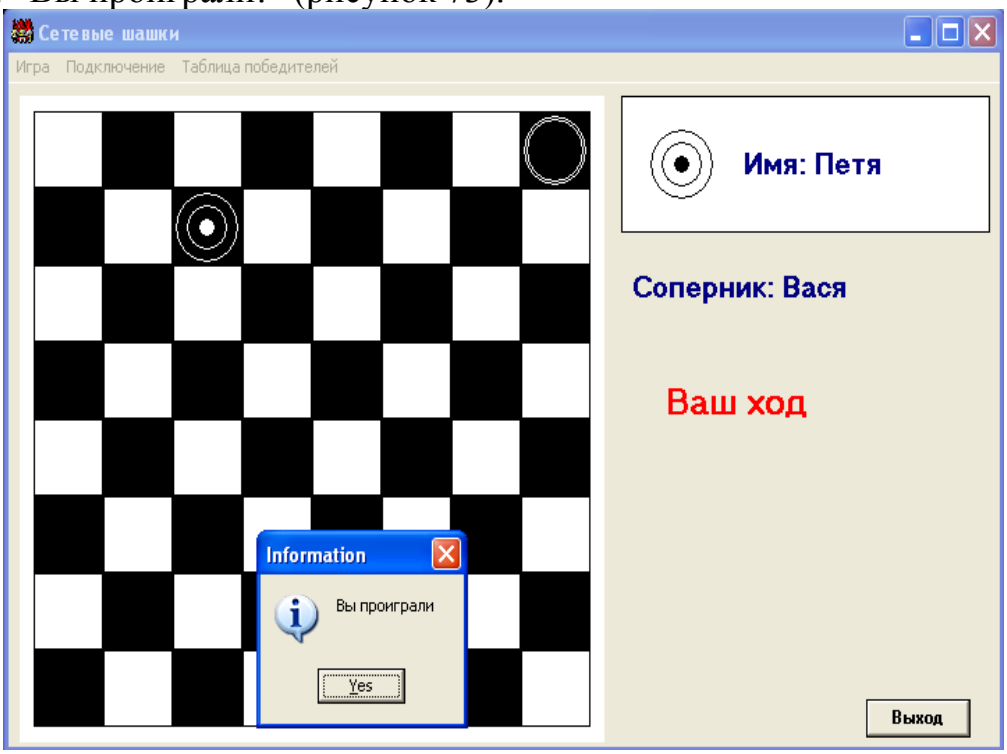


Рисунок 73 – Сообщение о проигрыше

В программе после завершения игры и после выдачи сообщения об ее итоге выдается таблица рекордов, в которой представлена информация по игрокам, а именно: зарегистрированное имя игрока, количество побед и

проигрышей каждого игравшего. Для того чтобы закрыть Таблицу рекордов необходимо щелкнуть левой кнопкой мыши на кнопке “Закрыть”.

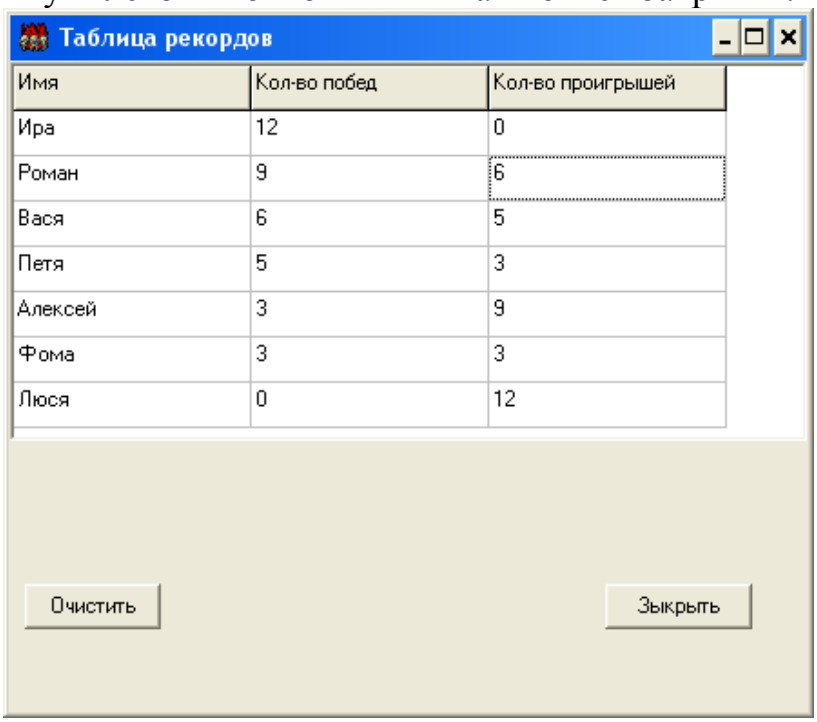


Рисунок 74 – Таблица рекордов

В том случае, если есть необходимость играть следующую партию, требуется выбрать пункт меню “Начать заново” из меню “Игра”. Данная ситуация наглядно представлена на рисунке 75.

