

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ**

Г. Г. Швачич, О. В. Овсянніков, Л.М.Петречук

Методичні вказівки до виконання лабораторних робіт з дисципліни

Комп'ютерні мережі та телекомунікації
для студентів спеціальностей 6.020105 «Документознавство та інформаційна
діяльність»

Дніпропетровськ НМетАУ 2010

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ**

Г. Г. Швачич, О. В. Овсянніков, Л.М.Петречук

Методичні вказівки до виконання лабораторних робіт з дисципліни

Комп'ютерні мережі та телекомунікації
для студентів спеціальностей 6.020105 «Документознавство та інформаційна
діяльність»

Дніпропетровськ НМетАУ 2010

УДК 004 (075.8)

Г.Г. Швачич, О.В. Овсянніков, Л.М.Петречук. **Документознавство та інформаційна діяльність.** Методичні вказівки. – Дніпропетровськ: – НМетАУ, 2010. – 48 с.

Викладені основи ознайомлення з сучасними засобами роботи в комп'ютерних мережах та розробки Windows додатків.

Призначений для студентів спеціальностей «Документознавство та інформаційна діяльність».

Лл. 31. Бібліогр.: 5 найм.

Відповідальний за випуск

Г.Г. Швачич, канд. техн. наук, проф.

Рецензенти: Б. И. Мороз д-р техн. наук, проф. (Академія митної Служби України)

Т. И. Пашова канд. техн. наук, доц. (Дніпропетровський державний аграрний університет)

© Національна металургійна академія України, 2010

Лабораторная работа №1

Тема: Физическая и логическая структура сети.

Цель работы: Ознакомление с архитектурой и структурой реальной сети.

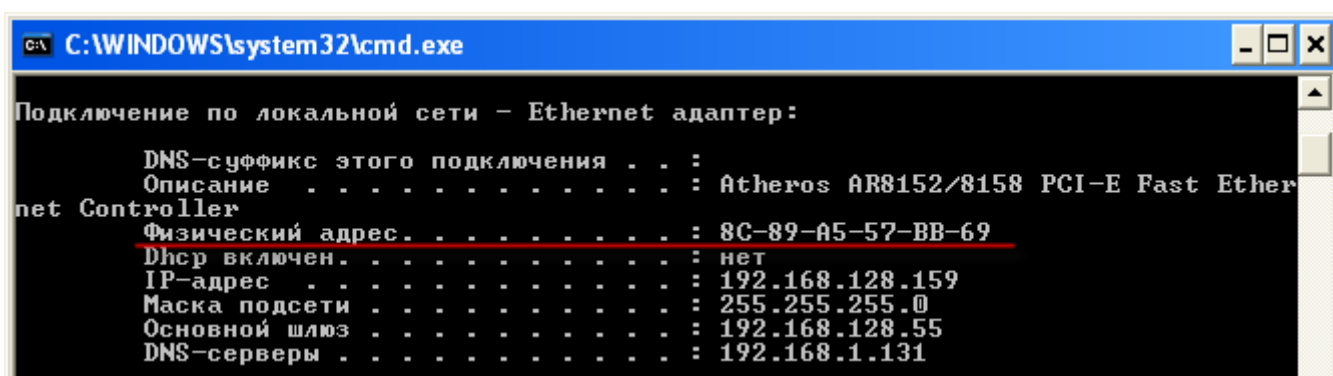
Физический адрес компьютера. MAC-адрес (от англ. Media Access Control — управление доступом к среде, также Hardware Address) — это уникальный идентификатор, присваиваемый каждой единице оборудования компьютерных сетей.

MAC-адрес жестко “зашивается” в сетевую карту ее производителем и обычно записывается в виде 12 шестнадцатеричных цифр (например, 00-03-BC-12-5D-4E). Это гарантированно уникальный адрес: первые шесть символов идентифицируют фирму-производителя, которая следит, чтобы остальные шесть символов не повторялись на производственном конвейере. Когда у машины заменяется сетевой адаптер, то меняется и ее MAC-адрес.

Узнать MAC-адрес сетевой карты вашего компьютера можно следующим образом:

1. Зайдите в “Пуск” – “Выполнить” – введите с клавиатуры команду `cmd` – “ОК”.
2. Введите команду `ipconfig /all` и нажмите клавишу Enter.

Данная команда позволяет получить полную информацию обо всех сетевых картах ПК. Поэтому найдите в этом окошке строку Физический адрес – в ней будет обозначен MAC-адрес вашей сетевой карты. В моем случае это выглядит так:



```
C:\WINDOWS\system32\cmd.exe
Подключение по локальной сети - Ethernet адаптер:
    DNS-суффикс этого подключения . . . :
    Описание . . . . . : Atheros AR8152/8158 PCI-E Fast Ether
net Controller
    Физический адрес . . . . . : 8C-89-A5-57-BB-69
    Dhcp включен . . . . . : нет
    IP-адрес . . . . . : 192.168.128.159
    Маска подсети . . . . . : 255.255.255.0
    Основной шлюз . . . . . : 192.168.128.55
    DNS-серверы . . . . . : 192.168.1.131
```

Сетевой адрес компьютера. Сетевой адрес, или IP-адрес используется в сетях TCP/IP при обмене данными на сетевом уровне. IP расшифровывается как Internet Protocol – протокол интернета. IP-адрес компьютера имеет длину 32 бита и состоит из четырех частей, именуемых *октетами*. Каждый октет может принимать значения от 0 до 255 (например, 90.188.125.200). Октеты отделяются друг от друга точками.

IP-адрес компьютера, например 192.168.1.10, состоит из двух частей – номера сети (иногда называемого идентификатором сети) и номера сетевого компьютера (идентификатора хоста).



Номер сети должен быть одинаковым для всех компьютеров сети и в нашем примере номер сети будет равен 192.168.1. Номер компьютера должен быть уникален в данной сети, и компьютер в нашем примере имеет номер 10.

IP-адреса компьютеров в разных сетях могут иметь одинаковые номера. Например, компьютеры с IP-адресами 192.168.1.10 и 192.168.15.10 хоть и имеют одинаковые номера (10), но принадлежат к разным сетям (1 и 15). Поскольку адреса сетей различны, то компьютеры не могут быть спутаны друг с другом.

Чтобы отделить номер сети от номера компьютера, применяется маска подсети. Чисто внешне маска подсети представляет собой такой же набор из четырех октетов, разделенных между собой точками. Но, как правило, большинство цифр в ней – это 255 и 0.

255 указывает на биты, предназначенные для адреса сети, в остальных местах (которым соответствует значение 0) должен располагаться адрес компьютера. Чем меньше значение маски, тем больше компьютеров объединено в данную подсеть. Маска сети присваивается компьютеру одновременно с IP-адресом. Чтобы было понятно, приведем простой пример:

- сеть 192.168.0.0 с маской 255.255.255.0 может содержать в себе компьютеры с адресами

от 192.168.0.1 до 192.168.0.254,

- а сеть 192.168.0.0 с маской 255.255.255.128 допускает адреса от 192.168.0.1 до 192.168.0.127.

Сети с большим количеством компьютеров делят на части, называемые подсетями. Деление на подсети применяется для обеспечения повышенной безопасности и разграничения доступа к ресурсам различных подсетей. Компьютеры разных подсетей не смогут передавать пакеты друг другу без специального устройства – *маршрутизатора*, а, следовательно, никто не сможет проникнуть в защищенную таким образом подсеть. Чтобы создать подсети, часть места в IP-адресе, отведенном для номера хоста, отдают под номера подсети.

Рассмотрим пример, когда у нас в локальной сети 30 компьютеров и требуется настроить их так, чтобы 20 компьютеров могли “общаться” между собой, но не смогли передавать и принимать данные от остальных 10 компьютеров, которые также должны общаться только между собой. Решение

этой задачи довольно простое – делим нашу сеть на две подсети. В первой подсети “раздаем” компьютерам (их у нас 20) номера из диапазона 192.168.1.1 – 192.168.1.20, а во второй подсети для оставшихся 10 компьютеров раздаем номера из диапазона 192.168.2.1 – 192.168.2.10.

Если ваш компьютер подключен к локальной сети или интернет, вы можете узнать его IP-адрес и маску подсети уже знакомым нам способом:

1. Зайдите в “Пуск” – “Выполнить” – наберите cmd и нажмите “ОК”.
2. В открывшемся окне введите команду ipconfig /all и нажмите клавишу Enter.

Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Интернет (Network Information Center – NIC), если сеть должна работать как составная часть Интернет. Обычно интернет-провайдеры получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами. Это внешние IP-адреса (доступные из интернета), например 90.188.125.200.

Для локальных сетей зарезервированы *внутренние* IP-адреса (к ним нельзя получить доступ через интернет без специального ПО) из диапазонов:

192.168.0.1 – 192.168.254.254

10.0.0.1 – 10.254.254.254

172.16.0.1 – 172.31.254.254

Из этих диапазонов вы, как системный администратор, и будете назначать адреса компьютерам в вашей локальной сети. Если вы “жестко” зафиксируете IP-адрес в настройках компьютера, то такой адрес будет называться *статическим* – это постоянный, неизменяемый IP-адрес ПК.

Существует и другой тип IP-адресов – динамические, которые изменяются при каждом входе компьютера в сеть. За управление процессом распределения динамических адресов отвечает служба DHCP.