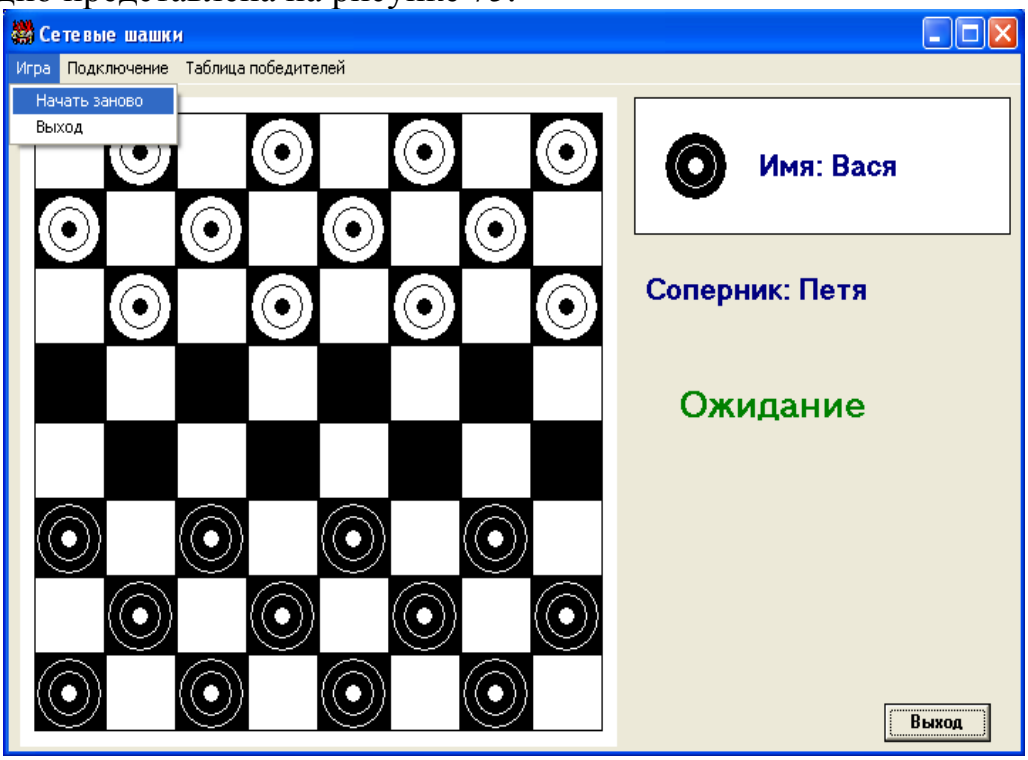


Рисунок 75 – Начать заново

Для выхода из программы необходимо нажать кнопку “Закрыть” в правом верхнем углу строки заголовка или можно нажать на кнопке “Выход” в правом нижнем углу рабочей формы.

3.3 Программный код

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ScktComp, Menus, ImgList;
type
  TForm1 = class(TForm)
    Button1: TButton;
    ClientSocket1: TClientSocket;
    ServerSocket1: TServerSocket;
    Image3: TImage;
    Image4: TImage;
    Image5: TImage;
    Image6: TImage;
    Image7: TImage;
    Image8: TImage;
    Image9: TImage;
    Image10: TImage;
    Image11: TImage;
    Image12: TImage;
    Image1: TImage;
    Image2: TImage;
    Place: TImage;
    Image13: TImage;
    Image14: TImage;
    Image15: TImage;
    Image16: TImage;
    Image17: TImage;
    Image18: TImage;
    Image19: TImage;
    Image20: TImage;
    Image21: TImage;
    Image22: TImage;
    Image23: TImage;
    Image24: TImage;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
```

```
    N8: TMenuItem;
    Label1: TLabel;
    Label2: TLabel;
    N9: TMenuItem;
    Label4: TLabel;
    Image26: TImage;
    Shape1: TShape;
    function  YesStep(i,j:integer):boolean;
    procedure CreatPlace;
    procedure Button1Click(Sender: TObject);
    procedure Fust;
    procedure Stroi(s:byte);
    procedure PlaceMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ClickImage(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
    function Err(i,j:integer;var s:integer):byte;
    procedure SetGame(f:byte);
    procedure ClientSocket1Read(Sender: TObject; Socket: TCustomWinSocket);
    procedure ServerSocket1ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure N6Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure N8Click(Sender: TObject);
    function dga(i,j,a,b,t:integer;var k:integer):byte;
    procedure ServerSocket1ClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure Read(Socket:TCustomWinSocket);
    procedure ClientSocket1Connect(Sender: TObject;
      Socket: TCustomWinSocket);
    function stepy(k:integer):boolean;
    function Victoria:boolean;
    procedure N9Click(Sender: TObject);
    procedure mou(Sender: TObject);
    procedure N4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
type rec=record
     s1:string[15];
     n1,n2:integer;
```

```
        end;
    type ghj=array [1..8,1..8] of integer;
    type gamer=record                  //Тип - игрок
        n:integer;                     //Идентификатор игрока
        wb:byte;                       //Цвет за к-ый он играет
        ud:byte;                       //Направление перемещения шашка на
экране(вверх, вниз)
        name:string;
        end;
    var
      Form1: TForm1;
      gk:ghj;
      name2:string;
      g:gamer;
      k:integer;
    implementation
    uses Unit2, Unit3, Unit4;
    {$R *.dfm}
    type obj=object                    //Объект - шашка
        Im:Timage;                     //Поле типа картинка
        wb:byte;                       //Цвет шашки
        tp:byte;                       //Тип шашки (простая или дамка)
        i,j:integer;                   //Положение шашки на шахматном столе
        procedure Setij(a,b:integer); //Задать положение шашки
        procedure SetT(a:byte);       //Задает тип шашки (дамка или простая)
        procedure Put(a,b:integer);    //Переставляет шашку на нужное положение
        end;
    procedure obj.Setij(a,b:integer);
    begin
    i:=a;j:=b;
    Put(i,j); end;
    procedure obj.SetT(a:byte);
    begin
    tp:=a;
    case a of
    1:if wb=1 then im.Picture.LoadFromFile('Шашка11.bmp')
            else im.Picture.LoadFromFile('Шашка21.bmp');

    2:if wb=1 then im.Picture.LoadFromFile('Шашка12.bmp')
            else im.Picture.LoadFromFile('Шашка22.bmp');
    end; end;
    procedure obj.Put(a,b:integer);
    var x,y:integer;
    begin
    x:=15+(a-1)*50;
```