DHCP (англ. Dynamic Host Configuration Protocol — протокол динамической настройки узла) — сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер».

Имя сетевого компьютера. Помимо *физического* и *сетевого* адресов компьютер может также иметь *символьный* адрес – **имя компьютера**. Имя компьютера – это более удобное и понятное для человека обозначение компьютера в сети. Различают:

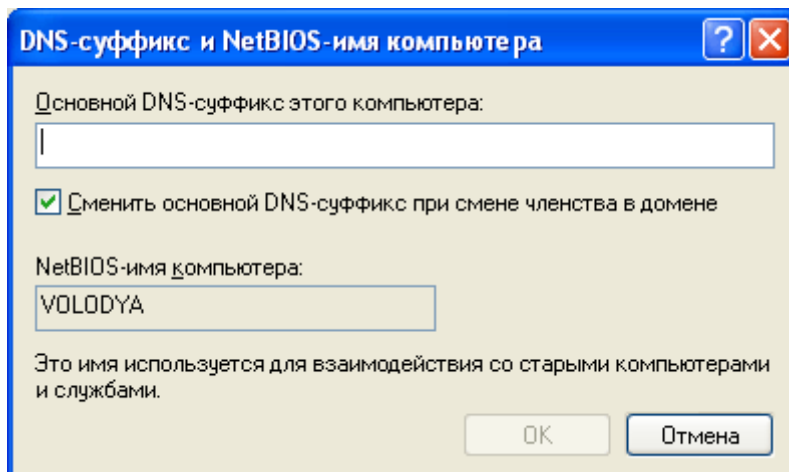
- NetBIOS имена;
- и полные **доменные** имена компьютеров.

Имена NetBIOS используются в одноранговых локальных сетях, в которых компьютеры организованы в рабочие группы. **NetBIOS** – *протокол для взаимодействия программ через компьютерную сеть*. Протокол NetBIOS распознает обычные буквенные имена компьютеров и отвечает за передачу данных между ними. Проводник Windows для просмотра локальной сети предоставляет папку **Сетевое окружение**, автоматически отображающей имена NetBIOS компьютеров вашей локальной сети.

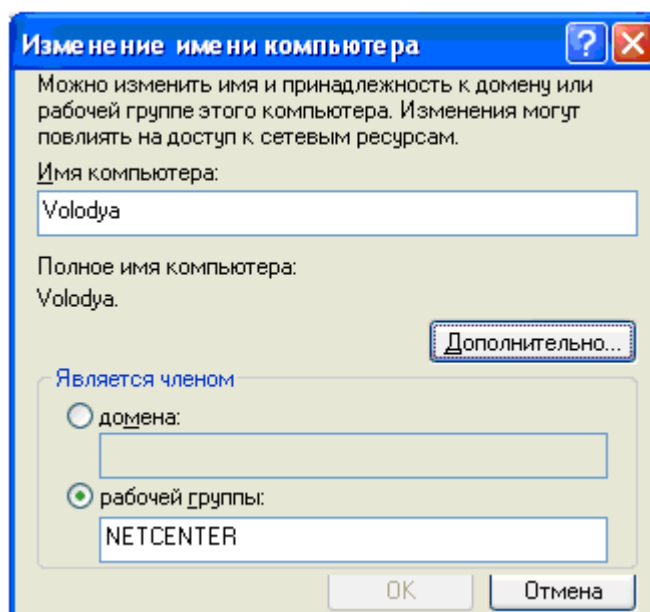
Имя NetBIOS может содержать не более 15 символов и должно быть на английском языке.

Чтобы узнать NetBIOS-имя вашего компьютера выполните следующие действия:

1. Щелкните правой кнопкой мыши по значку “Мой компьютер” на рабочем столе – выберите “Свойства”.
2. Перейдите на вкладку “Имя компьютера”.
3. Нажмите кнопку “Изменить” – затем “Дополнительно”.
4. Найдите строку “NetBIOS-имя компьютера”. Ниже и будет указано имя вашего ПК:



5. Нажмите кнопку “Отмена”. В окне “Изменение имени компьютера” вы можете изменить NetBIOS-имя в поле “Имя компьютера”:



В крупных иерархических сетях на базе домена используются полные **доменные имена** компьютеров, например, **webserver.ibm.com**.

Доменное Имя (англ. domain name) — уникальный идентификатор, который присваивается определенному IP-адресу (двух одинаковых быть не может). Доменное Имя это буквенный адрес компьютера. Доменное имя - это уникальное сочетание символов латинского алфавита, по которому можно идентифицировать ваш сайт среди множества других. Кроме букв, в домен могут входить цифры от 1 до 9 и символы дефиса «-», но дефис не может находиться в начале и в конце домена. Длина домена может быть от 2 до 63 символов.

Доменное имя компьютера состоит из трех составляющих:

- первая часть – имя хоста (webserver). Хост (host) - главный компьютер; ведущий узел (в сети); сервер;
- вторая – имя домена компании (ibm),
- третья – имя домена страны (например, ru – Россия) или имя одного из специальных доменов, обозначающих принадлежность домена организации к одному из профилей деятельности (com, gov, edu).

Доменное имя компьютера схематически выглядит:

ИМЯ КОМПЬЮТЕРА и ДОМЕН, В КОТОРОМ ОНО НАХОДИТСЯ.

Доменное Имя или буквенный адрес компьютера может быть:

- доменное имя первого (верхнего) уровня — first level domain;
- доменное имя второго уровня — second level domain;
- доменное имя третьего уровня — third level domain.

Доменные Имена первого уровня подразделяются на:

- **домены общего пользования**, они могут устанавливать принадлежность сайта к определенной категории или виду деятельности:

- **национальные или географические домены**, они определяют принадлежность сайта к той или иной стране или географической территории:

Выполнить. Используя команду *tracert* определите IP-адрес указанных сайтов учебных заведений и количество пройденных пакетом маршрутизаторов. Командой *ping* оцените время прохождения тестового пакета к этим же сайтам. Постройте график зависимости количества маршрутизаторов от расстояния и график зависимости времени прохождения пакета от расстояния. Определите скорость пакета в каждом направлении с учетом того, что команда *ping* определяет время прохождения пакета в одну и другую сторону.

№ п/п	адрес	учреждение
1	http://www.donnu.edu.ua/	Донецкий национальный университет
2	http://www.ifdma.if.ua	Ивано-Франковский Государственный медицинский Университет
3	http://www.franko.lviv.ua	Львовский национальный университет им. Франко
4	http://www.osaft.odessa.ua	Одесская Национальная Академия Пищевых Технологий
5	http://sumdu.edu.ua	Сумский государственный университет
6	http://univer.kharkov.ua	Харьковский Национальный Университет
7	http://kture.kharkov.ua/	Харьковский национальный университет радиоэлектроники
8	http://www.ksau.kherson.ua	Херсонский государственный аграрный университет
9	http://www.university.kherson.ua	Херсонский Государственный Университет
10	http://www.nstu.ru/	Новосибирский государственный технический университет
11	http://www.omgau.ru/	Омский государственный аграрный университет
12	http://www.samgtu.ru/	Самарский государственный технический университет
13	http://www.spbu.ru/	Санкт-Петербургский государственный университет
14	http://www.usla.ru/	Уральская государственная юридическая академия (Екатеринбург)
15	http://www.msmu.ru/	Московский государственный горный университет

16	http://www.ugatu.ac.ru/	Уфимский государственный авиационный технический университет
17	http://www.susu.ac.ru/	Южно-Уральский государственный университет
18	http://www.ystu.ru/	Ярославский государственный технический университет
19	http://www.khstu.ru	Хабаровский государственный технический университет
20	http://www.tsu.ru/	Томский государственный университет

Лабораторная работа №2

Тема: Разработка приложения **IB_CLIENT**

Цель: Изучение и работа с компонентами поддержки Midas – технологий в среде Delphi

Дизайн приложения

1. Создайте в среде *Delphi* новый проект и сохраните в собственной папке файлы проекта с именами: Unit1 -> **U_SQL**, Project1 -> **IB_SQL**.
2. Установите в форму следующие компоненты (рис.1): **Memo**, **MainMenu** (страница *Standard*), **Query**, **DataSource** (страница *Data Access*), **DBGrid** (страница *Data Controls*), **OpenDialog** и **SaveDialog** (страница *Dialogs*).
3. Присвойте форме имя **IB_Form**, а свойству **Caption** формы значение **IB_CLIENT**.
Для компонентов **Memo1** и **DBGrid1** установите значение свойств **Align := AlTop**, **Align := alClient** соответственно.
4. В редакторе свойств **Lines** компонента **Memo1** введите текст: *Select * from Country*, предварительно удалив текст *Memo1*.
5. Присвойте новую иконку форме, выбрав свойство **Icon** формы и загрузив в редактор образов изображение (файл **Factory.ico**), находящееся в папке *.../Program Files/Borland/Delphi3/Images/Icons/*.
6. Сохраните файлы проекта, выполнив команду **Save ALL** меню *Delphi*.