```
    y:=15+(b-1)*50;
    Im.Left:=x;
    Im.Top:=y; end;
    type PlaceT=record                  //Тип - положение игры сейчас
        i,j,k,wb:integer;               //Соотв положение выбр. шашки, номер шашки в ма-
сиве и цвет
        t,s:boolean;                    //Определяет выбрана ли шашка
        kn:byte;
        mp:ghj;                         //Массив состояний поля игры
        end;
    var mas:array [1..24] of obj;       //Массив шашек
       plc:PlaceT;                      //Положение игры
       kn:array [1..2] of integer;
       msg:string;
    procedure TForm1.CreatPlace;       //Рисование шахматной доски
    var i,j:integer;
    begin
    Place.Picture.Bitmap.Width:=420;
    Place.Picture.Bitmap.Height:=420;
    for i:=1 to 8 do
    for j:=1 to 8 do
    begin
    if ((j+i)mod 2)=0
    then Place.Canvas.Brush.Color:=clwhite
    else Place.Canvas.Brush.Color:=clblack;
    Place.Canvas.Rectangle(10+(i-1)*50,10+(j-1)*50,10+i*50,10+j*50);
    end; end;
    procedure TForm1.Button1Click(Sender: TObject);
    begin
    Close; end;
    procedure TForm1.Fust;                  //Занесение картинок-шашек в массив
    begin
    mas[1].Im:=Image1;
    mas[2].Im:=Image2;
    mas[3].Im:=Image3;
    mas[4].Im:=Image4;
    mas[5].Im:=Image5;
    mas[6].Im:=Image6;
    mas[7].Im:=Image7;
    mas[8].Im:=Image8;
    mas[9].Im:=Image9;
    mas[10].Im:=Image10;
    mas[11].Im:=Image11;
    mas[12].Im:=Image12;
    mas[13].Im:=Image13;
```

```
mas[14].Im:=Image14;
mas[15].Im:=Image15;
mas[16].Im:=Image16;
mas[17].Im:=Image17;
mas[18].Im:=Image18;
mas[19].Im:=Image19;
mas[20].Im:=Image20;
mas[21].Im:=Image21;
mas[22].Im:=Image22;
mas[23].Im:=Image23;
mas[24].Im:=Image24;
end;
procedure TForm1.Stroi(s:byte);                    //Расположение шашек в массиве
var i,j,k:integer;
begin
Fust;
for i:=1 to 8 do
for j:=1 to 8 do
plc.mp[i,j]:=0;
if s=0 then k:=0 else k:=12;
for i:=1 to 8 do
for j:=1 to 3 do
begin
if ((j+i)mod 2)<>0
then begin
    k:=k+1;
    if g.ud=2
    then mas[k].wb:=g.wb
    else mas[k].wb:=2-g.wb+1;
    mas[k].Setij(i,j);
    mas[k].SetT(1);
    mas[k].Im.Visible:=true;
    plc.mp[i,j]:=k;
    end; end;
if s=0 then k:=12 else k:=0;
for i:=1 to 8 do
for j:=6 to 8 do
begin
if ((j+i)mod 2)<>0
then begin
    k:=k+1;
    if (g.ud=1)
    then mas[k].wb:=g.wb
    else mas[k].wb:=2-g.wb+1;
    mas[k].Setij(i,j);
```

```
    mas[k].SetT(1);
    mas[k].Im.Visible:=true;
    plc.mp[i,j]:=k;
    end; end; end;
function tip(k:integer):byte;                    //Определение типа шашки
begin
if k<=12 then tip:=1
        else tip:=2;
end;
function TForm1.dga(i,j,a,b,t:integer;var k:integer):byte;
var k1,k2,n,f,g:integer;
begin
dga:=0;
k1:=0;k2:=0;
if i>a then k1:=-1 else k1:=1;
if j>b then k2:=-1 else k2:=1;
i:=i+k1;j:=j+k2;n:=0;
while (i<>a) and (j<>b) do
begin
if (plc.mp[i,j]<>0)
then begin
    if (n<1) and (mas[plc.mp[i,j]].wb<>t)
    then begin n:=n+1;k:=plc.mp[i,j];f:=i;g:=j; end
    else begin dga:=0;exit;
end;
    end;
i:=i+k1;
j:=j+k2;
end;
if n=1
then begin
    mas[plc.mp[f,g]].Im.Visible:=false;
    plc.mp[f,g]:=0;
    end;
dga:=n+1;
end;
function TForm1.Err(i,j:integer;var s:integer):byte;  //Проверка хода
var a,b:integer;
begin
if plc.mp[i,j]<>0
then begin
    messagedlg('Клетка занята',mterror,[mbyes],0);
    err:=0;
    end
else if mas[plc.k].tp=1                          //Для простых шашек
```

```
    then begin
        if  (abs(i-plc.i)=1)
        and (abs(j-plc.j)=1)
        then case g.ud of
            1:if plc.j>j then err:=1 else err:=0;
            2:if plc.j<j then err:=1 else err:=0;
            end
        else
 begin
            a:=(i+plc.i)div 2;
            b:=(j+plc.j)div 2;
            if  (abs(i-plc.i)=2) and (abs(j-plc.j)=2)
            and (plc.mp[a,b]<>0) and (tip(plc.mp[a,b])<>g.wb)
            then begin
                mas[plc.mp[a,b]].Im.Visible:=false;
                s:=plc.mp[a,b];
                plc.mp[a,b]:=0;
                err:=2;    //рубит
                end
            else err:=0;
            end;          end
    else begin                                          //Для дамок
        if  (abs(plc.i-i)=abs(plc.j-j))
        then begin
            err:=dga(plc.i,plc.j,i,j,mas[plc.k].wb,s);
            end
        else err:=0;
        end;
end;
function TForm1.YesStep(i,j:integer):boolean;
begin
yesstep:=false;
{правая верхняя}
if  ((i+1)<=8) and ((j+1)<=8)
and (plc.mp[i+1,j+1]<>0)
and (mas[plc.mp[i+1,j+1]].wb<>g.wb)
then begin
    if  ((i+1)+1<=8) and ((j+1)+1<=8)
    and (plc.mp[(i+1)+1,(j+1)+1]=0)
    then yesstep:=true;
    end;
{левая верхняя}
if  ((i-1)>=1) and ((j+1)<=8)
and (plc.mp[i-1,j+1]<>0)
and (mas[plc.mp[i-1,j+1]].wb<>g.wb)
```