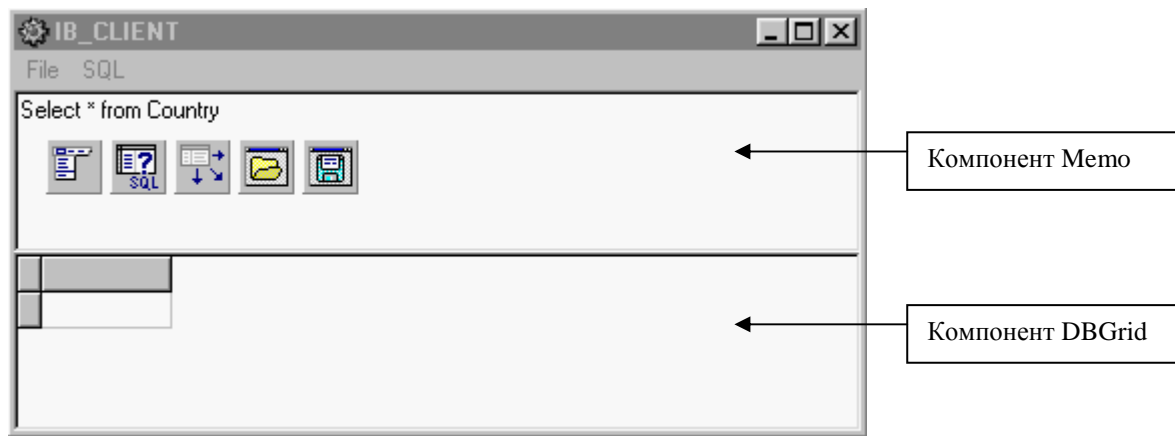


Рис. 1 Дизайн приложения

7. Войдите в редактор меню компонента **MainMenu1**, выбрав свойство **Items** или выполнив двойной щелчок мышью по компоненту **MainMenu1** в форме.
8. В редакторе меню компонента **MainMenu1** создайте группы меню **File** и **SQL** (рис. 2) последовательно выполнив следующие действия:

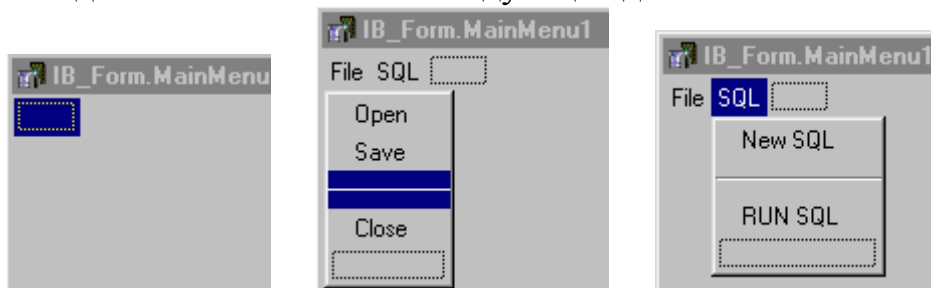


Рис. 2 Последовательность создания команд меню

- установите курсор мыши в пустой прямоугольник, который, окрасится, синим цветом, и в поле ввода свойства **Caption** введите текст **File**. Обратите внимание на тот факт, что свойству **Name** будет автоматически присвоено имя **File1**;
- для создания команд программной группы меню **File** выберите пустой прямоугольник, расположенный ниже созданного пункта меню **File**. Присвойте пустому значению свойства **Caption** значение **Open**. (имя **Open1** будет присвоено автоматически);
- аналогично создайте команду **Save**;
- для создания разделительной линии свойству **Caption** достаточно присвоить символ «-».
- последней в группе меню **File** создайте команду **Close**;
- для создания пункта меню **SQL** выберите прямоугольник, расположенный справа от пункта меню **File** и присвойте ему значение **SQL**;
- далее выберите прямоугольник, расположенный ниже созданного пункта **SQL** и присвойте свойству **Caption** значение **New** (но не **New SQL**);
- создайте разделительную линию, а затем команду меню **RUN** (но не **RUN SQL**);
- выполнив указанные действия, измените значения свойств **Caption** для команд **New** на **New SQL** и **RUN** на **RUN SQL**. Обратите внимание на тот факт, что имена команд не изменились, а остались **New1** и **RUN1**.

Примечание: В дальнейшем значения свойств **Caption** и **Name** пунктов и команд меню можно изменять. Однако свойство **Name**, являющиеся идентификатором команды меню, возможно изменять до написания программного кода команды.

9. Сохраните файлы проекта, выполнив команду **Save ALL** меню *Delphi*.

Настройка фильтров диалоговых окон

10. В форме выберите компонент **OpenDialog1**. В редакторе свойств щелкните мышью по кнопке расположенной справа от свойства **Filter**, что приведет к вызову окна редактора фильтра. Введите в поля ввода редактора текст, приведенный на (рис.3) и нажмите на кнопку **OK**.

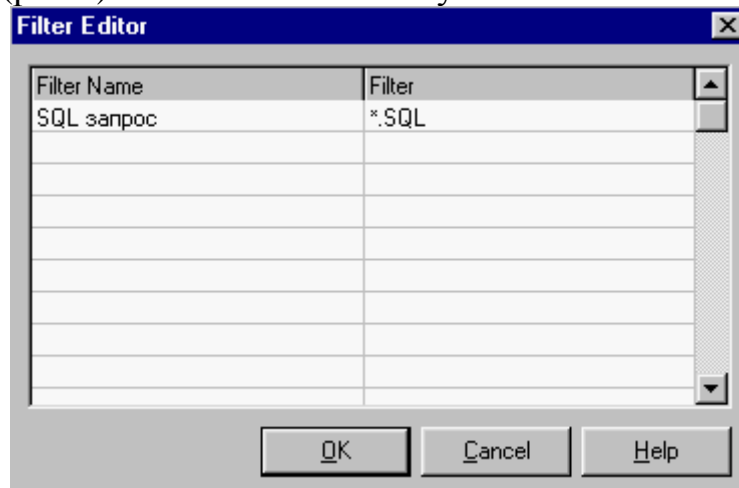


Рис. 3 Окно редактора фильтра компонента *OpenDialog1*

11. В форме выберите компонент **SaveDialog1**. В редакторе свойств компонента заполните поля свойств **DefaultExt** и **FileName** в соответствии рис. 4. В редакторе свойства **Filter** введите текст, аналогичный тексту приведенному на рис. 3.

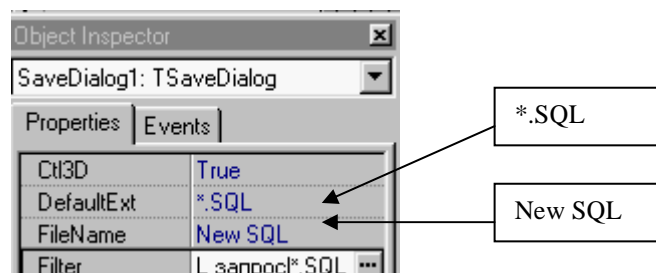


Рис. 4. Ввод значений свойств компонента *SaveDialog1*.

12. Сохраните файлы проекта, выполнив команду **Save ALL** меню *Delphi*.

Установка соединения с локальным сервером Interbase

13. Выберите в форме компонент **Query1**. Для свойства **DataBaseName** компонента выберите значение алиаса **IBLOCAL**. В редакторе свойства **SQL** введите следующий текст:

*Select * from Country*

14. Выберите в форме компонент **DataSource1**. В инспекторе объектов присвойте свойству **DataSet** значение **Query1**.

15. Выберите в форме элемент управления **DBGrid1**. В инспекторе объектов присвойте свойству **DataSource** значение **DataSource1**.

16. Для проверки корректности соединения с локальным сервером установите свойство **Active** компонента **Qwery1** в состояние **True**. Данное действие приведет к выводу окна **DataBase Logion** (рис. 6).

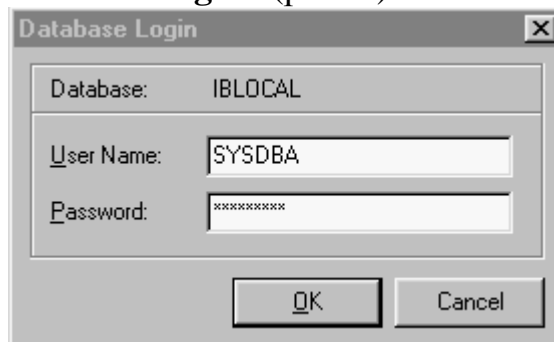


Рис. 5 Диалоговое окно соединения с локальной базой данных

17. В поле ввода **Password** необходимо ввести пароль «**masterkey**» и нажать на кнопку **OK**. Если связь установлена корректно, то в режиме дизайна приложения (без его компиляции) будет выполнен ранее введенный **SQL** запрос (рис.6).

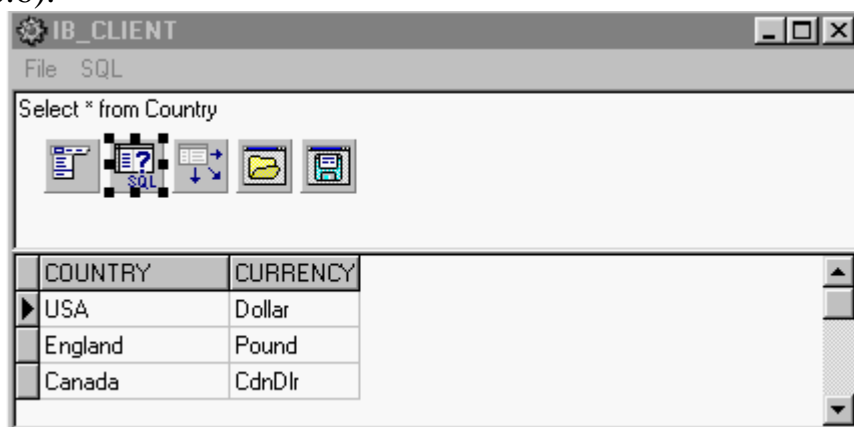


Рис. 6 SQL запрос, выполненный в период дизайна приложения

18. Деактивируйте запрос, установив свойство **Active** компонента **Qwery1** в состояние **False**.
19. Сохраните файлы проекта, выполнив команду **Save ALL** меню *Delphi*.

Написание программного кода

20. Выберите команду меню **Open** и создайте обработчик события **onClick**, выполнив щелчок мышью по пункту меню. Между ключевыми словами **begin** ... **end**; запишите текст программы, соответствующий открытию **SQL** файла:

```
procedure TIB_Form.Open1Click(Sender: TObject);  
begin  
    if OpenDialog1.Execute then  
        Memo1.Lines.LoadFromFile(OpenDialog1.FileName);  
end;
```

21. Выберите команду меню **Save** и создайте обработчик события **onClick**, выполнив щелчок мышью по пункту меню. Между ключевыми словами **begin** ... **end**; запишите текст программы, соответствующий сохранению **SQL** файла:

```
procedure TIB_Form.Save1Click(Sender: TObject);  
begin
```

```
if SaveDialog1.Execute then
  Memo1.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

22. Создайте обработчик события **onClick** команды завершения работы приложения и напишите единственную команду **Close**:

```
procedure TIB_Form.Close1Click(Sender: TObject);
begin
  Close;
end;
```

23. Для очистки окна ввода SQL запроса создайте обработчик события **onClick** команды **NEW SQL**:

```
procedure TIB_Form.New1Click(Sender: TObject);
begin
  Memo1.Clear;
end;
```

24. Процедура выполнения SQL запроса описывается в обработчике события **onClick** команды **RUN SQL**:

```
procedure TIB_Form.RUN1Click(Sender: TObject);
begin
  Query1.Active := false;
  Query1.SQL.Text := Memo1.Text;
  Query1.Active := true;
end;
```

25. Написав все строки программы, сохраните все файлы проекта (команда **Save ALL**), запустите приложение на выполнение и проверьте выполнение всех команд. При выполнении первого **SQL** запроса будет выведено окно **Data Base Logion** в поле ввода **Password** которого необходимо ввести пароль «**masterkey**».

26. Выйдите из среды *Delphi* и запустите исполняемый файл **IB_SQL.exe** из Вашей рабочей папки. Введите в окно **SQL** запроса следующее предложение:
*Select * from Country where Country = 'USA'* и выполните его.

Создание формы “О программе”

27. Войдите в *Delphi* и откройте проект приложения.

28. Создайте новую форму, выполнив команду меню *File / New Form*.

29. Присвойте новой форме имя (Name) **AboutForm**.

30. В инспекторе объектов установите следующие свойства формы:

```
Caption := ‘О программе’; BorderIcons.BiMinimize := false;  
BorderIcons.BiMaximize := false; BorderStyle := bsSingle; FormStyle :=  
fsStayOnTop и Position := poScreenCenter.
```

31. Установите в форму кнопку **BitBtn** (*страница Additional*) и в инспекторе объектов определите свойства **ModalResult := mrOk** и **Caption := ‘Ok’**.

32. Установите в форму компонент **Image** (*страница Additional*) и загрузите в компонент **Image** (*свойство Picture*) выбранную Вами картинку.

33. Самостоятельно оформите внешний вид формы в соответствии с примером, приведенном на рисунке 7, используя компоненты **Label** (*Страница Standard*).

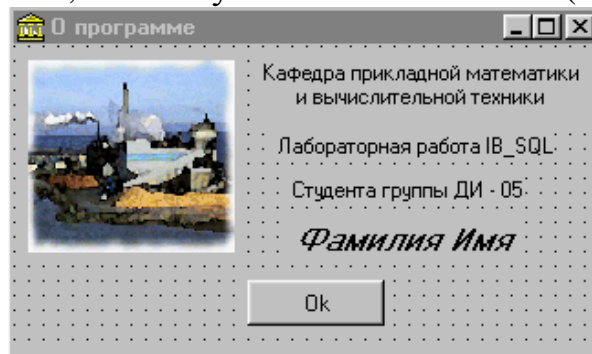


Рис. 7 Примерный вид формы “О программе”

34. Сохраните форму в папке проекта под именем **About**, выполнив команду File / **Save As**.

Подключение формы к проекту

Перед тем, как подключить форму к проекту необходимо создать новую группу меню в главной форме проекта.

35. Самостоятельно создайте новый пункт меню **Info** и команду **About**.
 36. Далее воспользуйтесь командой *ADD Project* меню *Project Delphi* и откройте файл **About.pas**.

37. Создайте процедуру обработки события для выполнения команды **About**:

```
procedure TIB_Form.About1Click(Sender: TObject);  
begin  
    AboutForm.ShowModal;  
end;
```

38. Запустите приложение на выполнение и на вопрос *Delphi* “Добавить форму” дайте утвердительный ответ.

39. Повторно запустите приложение на выполнение и проверьте выполнение команды **About**.

40. Сохраните все файлы проекта (команда *Save ALL*).

Регистрация авторских прав

41. Для регистрации авторских прав воспользуйтесь командой *Options* меню *Project Delphi*, которая вызовет окно *Project Option*.

42. Выберите страницу *Application* и загрузите ту же иконку, которую Вы присвоили главной форме проекта.

43. Выберите страницу *VersionInfo*, активизируйте опцию *Include version information* и заполните табличку приблизительно следующей информацией:

CompanyName	ИМетАУ
FileDiscription	Лабораторная работа
InternalName	IB_SQL
LegalCopyright	Фамилия Имя
OriginalFilename	IB_SQL.exe
ProductNane	Учебная программа

44. После заполнения полей таблицы нажмите на кнопку ОК и сохраните все файлы проекта.

45. Запустите приложение на выполнение

46. Выйдите из *Delphi* и прочитайте свойства исполняемого файла.

Русификация интерфейса

47. Выполните самостоятельно русификацию пунктов меню и команд меню приложения.

В период русификации установите «быстрые клавиши» для команд меню, посредством

свойства **ShortCut** редактора меню:

File		- Файл;
Open	[Ctrl + O]	- Открыть;
Save	[Ctrl + S]	- Сохранить как;
Close	[Alt + X]	- Выход;
SQL		- SQL запросы;
New SQL	[F4]	- Новый запрос;
RUN SQL	[F5]	- Выполнить запрос;
Info		- Информация;
About	[F1]	- О программе.

Сохраните файлы проекта.

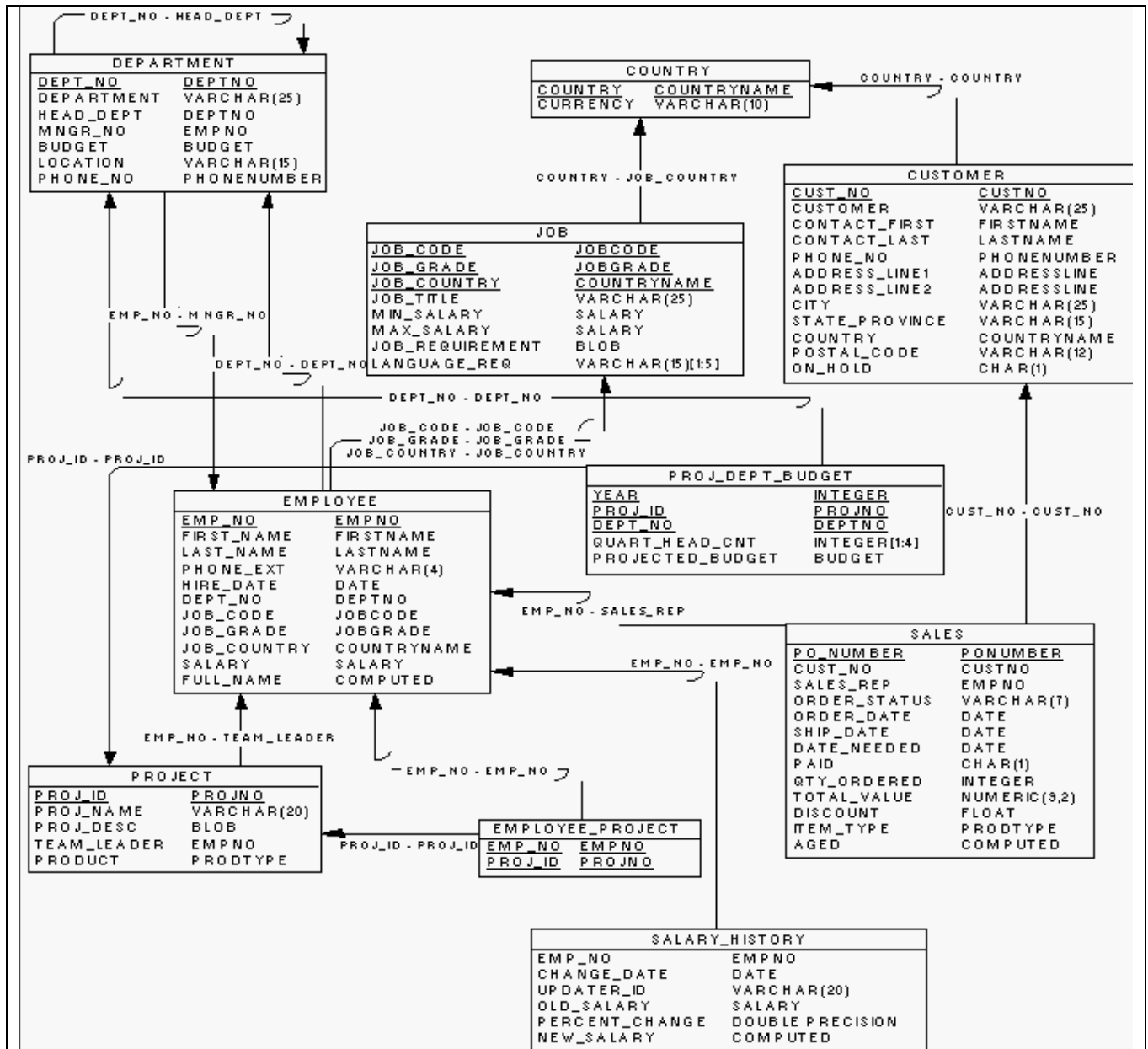
Запустите приложение на выполнение.

Изучение языка SQL

Изучение языка SQL выполняется с использованием разработанного приложения и на основе материала, изложенного в лекциях.

IB_SERVER

База данных Employee (служащие предприятия)



Лабораторная работа №3

Тема: Распределенные вычисления. Генератор отчетов в среде Excel

Цель: Изучение компонентов поддержки **Com** и протоколов среды **Delphi**.
Создание простого приложения **Com**-клиент.

Дизайн приложения

1. Откройте ранее разработанный проект **IB_SQL**.
2. Установите свойство формы **FormStyle** в состояние **fsStayOnTop** и свойство **Position** в состояние **poScreenCenter**.
3. Установите в форму компонент **ExCtrl**, расположенный на станции **Excel Control** палитры компонентов *Delphi*.
4. Создайте новый пункт меню **Excel**, используя команду **Insert** контекстного меню редактора меню и новую команду меню **Report** (рис.1).

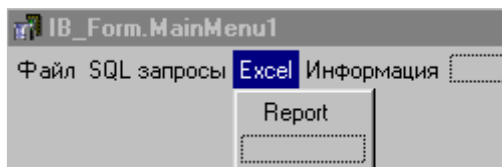


Рис. 1 Добавление пункта меню *Excel* и команды *Report*

5. Измените значение свойства **Report** команды **Report1** на **Отчет**.
6. Сохраните файлы проекта, выполнив команду **Save ALL** меню *Delphi*.

Написание программного кода

1. Для обработчика события **OnClick** пункта меню **Excel** напишите следующую программную конструкцию, которая позволит получить доступ к команде **Отчет**, только в случае, если **DBGrid** будет содержать данные результата **SQL** запроса:

```
procedure TIB_Form.Excel1Click(Sender: TObject);  
begin  
if DBGrid1.FieldCount > 0 then  
    Report1.Enabled := true  
else  
    Report1.Enabled := false;  
end;
```

2. Для обработчика события **OnClose** формы напишите программную строку, завершающую работу приложения *Excel* в случае завершения работы приложения *IB_SQL*:

```
procedure TIB_Form.FormClose(Sender: TObject; var Action: TCloseAction);
```

begin

ExCtrl1.Excel_DeActive;

end;

3. В обработчике события **OnClick** команды **Отчет** необходимо описать протокол обмена информацией между приложением **IB_SQL** и COM - сервером **Excel**. Данный протокол содержит следующие блоки программы:

- *Определение размеров таблицы*
- *Установление связи и запуск Excel*
- *Формирование заголовка документа*
- *Протокол передачи заголовков полей*
- *Протокол передачи данных*

4. Отладку протокола рекомендуется выполнять по частям (блокам), например, после написания процедуры *ExCtrl1.Excel_Active* или *ExCtrl1.Range_Write('A1','Результат выполнения SQL запроса')* и т.д.

Ниже приводится полный текст процедуры, реализующей протокол формирования отчета в *Com сервере Excel*:

procedure TIB_Form.Report1Click(Sender: TObject);

var

i, j, Ncol, Nrow :Integer;

begin

//===== Определение размеров таблицы =====

Ncol := DBGrid1.Columns.Count;

Nrow := DBGrid1.DataSource.DataSet.RecordCount;

//===== Установление связи и запуск Excel =====

ExCtrl1.Excel_Active;

//===== Формирование заголовка документа =====

ExCtrl1.Range_SetWidthCells('A1:K1',12);

ExCtrl1.Range_Alignment('A1',1);

ExCtrl1.Range_SetFontItalic('A1',true);

ExCtrl1.Range_SetFontSize('A1',14);

ExCtrl1.Range_SetFontColor('A1',clGreen);

ExCtrl1.Range_Write('A1','Результат выполнения SQL запроса');

ExCtrl1.Range_MergeCells('A1:K1');

ExCtrl1.Range_SetColor('A1:K1',clYellow);

ExCtrl1.Range_Border('A1:K1',2,clRed);

//===== Протокол передачи заголовков полей =====

for i := 1 **to** Ncol **do**

```

begin
ExCtrl1.Cells_Write(2,i,DBGrid1.Fields[i-1].DisplayName);
ExCtrl1.Cells_SetColor(2,i,clSilver);
ExCtrl1.Cells_Border(2,i,1,clBlue);
end;