```
then begin
    if  ((i-1)-1>=1) and ((j+1)+1<=8)
    and (plc.mp[(i-1)-1,(j+1)+1]=0)
    then yesstep:=true;
    end;
{правая нижняя}
if  ((i+1)<=8) and ((j-1)>=1)
and (plc.mp[i+1,j-1]<>0)
and (mas[plc.mp[i+1,j-1]].wb<>g.wb)
then begin
    if  ((i+1)+1<=8) and ((j-1)-1>=1)
    and (plc.mp[(i+1)+1,(j-1)-1]=0)
    then yesstep:=true;
    end;
{левая нижняя}
if  ((i-1)>=1) and ((j-1)>=1)
and (plc.mp[i-1,j-1]<>0)
and (mas[plc.mp[i-1,j-1]].wb<>g.wb)
then begin
    if  ((i-1)-1>=1) and ((j-1)-1>=1)
    and (plc.mp[(i-1)-1,(j-1)-1]=0)
    then yesstep:=true;
    end;
 end;
procedure TForm1.mou(Sender: TObject);
var n:integer;
begin
for k:=1 to 24 do mas[k].Im.Cursor:=crNo;
if g.ud=1 then n:=12 else n:=24;
for k:=n-11 to n do mas[k].Im.Cursor:=crHandPoint;
end;
procedure revers;
var n:Tcursor;
begin
if plc.kn=1 then begin
form1.Label4.Caption:='Ожидание';
Form1.Label4.Font.Color:=clgreen;
Form1.Cursor:=crhourGlass;
for k:=1 to 24 do mas[k].Im.Cursor:=crNo;
if g.ud=1 then n:=12 else n:=24;
for k:=n-11 to n do mas[k].Im.Cursor:=crHandPoint;
mas[k].Im.Cursor:=n;
plc.kn:=2; end
else begin plc.kn:=1;
Form1.Label4.Caption:='Ожидание';
```

```
    Form1.Label4.Font.Color:=clgreen;
    Form1.Cursor:=crhourGlass;
    for k:=1 to 24 do mas[k].Im.Cursor:=crNo;
    if g.ud=1 then n:=12 else n:=24;
    for k:=n-11 to n do mas[k].Im.Cursor:=crHandPoint;
    end;
    end;
    procedure TForm1.PlaceMouseDown(Sender: TObject; Button: TMouseButton; //Собы-
тия при нажатии на доску
      Shift: TShiftState; X, Y: Integer);
    var a,b,s1:integer;
       s:rec;
       F:file of rec;
       op:boolean;
       weri,werj:integer;
       serwer:string;
    begin
    if (plc.t) and (g.n=kn[plc.kn])
    then begin
        if  (x>=10) and (x<=410)
        and (y>=10) and (y<=410)
        then begin
             a:=((x-10) div 50)+1;
             b:=((y-10) div 50)+1;
             if ((a+b)mod 2)<>0
             then begin
                 case err(a,b,s1) of
                 0:begin end;
                 1:begin
                    plc.mp[plc.i,plc.j]:=0;
                    msg:=inttostr(plc.i)+inttostr(plc.j);
                    plc.i:=a;plc.j:=b;
                    msg:=msg+inttostr(a)+inttostr(b);
                    plc.mp[a,b]:=plc.k;
                    mas[plc.k].Setij(a,b);
                    if ((g.ud=1) and (b=1))
                    or ((g.ud=2) and (b=8))
                    then mas[plc.k].SetT(2);
                    plc.t:=false;
                    plc.k:=0;
                    revers;
                    msg:=msg+inttostr(plc.kn);
                    if ServerSocket1.Active
                    then ServerSocket1.Socket.Connections[0].SendText(msg)
                    else ClientSocket1.Socket.SendText(msg);
```

```
                    if Victoria then begin
                      messagedlg('Вы победили',mtinformation,[mbyes],0);
                      Form4.ShowModal;
                      op:=false;
                      Form4.FormShow(Sender);
                      AssignFile(f,'Victor.txt');
                      rewrite(f);
                      For werj:=1 to 2 do begin
                        if serwer=name2 then serwer:=g.name else serwer:=name2;
                        for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                          if Form4.StringGrid1.Cells[0,weri]=serwer then begin
                         Form4.StringGrid1.Cells[werj,weri]:=inttostr(strtoint(Form4.StringGrid1.
Cells[werj,weri])+1);
                              op:=true;
                        end; end;
                        if op=false then begin
                          Form4.StringGrid1.RowCount:=Form4.StringGrid1.RowCount+1;
                          Form4.StringGrid1.Cells[0,Form4.StringGrid1.RowCount-2]:=serwer;
                          Form4.StringGrid1.Cells[1,Form4.StringGrid1.RowCount-
2]:=inttostr(1);
                          Form4.StringGrid1.Cells[2,Form4.StringGrid1.RowCount-
2]:=inttostr(0);
                        end; end;
                       for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                        s.s1:=Form4.StringGrid1.Cells[0,weri];
                        s.n1:=strtoint(Form4.StringGrid1.Cells[1,weri]);
                        s.n2:=strtoint(Form4.StringGrid1.Cells[2,weri]);
                        write(f,s); end;
                      msg:='y';
                      if ServerSocket1.Active
                      then ServerSocket1.Socket.Connections[0].SendText(msg)
                      else ClientSocket1.Socket.SendText(msg);
                      Form4.ShowModal;  end; end;
                   2:begin
                     plc.s:=true;
                     plc.mp[plc.i,plc.j]:=0;
                     msg:=inttostr(plc.i)+inttostr(plc.j);
                     plc.i:=a;plc.j:=b;
                     msg:=msg+inttostr(a)+inttostr(b);
                     plc.mp[a,b]:=plc.k;
                     mas[plc.k].Setij(a,b);
                     if ((g.ud=1) and (b=1))
                     or ((g.ud=2) and (b=8))
                     then mas[plc.k].SetT(2);
                     plc.t:=YesStep(plc.i,plc.j);
```

```
                    if not(plc.t) then begin plc.k:=0;revers; end;
                    msg:=msg+inttostr(plc.kn);
                    if s1>9 then msg:=msg+inttostr(s1)
                          else msg:=msg+'0'+inttostr(s1);
                    if ServerSocket1.Active
                    then ServerSocket1.Socket.Connections[0].SendText(msg)
                    else ClientSocket1.Socket.SendText(msg);
                    if Victoria then begin
                      messagedlg('Вы победили',mtinformation,[mbyes],0);
                      op:=false;
                      Form4.FormShow(Sender);
                      AssignFile(f,'Victor.txt');
                      rewrite(f);
                      For werj:=1 to 2 do begin
                        if serwer=name2 then serwer:=g.name else serwer:=name2;
                        for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                          if Form4.StringGrid1.Cells[0,weri]=serwer then begin
                            Form4.StringGrid1.Cells[werj,weri]:=inttostr(strtoint(Form4.String-
Grid1.Cells[werj,weri])+1);
                            op:=true;
                          end;
                        end;
                        if op=false then begin
                          Form4.StringGrid1.RowCount:=Form4.StringGrid1.RowCount+1;
                          Form4.StringGrid1.Cells[0,Form4.StringGrid1.RowCount-2]:=serwer;
                          Form4.StringGrid1.Cells[1,Form4.StringGrid1.RowCount-
2]:=inttostr(1);
                          Form4.StringGrid1.Cells[2,Form4.StringGrid1.RowCount-
2]:=inttostr(0);
                        end;
                      end;
                      for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
                        s.s1:=Form4.StringGrid1.Cells[0,weri];
                        s.n1:=strtoint(Form4.StringGrid1.Cells[1,weri]);
                        s.n2:=strtoint(Form4.StringGrid1.Cells[2,weri]);
                        write(f,s);
                      end;
                      msg:='y';
                      if ServerSocket1.Active
                      then ServerSocket1.Socket.Connections[0].SendText(msg)
                      else ClientSocket1.Socket.SendText(msg);
                      Form4.ShowModal;
                    end;
                  end;
                end;
```

```
                end;
            end;
        end;
    end;
    procedure TForm1.ClickImage(Sender: TObject; Button: TMouseButton; //Событие при
нажатии на шашку
      Shift: TShiftState; X, Y: Integer);
    var t:Timage;
       h:integer;
    begin
    if g.n=kn[plc.kn]
    then begin
        t.BringToFront;
        h:=strtoint(t.Hint);
        if plc.s then if plc.k<>0 then if plc.k<>h then exit;
        if ((g.wb=1) and (h<=12))
        or ((g.wb=2) and (h>12))
        then begin
            plc.k:=h;
            plc.t:=true;
            plc.wb:=tip(h);
            plc.i:=mas[h].i;
            plc.j:=mas[h].j;
            plc.s:=false;
            end;
        end;
    end;
    procedure TForm1.SetGame(f:byte);
    begin
    g.n:=f;
    plc.kn:=f;
    kn[plc.kn]:=f;
    plc.kn:=2;
    g.wb:=f;
    g.ud:=f;
    end;
    procedure TForm1.Read(Socket:TCustomWinSocket);
    var s:string;
       i,j,a,b,c:integer;
       n:Tcursor;
       op:boolean;
       Sender:Tobject;
       serwer:string;
       weri,werj:integer;
       sss:rec;
```

```
      F:file of rec;
    begin
    s:=Socket.ReceiveText;
    case length(s) of
    1:begin
      messagedlg('Вы проиграли',mtinformation,[mbyes],0);
      op:=false;
      Form4.FormShow(Sender);
      AssignFile(f,'Victor.txt');
      rewrite(f);
      for werj:=1 to 2 do begin
        if (serwer=g.name) then serwer:=name2 else serwer:=g.name;
        for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
          if Form4.StringGrid1.Cells[0,weri]=serwer then begin
            Form4.StringGrid1.Cells[werj,weri]:=inttostr(strtoint(Form4.StringGrid1.Cells[werj,
weri])+1);
            op:=true;
          end;
        end;
        if op=false then begin
          Form4.StringGrid1.RowCount:=Form4.StringGrid1.RowCount+1;
          Form4.StringGrid1.Cells[0,Form4.StringGrid1.RowCount-2]:=serwer;
          Form4.StringGrid1.Cells[1,Form4.StringGrid1.RowCount-2]:=inttostr(0);
          Form4.StringGrid1.Cells[2,Form4.StringGrid1.RowCount-2]:=inttostr(1);
        end;
      end;
      for weri:=1 to Form4.StringGrid1.RowCount-2 do begin
        sss.s1:=Form4.StringGrid1.Cells[0,weri];
        sss.n1:=strtoint(Form4.StringGrid1.Cells[1,weri]);
        sss.n2:=strtoint(Form4.StringGrid1.Cells[2,weri]);
        write(f,sss);
      end;
      Form4.ShowModal;
    end;
    2:begin
      Stroi(1);
      messagedlg('Новая игра',mtinformation,[mbyes],0);
    end;
    5,7:begin
      i:=strtoint(copy(s,1,1));
      j:=strtoint(copy(s,2,1));
      a:=strtoint(copy(s,3,1));
      b:=strtoint(copy(s,4,1));
      plc.mp[a,b]:=plc.mp[i,j];
      plc.mp[i,j]:=0;
```

```
    c:=strtoint(copy(s,5,1));
    if plc.kn<>c then begin
    Cursor:=crarrow;
    for k:=1 to 24 do mas[k].Im.Cursor:=crNo;
    if g.ud=1 then n:=12 else n:=24;
    for k:=n-11 to n do mas[k].Im.Cursor:=crHandPoint;
    Label4.Caption:='Ваш ход';
    Label4.Font.Color:=clred;
    end;
    mas[plc.mp[a,b]].Setij(a,b);
    if ((g.ud=2) and (b=1))
    or ((g.ud=1) and (b=8))
    then mas[plc.mp[a,b]].SetT(2);
    plc.kn:=c;
    if length(s)=7
    then begin
        a:=strtoint(copy(s,6,2));
        plc.mp[mas[a].i,mas[a].j]:=0;
        mas[a].Im.Visible:=false;
        end;
    end;
else begin
     name2:=copy(s,9,length(s)-8);
     label2.Caption:='Соперник: '+name2;
     N6.Enabled:=false;     N7.Enabled:=false;
     end;
end;
end;
procedure TForm1.ClientSocket1Read(Sender: TObject;
  Socket: TCustomWinSocket);
begin
Read(Socket);
end;
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin
Read(Socket);
end;
procedure TForm1.N6Click(Sender: TObject);
begin
Form2.Show;
end;
procedure TForm1.N7Click(Sender: TObject);
begin
Form3.Show;
```

```
    end;
    procedure TForm1.N2Click(Sender: TObject);
    begin
    if messagedlg('Вы хотите начать с начала?',mtconfirmation,[mbyes,mbno],0)=mryes
then begin
    Stroi(1);
    msg:='да';
    if ServerSocket1.Active
    then ServerSocket1.Socket.Connections[0].SendText(msg)
    else ClientSocket1.Socket.SendText(msg);
    end;
    end;
    procedure TForm1.FormShow(Sender: TObject);
    begin
    g.wb:=1;
    CreatPlace;
    Stroi(1);
    end;
    procedure TForm1.N8Click(Sender: TObject);
    begin
    if ClientSocket1.Active then ClientSocket1.Active:=false;
    if ServerSocket1.Active then ServerSocket1.Active:=false;
     Cursor:=crarrow;
        for k:=1 to 24 do mas[k].Im.Cursor:=crarrow;
            N6.Enabled:=true;     N7.Enabled:=true;
    end;
    procedure TForm1.ServerSocket1ClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    begin
    ServerSocket1.Socket.Connections[0].SendText('00000000'+g.name)
    end;
    procedure TForm1.ClientSocket1Connect(Sender: TObject;
      Socket: TCustomWinSocket);
    begin
    ClientSocket1.Socket.SendText('00000000'+g.name);
    end;
    function TForm1.stepy(k:integer):boolean;
    begin
    stepy:=false;
    if mas[k].tp=1
    then begin
        if YesStep(mas[k].i,mas[k].j) then stepy:=true
        else begin
            case 2-g.ud+1 of
            1:if (mas[k].i+1<=8) and (mas[k].j+1<=8)
```

```
            and (plc.mp[mas[k].i+1,mas[k].j+1]=0)
            then stepy:=true;
          2:if (mas[k].i-1>=1) and (mas[k].j>=1)
            and (plc.mp[mas[k].i-1,mas[k].j-1]=0)
            then stepy:=true;
          end       end;    end
    else begin    end; end;
    function TForm1.Victoria:boolean;
    var i,n:integer;
    begin
    victoria:=true;
    if g.ud=2 then n:=12 else n:=24;
    for i:=n-11 to n do if mas[i].Im.Visible and stepy(i) then Victoria:=false;
    end;
    procedure TForm1.N9Click(Sender: TObject);
    begin
    Form4.ShowModal;
    end;
    procedure TForm1.N4Click(Sender: TObject);
begin
close;
end;
end.
```

4 Контрольные вопросы

1) Актуальность процесса программирования для взаимодействия удаленных компьютеров сети.
2) Возможности инструментальной среды Delphi 6.0 для программирования по протоколу TCP/IP.
3) Понятие и назначение сокета.
4) Технология сокетов и типы сокетов.
5) Семейства каких протоколов поддерживает Windows socket?
6) Какой компонент Delphi 6.0 используется для создания сокета, инициирующего соединение с удаленным узлом сети?
7) Какой компонент Delphi 6.0 используется для создания сокета, отвечающего на запросы с других машин?
8) Какие параметры необходимо указать Сокету для работы?
9) Для чего необходим IP-адрес?
10) Понятие компьютерного порта.
11) Понятие протокола, примеры существующих протоколов.
12) Какой протокол может использоваться для обмена данными без подтверждения о приеме?
13) Дать обзор некоторых компонентов страницы Internet.
14) Назначение и формат вызова функции SendText.
15) Назначение и формат вызова функции ReceiveText.

16) Какая вкладка палитры компонентов Delphi 6.0 содержит компоненты ServerSocket и ClientSocket?
17) Обзор свойств и методов компонента ServerSocket.
18) Обзор свойств и методов компонента ClientSocket.
19) Варианты запуска инструментальной среды Delphi 6.0.
20) Описать алгоритм работы сокета для клиентского приложения.
21) Описать алгоритм работы сокета для приложения сервера.
22) Результат выполнения операции ServerSocket1.Active := True?
23) Пояснить назначение обработчика события OnClientConnect для компонента ServerSocket1.
24) Пояснить назначение обработчика события OnAccept для компонента ServerSocket1.
25) Назначение свойства Host компонента ClientSocket1.
26) Когда инициируется событие OnDisconnect для компонента ClientSocket1?
27) Пояснить назначение обработчика события OnError для компонента ClientSocket1.
28) Результат выполнения операции ClientSocket1.Active := False?
29) Общие алгоритмические принципы взаимодействия клиентского и серверного приложений.
30) Варианты сохранения проекта и рабочих модулей в инструментальной среде Delphi 6.0.

Заключение

Методические указания предназначены для выполнения лабораторной работы по созданию сетевых программ по протоколу TCP/IP применительно к среде Delphi 6.0.

Среда Delphi 6.0 обеспечивает высокоэффективную работу программиста, позволяет с небольшими затратами сил и времени создавать прикладные программы, удовлетворяющие всем требованиям Windows. Простота и естественность языка, ориентация системы на разработку именно такого рода приложений сделали Delphi незаменимым средством разработки различного рода программ.

В настоящие методические указания были включены подробные руководства по программированию, включая процесс визуализации и, собственно, написание кода программы в инструментальной среде программирования Delphi 6.0.

Целью данных методических указаний являлось обучение студентов работе с протоколом TCP/IP на основе архитектуры “клиент-сервер” для выполнения лабораторной работы по дисциплине “Сети ЭВМ и телекоммуникации”.

Методические указания состоят из введения, четырех глав, заключения, списка использованных источников и приложений.

Так, в первой главе методических указаний рассматривались основы реализации межсетевого взаимодействия на основе протокола TCP/IP, был дан обзор основных сетевых компонентов, подробно описывались свойства и события компонентов ServerSocket и ClientSocket.

Вторая глава была посвящена рассмотрению вопроса обмена сообщениями на базе сетевых компонентов Delphi 6.0. В данной главе подробно рассматривались процессы разработки chat-программ с интегрированными и раздельно организованными клиентскими и серверными частями. Т.е. варианты, когда клиентская и серверная части реализованы в одном приложении. Также рассматривается вариант, когда клиент и сервер реализованы различными программными кодами.

В третьей главе методических указаний рассматривался процесс создания сетевых компьютерных игр на примере игры в Шашки, было дано подробное руководство по созданию сетевого приложения данного рода, приводились фрагменты программного кода.

В четвертой главе приведены контрольные вопросы для закрепления изученного материала. И, наконец, в приложениях приведены варианты заданий на лабораторную работу, а также правила оформления результатов лабораторной работы.

Таким образом, по окончании изучения представленных методических указаний студенты будут уметь писать программы на основе архитектуры “клиент-сервер” по обмену текстовыми сообщениями, а так же различные сетевые игры с использованием протокола TCP/IP в среде Delphi 6.0.

Список использованных источников

1 **Архангельский, А.Я.** Программирование в Delphi 6 – М.: ЗАО “Издательство БИНОМ”, 2003. – 1120 с., ил.

2 **Фаронов, В.В.** Delphi 6. Учебный курс. – М.: Издатель Молгачева С.В., 2001. – 672 с.

3 **Баженова, И.Ю.** Delphi 6. Самоучитель программиста. – М.: Кудиц-Образ, 2002. – 432 с.

4 **Пятибратов, А.П**. и др. Вычислительные системы, сети и телекоммуникации: Учебник. – М.: Финансы и статистика, 2001. – 512 с.

5 **Бройдо, В.Л.** Вычислительные системы, сети и телекоммуникации. – СПб.: Питер, 2003. – 688 с.

6 **Гофман, В.Э.** Delphi 6 /В.Э. Гофман, А.Д. Хомоненко. – СПб.: БХВ – Петербург, 2002. – 1152 с.

7 **Олифер, В.Г.** Компьютерные сети. Принципы, технологии, протоколы / В.Г. Олифер, Н.А. Олифер.- СПб.: Издательство “Питер”, 2000. - 672 с.

8 **Гайсина**, **Л.Ф.** Сетевые информационные технологии. учебное пособие. – Оренбург: ИПК ГОУ ОГУ, 2005. – 167 с.

9 **Основы сетей передачи данных** / В.Г. Олифер, Н.А. Олифер. – М.: ИНТУИТ.РУ “Интернет-Университет Информационных технологий”, 2003. – 248 с.

10 **Орлов, С.А.** Технологии разработки программного обеспечения. учебное пособие. 2-е изд. – СПб.: Питер, 2003. – 480 с.

11 **Гайсина, Л.Ф.** Основы TCP/IP. методические указания к лабораторному практикуму. – Оренбург: ГОУ ОГУ, 2005. – 68 с.

12 **Чиртик, А. А.** Delphi. Трюки и эффекты /А.А. Чиртик, В.В. Борисок, Ю.И. Корвель, СПБ.: Питер, 2006. – 400 с.

13 **Кэнту, М.** Delphi 2005. Для профессионалов, СПБ.: Питер, 2006. – 912 с.

14 **Дансмор, Брэд,** Справочник по телекоммуникационным технологиям /Б. Дансмор, Т. Скандьер: Издательский дом “Вильямс”, 2004. – 640 с. ISBN 5-8459-0562-1.

15 **Гордеев, А. В.** Системное программное обеспечение /А.В. Гордеев, А.Ю. Молчанов. - СПб.: Питер, 2003. – 736 с.

16 **Бобровский, С.И.** Технологии Delphi 2006. Новые возможности, СПБ.: Питер, 2006. – 288 с.

17 **Олифер, В.Г.** Новые технологии и оборудование IP-сетей /В.Г. Олифер, Н.А. Олифер, - СПб, 2000. - 512 с.

18 **Комер, Д.** Межсетевой обмен с помощью TCP/IP [Электронный ресурс]. - Режим доступа: http:// lemoi-www.dvgu.ru/lect/protoc/tcpip/comer/ contents.htm.

19 **Брежнев, А.Ф.** Семейство протоколов TCP/IP [Электронный ресурс]/ А.Ф.Брежнев, Р.Л. Смелянский. - Режим доступа:WWW.URL: http://lemoi-www.dvgu.ru/lect/protoc/tcpip/tcpip/index.htm.

Приложение А
(справочное)
Варианты заданий на лабораторную работу

Написать программу клиент-сервер, используя протокол TCP/IP. Номер порта должен быть >=1024.

1 Имитация работы “Chat” (обмен тестовыми сообщениями),
2 Сетевая игра “Шашки”,
3 Сетевая игра “Дурак”,
4 Сетевая игра “Покер”,
5 Сетевая игра “Крестики-нолики“(n=10),
6 Сетевая игра “Морской бой” (n=10),
7 Сетевая программа “Криптография” (шифрование и дешифрование сообщений по сети),
8 Сетевая программа заказа билетов,
9 Сетевая игра “Волейбол”,
10 Сетевая игра “Теннис”,
11 Сетевая игра “Футбол”,
12 Сетевая игра “Танковые баталии” (защита штаба) (n=10),
13 Сетевая игра “Шахматы”,
14 Сетевая игра карты “21 очко”,
15 Автосимулятор “гонки”,
16 Сетевая игра “Тир”,
17 Сетевая игра “Хоккей”,
18 Сетевая игра “Водное поло”,
19 Имитация работы электронной почты,
20 Сетевая игра “Пасьянс”,
21 Сетевая игра “Поле чудес”,
22 Сетевая игра “Нарды”,
23 Сетевая игра “Города”,
24 Сетевая игра “Бильярд”,
25 Сетевая игра “Боулинг”,
26 Сетевая игра “Истребитель” (стрельба по движущимся мишеням),
27 Карточная сетевая игра “Червы”.

Лабораторная работа может быть представлена либо одной программой, включающей модуль клиент и модуль сервер, так и отдельными клиентской и серверной программой. В первом случае в качестве IP – адреса использовать Localhost 127.0.0.1. Во втором случае – соответствующие IP- адреса машин, на которых запускаются программы.

Приложение Б
(обязательное)
Структура отчета

Отчет по лабораторной работе дожен содержать следующие пункты:

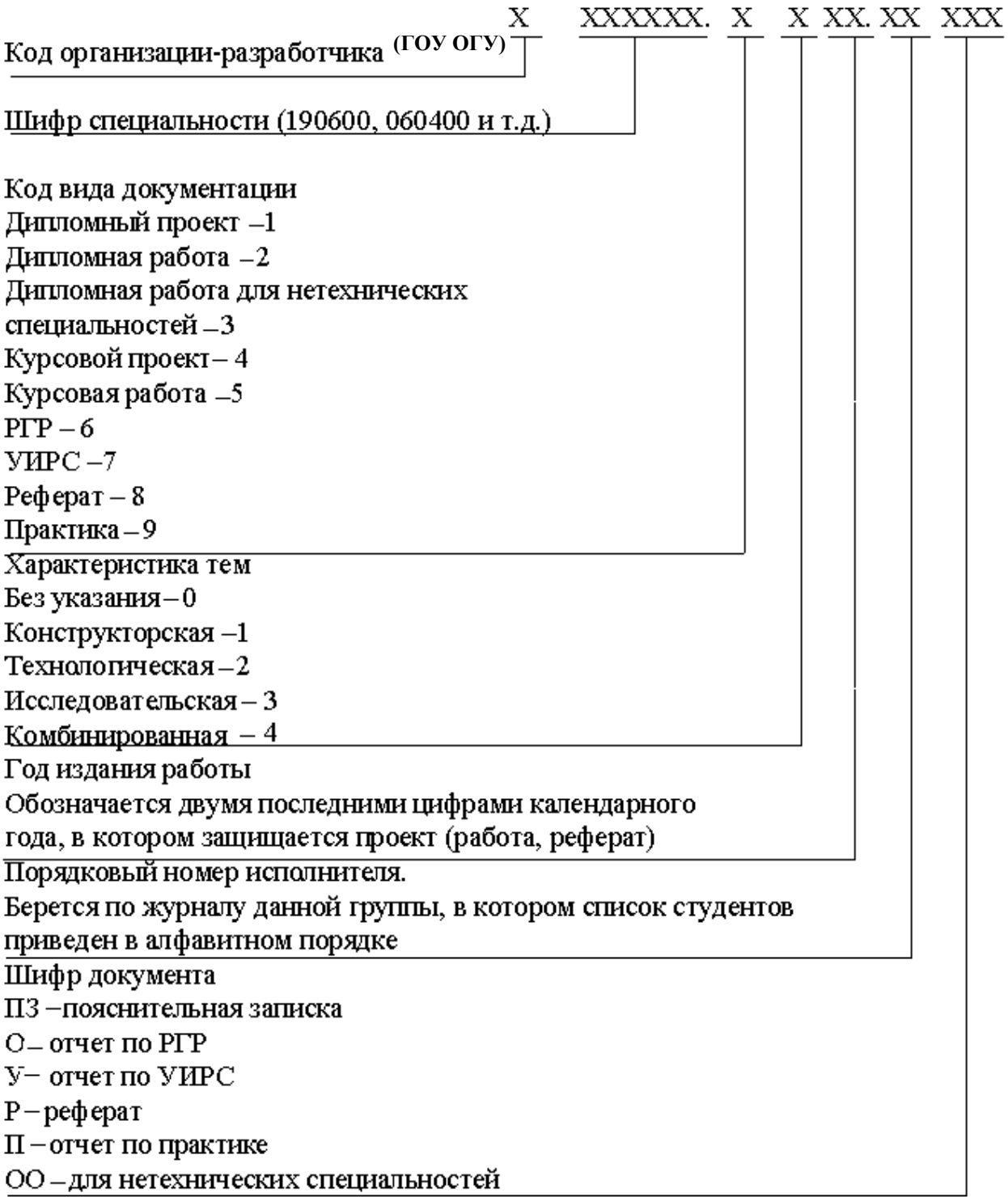
1 Титульный лист.
2 Постановка задачи.
3 Содержание.
4 Теоретические предпосылки.
5 Список использованных источников.
6 Приложение А Программный код.
7 Приложение Б Схема алгоритма программы и ее основных модулей.
8 Приложение В Контрольный пример работы программы.

Отчет по лабораторной работе выполняется на листах формата А4, каждый лист, кроме первого нумеруется. Все страницы отчета должны содержать рамку: всерху, справа, снизу отступ - 0,5 см., слева - 2 см. В правом нижнем углу каждой страницы должен быть угловой штамп (высота 15 мм, ширина 10 см), разделенный на два прямоугольника. Верхний (высота 7 мм) содержит надпись *Лист*, нижний (высота 8 мм) – номер листа.

Приложение В

(справочное)

Правила присвоения классификационного кода



Приложение Г
(справочное)
Пример оформления титульного листа

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
“ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет информационных технологий

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

Лабораторная работа

по дисциплине “Сети ЭВМ и телекоммуникации”

“Разработка клиент-серверных приложений на основе протокола TCP/IP ”

ГОУ ОГУ 230105.65.9007.02 О

Руководитель работы
_____Насейкина Л.Ф.
“____”_____2007г.
Исполнитель
студент гр. 03ПО1
_____Баранов А.И.
“____”_____2007г.

Оренбург 2007