//===== Протокол передачи данных =====
DBGrid1.DataSource.DataSet.First;
  for j := 1 to Nrow do
  begin
    for i := 1 to Ncol do
    begin
      ExCtrl1.Cells_Write(2+j,i,DBGrid1.Fields[i-1].AsString);
    end;
    DBGrid1.DataSource.DataSet.Next;
  end;
end;

```

После отладки протокола сохраните файлы проекта, выполнив команду **Save ALL** меню *Delphi*.

Работа с приложением IB_SQL

Работа по изучению языка *SQL* выполняется с использованием индивидуально разработанных приложений *IB_SQL* с различными базами данных, путем переопределения алиасов в проекте разработки и перекомпиляции приложения.

Лабораторная работа № 4

Тема: Изучение методов доступа к COM серверам компонента **EX_Control**.

Цель: Разработка приложения управления COM сервером. Изучение методов доступа к COM серверов компонента **EX_Control**.

В практической работе часто возникает необходимость передать данные, полученные в одном приложении другому приложению, для дальнейшей их обработки. При этом внешнее приложение (*сервер*), выполняющее дальнейшую обработку информации, может находиться на удаленном компьютере. Такие задачи реализуются различными средствами и технологиями, одной из которых является **COM** технология, поддерживаемая пакетом программ **Ms Office**.

В старших версиях среды *Delphi* (начиная с 5) в палитру компонентов включена страница **Servers**, содержащая компоненты, взаимодействующие с **COM** серверами (*Word, Excel, Binder, Access, Powr Point, Out Look и др.*) пакета **Ms Office**. Также, для 3 и 4 версий среды *Delphi* в 2001 году компанией *DeVries*

Inc. разработана библиотека компонентов *OfficePartner 1.5* (коммерческая версия), выполняющая аналогичные функции.

Для выполнения лабораторных работ по изучению возможностей распределенных вычислений и обмена данными с резидентным и удаленным **COM** сервером **Ms Excel** на кафедре *ПМуВТ НМетАУ* разработан процедурный компонент **ExCRTL**, который позволяет разрабатывать протоколы обмена информацией и осуществлять управление **Ms Excel**.

Перед выполнением лабораторных работ потребуется вспомнить работу в среде **Excel**, а также некоторые свойства ячейки, такие как **Range, Cells, Interior, Color, Font, Formula** и другие, декларированные в среде **VBA for Excel**. Также необходимо ознакомиться с методами компонента **ExCRTL**.

Примечание: Компонент **ExCRTL** должен быть установлен в среду *Delphi*. В компьютерных классах кафедры *ПМуВТ* компонент **ExCRTL** установлен в среду *Delphi 3* по умолчанию.

Основные методы компонента ExCRTL

В приведенных ниже таблицах описаны некоторые методы компонента ExCRTL, которые могут быть использованы при выполнении лабораторных работ.

Таблица 1

Методы управления приложением	Выполняемое действие
procedure Excel_Active;	<i>Запускает Excel</i>
procedure Excel_DeActive;	<i>Закрывает Excel</i>
procedure Excel_ADDSheets;	<i>Добавляет лист</i>
function Excel_GetCountSheets:byte;	Возвращает номер текущего листа
procedure Excel_SetSheetsName(sName: String; ID: Byte);	Присваивает имя листу по индексу
function Excel_GetSheetsName(ID: byte):String;	Возвращает имя листа по индексу
procedure Excel_SheetsIdSelect(ID: Byte);	Выбирает лист по индексу
procedure Excel_SheetsNameSelect(xName: String);	Выбирает лист по имени

Таблица 2

Методы управления свойством Cells	Выполняемое действие
procedure Cells_Select(Row:Word;Col:Byte);	Выбирает ячейку
procedure Cells_Write(Row:Word; Col:Byte; Data:Variant);	Записывает значения в ячейку
function Cells_Read(Row:Word; Col:Byte):Variant;	Возвращает значение ячейки
procedure Cells_FormulaWrite(Row:Word; Col:Byte; xFormula:String);	Записывает формулу в ячейку
function Cells_FormulaRead(Row:Word; Col:Byte):String;	Считывает формулу из ячейки
procedure Cells_SetName(Row:Word; Col:Byte; xName:String);	Присваивает имя ячейке
procedure	Закрашивает ячейку цветом

<pre>Cells_SetColor(Row:Word;Col:Byte;xColor:TColor); function Cells_GetColor(Row:Word;Col:Byte):TColor; procedure Cells_SetColorId(Row:Word;Col:Byte;ColorIndex:Byte); function Cells_GetColorId(Row:Word;Col:Byte):Byte; procedure Cells_SetFontName(Row: Word; Col:Byte; xFontName:String); function Cells_GetFontName(Row:Word;Col:Byte):String; procedure Cells_SetFontSize(Row: Word; Col:Byte; xFontSize:Byte); function Cells_GetFontSize(Row:Word;Col:Byte):Byte; procedure Cells_SetFontColor(Row: Word; Col:Byte; xFontColor:TColor); function Cells_GetFontColor(Row:Word;Col:Byte):TColor; procedure Cells_SetFontBold(Row: Word; Col:Byte; vol:Boolean); procedure Cells_SetFontItalic(Row: Word; Col:Byte; vol:Boolean); procedure Cells_Border(Row:Word; Col:Byte; xType:byte; xColor:TColor); procedure Cells_Pattern(Row:Word; Col:byte; xType:byte);</pre>	<p>Возвращает цвет ячейки Закрашивает ячейку цветом по индексу</p> <p>Возвращает индекс цвета ячейки Устанавливает имя шрифта для ячейки Возвращает имя шрифта ячейки Устанавливает размер шрифта ячейки Возвращает размер шрифта ячейки Устанавливает цвет шрифта ячейки Возвращает цвет шрифта ячейки Устанавливает полужирный шрифт Устанавливает наклонный шрифт Определяет вид контура ячейки Заливает ячейку узором</p>
--	---

Таблица 3

Методы управления свойством Range	Выполняемое действие
<pre>procedure Range_Select(xRange:String); procedure Range_Clear(xRange: String); procedure Range_Copy(xRange: String);</pre>	<p>Выбирает диапазон ячеек (ячейку) Очищает диапазон ячеек (ячейку) Копирует значения диапазона ячеек (ячейки)</p>
<pre>procedure Range_Paste(xRange: String);</pre>	<p>Вставляет значения диапазона ячеек (ячейки)</p>
<pre>procedure Range_Write(xRange: string; Data:Variant); function Range_Read(xRange: string):Variant;</pre>	<p>Записывает значения в диапазон ячеек (ячейку) Считывает значения из диапазона ячеек (ячейки)</p>
<pre>procedure Range_FormulaWrite(xRange:String; xFormula:String); function Range_FormulaRead(xRange: string):String;</pre>	<p>Записывает формулу в диапазон ячеек (ячейку) Считывает формулы из диапазона ячеек (ячейки)</p>
<pre>procedure Range_SetName(xRange:String; xName:String);</pre>	<p>Присваивает имя диапазону ячеек (ячейке)</p>

<pre> procedure Range_SetColor(xRange:String; xColor:TColor); function Range_GetColor(xRange:String):TColor; procedure Range_SetColorId(xRange:String;ColorIndex:Byte); function Range_GetColorId(xRange:String):Byte; procedure Range_SetFontName(xRange:String; xFontName:String); function Range_GetFontName(xRange:String):String; procedure Range_SetFontSize(xRange:String; xFontSize:Byte); function Range_GetFontSize(xRange:String):Byte; procedure Range_SetFontColor(xRange:String; xFontColor:TColor); function Range_GetFontColor(xRange:String):TColor; procedure Range_SetFontBold(xRange:String; vol:Boolean); procedure Range_SetFontItalic(xRange:String; vol:Boolean); procedure Range_SetWidthCells(xRange:String;xWidth:Single); function Range_GetWidthCells(xRange:String):Single; procedure Range_Alignment(xRange:String;Align:Byte); procedure Range_MergeCells(xRange:String); procedure Range_Border(xRange:String; xType:byte; xColor:TColor); procedure Range_Pattern(xRange:String; xType:byte); </pre>	<p>Закрашивает диапазон ячеек (ячейку)</p> <p>Возвращает цвет диапазона ячеек (ячейки)</p> <p>Закрашивает диапазон ячеек (ячейку) по индексу</p> <p>Возвращает индекс цвета диапазона ячеек (ячейки)</p> <p>Устанавливает имя шрифта для диапазона ячеек (ячейки)</p> <p>Возвращает имя шрифта для диапазона ячеек (ячейки)</p> <p>Устанавливает размер шрифта для диапазона ячеек (ячейки)</p> <p>Возвращает размер шрифта для диапазона ячеек (ячейки)</p> <p>Устанавливает цвет шрифта для диапазона ячеек (ячейки)</p> <p>Возвращает цвет шрифта для диапазона ячеек (ячейки)</p> <p>Устанавливает полужирный шрифт для диапазона ячеек (ячейки)</p> <p>Устанавливает наклонный шрифт для диапазона ячеек (ячейки)</p> <p>Устанавливает ширину для диапазона ячеек (ячейки)</p> <p>Возвращает ширину для диапазона ячеек (ячейки)</p> <p>Определяет расположение информации для диапазона ячеек (ячейки)</p> <p>Объединяет диапазон ячеек в одну ячейку</p> <p>Определяет вид обрамления диапазона ячеек (ячейки)</p> <p>Заливает узором диапазон ячеек (ячейку)</p>
--	--

Пример распределенного вычисления

Пусть необходимо вычислить значение: $y = \sin^2 x + \cos^2 x$, причем $P_1 = \sin^2 x$ и $P_2 = \cos^2 x$ вычисляются в клиент-приложении, а сумма значений $y = P_1 + P_2$ в сервере *Excel*. Результат вычислений должен быть передан в приложение клиент и

выведен в заголовок формы. Тогда протокол обмена данными между приложением клиент и сервером будет иметь следующий вид:

```
procedure TForm1.Button1Click(Sender: TObject);  
var X, P1,P2,Y : Double;  
begin  
  X := 0.5; P1 := SQR(Sin(0.5)); P2 := SQR(Cos(0.5));           // определение X и  
вычисление P1 и P2  
  ExCtrl1.Excel_Active;                                       // запуск Excel  
  ExCtrl1.Range_Write('A1',P1); ExCtrl1.Range_Write('B1',P2); // запись значений  
P1 и P2 в ячейки A1 и B1  
  ExCtrl1.Range_FormulaWrite('C1', '= A1+B1');               // запись формулы  
вычисления суммы в ячейку C1  
  Y := ExCtrl1.Range_Read('C1');                               // считывание  
результата из ячейки C1  
  Form1.Caption := 'Result = ' + FloatToStr(Y);              // вывод результата  
в заголовок формы  
end;
```

Примечание: В форму установлен компонент **ExCtrl** и кнопка **Button**. Процедура, обеспечивающая протокол обмена реализована в обработчике событий **onClick** кнопки.

Лабораторная работа №5

Тема: ПРОСТОЙ ЧАТ ДЛЯ ЛОКАЛЬНОЙ СЕТИ

Цель работы: Изучение различных методов и приемов работы с сокетами (компонентами Server Socket и Client Socket) на простом наглядном примере приложения - чата для локальной сети. В этом примере мы будем использовать два приложения - чат-сервер и чат-клиент. Обратите внимание на тот факт, что чат-клиенты подключаются к чат-серверу и через него обмениваются сообщениями. Чат-сервер может быть запущен как на отдельном, так и на том компьютере, где запущен один из клиентов. Кроме того, для тестирования Вы можете запустить на своем компьютере чат-сервер и несколько чат-клиентов. Для этого необходимо будет указать значение localhost в поле Host, а в поле Port у сервера и у клиентов должны быть одинаковые значения номеров портов.

Подготовка к выполнению работы

Для выполнения данной работы необходимо подключить к Delphi библиотеку dclsockets.bpl, находящуюся в папке Program Files / Borland / Delphi 7 / Bin / ... Подключение выполняется командой Install Packages меню Component, которая вызывает диалоговое окно Default Project Options (рис. 1). В указанном

окне выполните команду ADD и папке Bin выберите файл dclsockets.bpl, и затем нажмите на кнопку ОК. Компоненты Client Socket и Server Socket появятся в закладке Internet палитры компонентов Delphi.

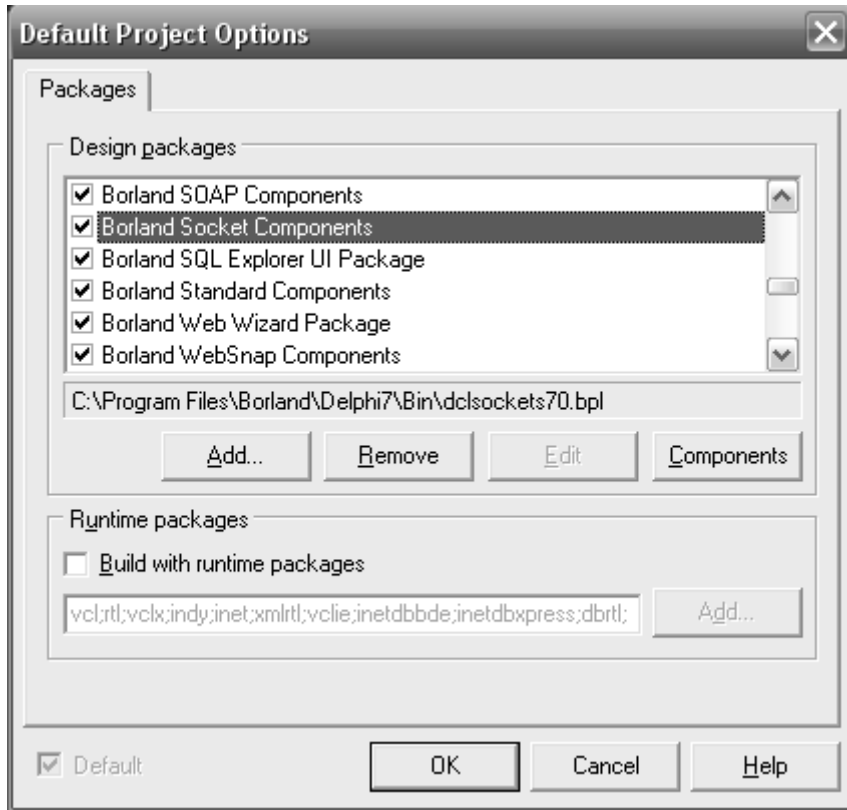


Рис. 1 Диалоговое окно *Default Project Options*

Дизайн приложения Сервер

Установите в форму следующие компоненты (Рис. 2):

Panel1, свойство Align := alTop;

Listbox1, свойство Align := alClient;

Кнопки: Button1 и Button1 (устанавливаются в Panel1);

ServerSocket1;

Измените номера и заголовки компонентов (Рис. 3):

Form1 – Name: SvrForm; Caption: Сервер;

Button1 – Name: StartBtn, Caption: Старт;

Button2 – Name: StopBtn, Caption: Стоп;

Panel1 – Caption: (удалить, пусто).

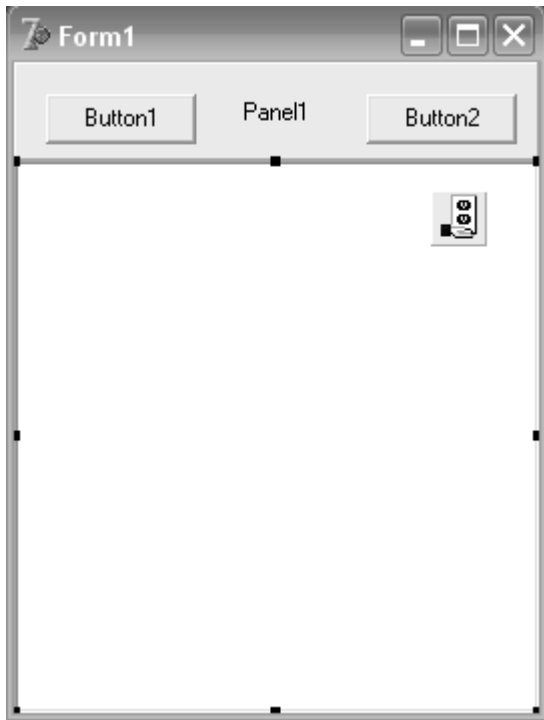


Рис 2. Установка компонентов

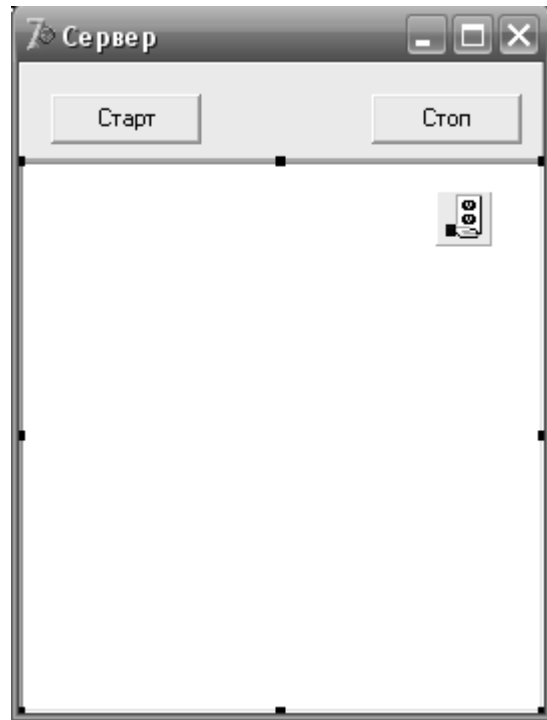


Рис 3. Именованние компонентов

Сохраните проект в папке Сервер с именами: unit1 -> Svr; Project1 -> Server.

Разработка программы Сервер

Для запуска и останова работы сервера напишите два обработчика событий щелчка кнопок `OnClick`.

Для кнопки `StartBtnClick`:

```

procedure TSVRForm.StartBtnClick(Sender: TObject);
var
  s: string;
begin
  {Запрашиваем порт}
  s := InputBox('Start chat server','Enter port:','1001');
  if s = '' then Exit;
  {Очищаем список пользователей - ListBox1}
  ListBox1.Items.Clear;
  {Устанавливаем порт}
  ServerSocket1.Port := StrToInt(s);
  {Запускаем сервер}
  ServerSocket1.Open;
end;
  
```

Для кнопки `StopBtnClick`:

```

procedure TSVRForm.StopBtnClick(Sender: TObject);
begin
  {Очищаем список пользователей и останавливаем сервер}
  ListBox1.Items.Clear;
  if ServerSocket1.Active then ServerSocket1.Close;
end;

```

Скомпилируйте приложение.

В инспекторе объектов для компонента ServerSocket1 (закладка Events) создайте обработчик события onClientRead, выполнив двойной щелчок мышью по полю справа от названия обработчика события.

Запишите в созданный метод следующий текст программы:

```

procedure TSVRForm.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  s: string;
  i: Integer;
begin
  {сохраняем в s присланную строку}
  s := Socket.ReceiveText;
  {Если какой либо пользователь прислал нам свое имя, то:}
  if Copy(s,1,2) = '#N' then
  begin
    Delete(s,1,2); // удаляем первые два символа
    {Добавляем пользователя в список пользователей}
    ListBox1.Items.Add(s);
    {Записываем в s команду для послыки 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 какой либо из пользователей отослал сообщение,
   то 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;

```

Скомпилируйте приложение.

Пояснение:

Первая строка сохраняет полученные из сокета данные в переменной s. Далее, функция Copy(s,1,2) возвращает первые два символа строки s, которые затем проверяются на соответствие с условно-введенной нами командой "#N",

которая означает, что в строке *s* после самой команды содержится присланное кем-либо из клиентов имя (ник - псевдоним). Это имя затем добавляется в *ListBox1*. Таким образом, *ListBox1* становится списком подключенных клиентов. Далее, в строку *s* (информация в которой уже не нужна) записываем команду "#U" и последовательно всех пользователей из списка *ListBox1*. Затем всю эту строку рассылаем ВСЕМ клиентам. Далее, если мы получили не "#N", а "#M" или "#P" (простое или приватное сообщение) - рассылаем его всем клиентам так как они сами разберутся, кому прислано сообщение.

Аналогично, создайте обработчик события `onClientDisconnect` и напишите в нем следующий текст программы:

```

procedure TSVRForm.ServerSocket1ClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
var
  i: Integer;
begin
  {Если подключился или отключился какой либо пользователь,
  то 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;

```

Скомпилируйте приложение.

Пояснение:

ServerSocket1ClientDisconnect - обработчик события, который возникает в том случае, когда кто-либо из клиентов отсоединился от сервера. Здесь мы очищаем список пользователей и посылаем всем клиентам запросы на получение их псевдонимов.

В заключение разработки свяжите обработчики событий `OnAccept` и `OnClientDisconnect`, выбрав в поле `OnAccept` значение `ServerSocket1ClientDisconnect` (рис. 4).

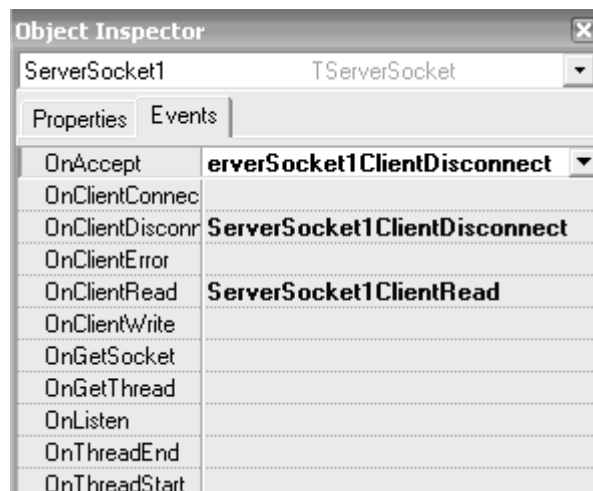


Рис. 4. Связывание обработчиков событий

Запустите приложение и проверьте его функциональность.

Примечание: Данный пример максимально упрощен, чтобы просто была понятна технология создания подобных приложений. Возможности реального чата должны быть намного шире. Также имейте в виду, что команды типа "#N", "#U", "#M", и т.д. вводятся самим разработчиком для того, чтобы определить, какую информацию прислали из сокета. Эти команды не относятся к командам сокетов, а являются пользовательскими командами .

Дизайн приложения Клиент

Создайте папку с названием Клиент, в которой будет сохранен проект приложения.

Создайте в Delphi новый проект.

Установите в форму следующие компоненты и измените их свойства (рис 5):

Панель Panel1: Caption -> "(пусто), Align -> alTop.

В панель установите:

кнопку SpinButton1: свойство Name -> ConnectBtn, свойство Caption -> Connect, свойство AllowAllUp – true, свойство GroupIndex -> 1.

3 компонента класса TEdit.

Для компонента Edit1 свойство Name -> HostEdit, Text -> localhost.

Для компонента Edit2 свойство Name -> PortEdit, Text -> 1001.

Для компонента Edit3 свойство Name -> NickEdit, Text -> Введите псевдоним.

4 компонента класса TLabel. Свойства: Caption -> (Host:, Port:, Nick: и Список участников) соответственно.

Панель Panel2: Caption -> "(пусто), Align -> alBottom.

В панель установите:

Компонент класса TEdit, свойство Name -> SentEdit, свойство Text -> " (пусто)

Компонент класса TLabel, свойство Caption -> Текст сообщения:

Компонент CheckBox1, свойство Caption -> Личное сообщение

Компонент Button1 свойство Name -> SendBtn, свойство Caption -> Отправить.

Компонент ListBox1, свойство Align -> alRight.

Компонент Memo1, свойство Align -> alClient, свойство ScrollBars -> ssBoth

Компонент ClientSocket1.

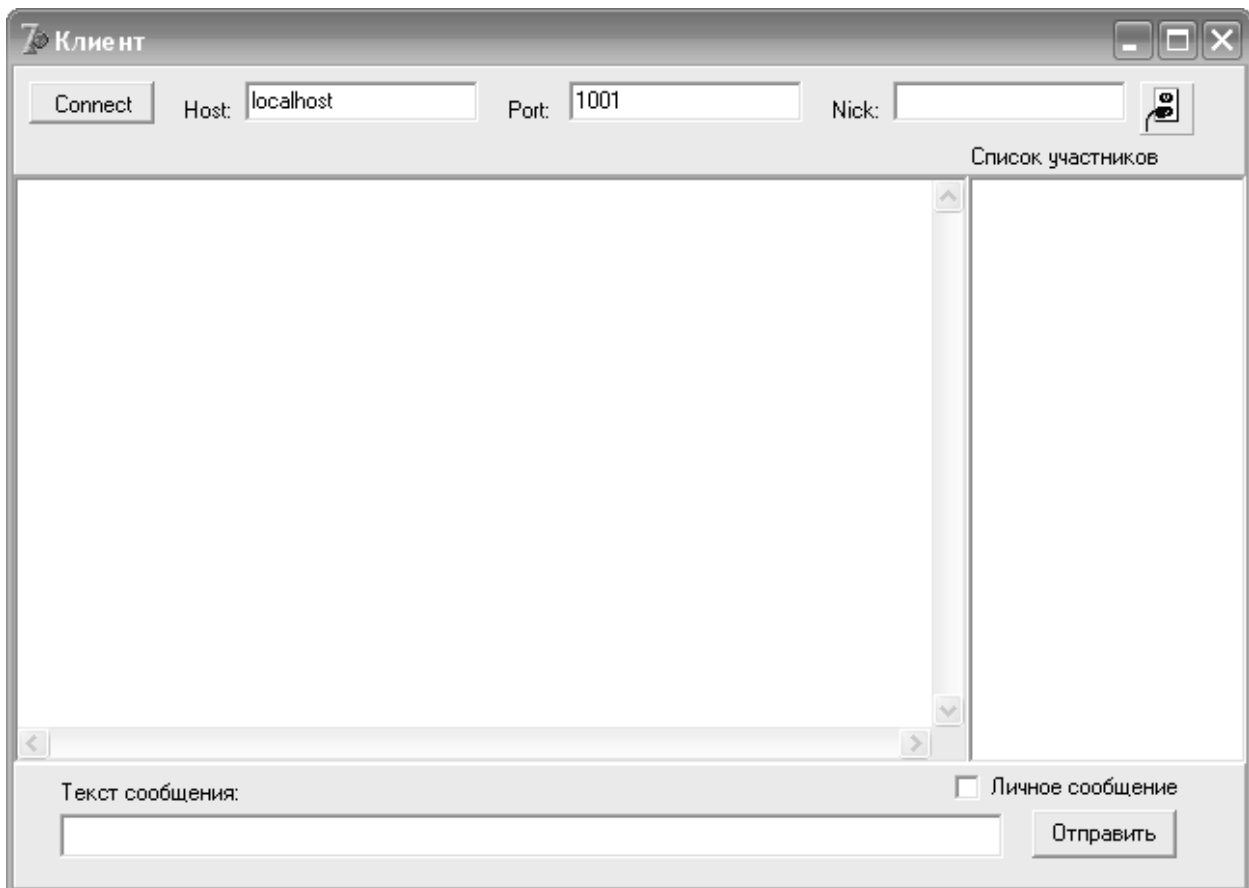


Рис. 5 Общий вид формы приложения Client

Разработка программы Клиент

Согласно логике работы приложения соединение с сервером должно осуществляться при нажатой кнопке Connect. При этом все поля: Host, Port и Nick должны быть заполнены.

Соединение выполним в обработчике события щелчка кнопки onConnectBtnClick:

```

procedure TForm1.ConnectBtnClick(Sender: TObject);
begin
    if ConnectBtn.Down then // Если нажата кнопка
    begin
        nickname := NickEdit.Text; // определяем псевдоним
        if ClientSocket1.Active then ClientSocket1.Close;
        {Устанавливаем свойства Host и Port}
        ClientSocket1.Host := HostEdit.Text; //host;
        ClientSocket1.Port := StrToInt(PortEdit.Text); // port
        ClientSocket1.Open; //устанавливаем соединение
    end else // в противном случае
    begin
        ClientSocket1.Close; // разрываем соединение
    end;
end;

```

Скомпилируйте приложение.

Вывод кода ошибки соединения опишем в обработчике события onError компонента ClientSocket1.

```
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;
```

Поиск host'a выполним в обработчике события onLookup компонента ClientSocket1.

```
procedure TForm1.ClientSocket1Lookup(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Сообщаем о том, что идет поиск хоста}
  Memo1.Lines.Insert(0, 'Looking up for server...');
end;
```

Результат соединения определим в обработчике события onConnecting компонента ClientSocket1.

```
procedure TForm1.ClientSocket1Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {соединяемся...}
  Memo1.Lines.Insert(0, 'connecting...');
end;
```

Вывод результата соединения опишем в обработчике события onConnect компонента ClientSocket1.

```
procedure TForm1.ClientSocket1Connect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {установили соединение}
  Memo1.Lines.Insert(0, 'connected!');
end;
```

Разрыв соединения опишем в обработчике события onDisconnect компонента ClientSocket1.

```
procedure TForm1.ClientSocket1Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {разорвали соединение}
  Memo1.Lines.Insert(0, 'disconnected');
end;
```

Скомпилируйте приложение

Логика обмена информацией пользователей опишем в обработчике события onRead onDisconnect компонента ClientSocket1.

```
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);
      {Очищаем ListBox1}
      ListBox1.Items.Clear;
      {Добавляем по одному юзеру 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;
```

{!!! Продолжение текста метода см. далее !!!}

```
  {Если нам прислали запрос на наш псевдоним}
  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;
```

Скомпилируйте приложение.

Отправку сообщения опишем в обработчике события щелчка кнопки “Отправить” onSentBtnClick.

```
procedure TForm1.SendBtnClick(Sender: TObject);
var s: string;
begin
  {Если мы хотим послать приватное сообщение, но не выбрали адресата -
  нас покажут замечанием :) и выгонят из обработчика}
  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
    {добавляем спец.команду и адресат}
    s := '#P'+ ListBox1.Items[ListBox1.ItemIndex] + ';'
  else {А если не очень приватное?}
    s := '#M'; {Просто спец.команду}
  {Добавляем наше имя (от кого) и само сообщение}
  s := s + nickname + ';' + SentEdit.Text;
  {Посылаем все это по сокету}
  ClientSocket1.Socket.SendText(s);
  {И снова ждем ввода в уже чистом TEdit-e}
  SentEdit.Text := '';
  ActiveControl := SentEdit;
end;
```

Скомпилируйте приложение.

Пояснение:

Разберем подробнее обработчик события OnRead - ClientSocket1Read. Сначала мы сохраняем полученные по сокету данные в строку s. Затем, если нам прислали список других подключенных клиентов, то мы выделяем из строки s по одному пользователю и добавляем их последовательно в ListBox1. Таким образом, ListBox1 становится списком всех пользователей. Далее - если нам прислали команду "#M" - обычное сообщение, то мы выделяем из s отправителя и само сообщение, а затем все это в стандартной для чатов форме выводим в Memo1. Если же был получен запрос на имя пользователя - команда "#N", то посылаем серверу свой псевдоним. Если пришло приватное сообщение ("#P"), то из строки s мы выделяем имя отправителя, адресата и само сообщение. Если сообщение адресовано нам, либо мы же его и отправили - выводим его в Memo1.

Теперь разберем обработчик нажатия кнопки отправки сообщения SentBtnClick. Если поставлен флаг CheckBox1 (т.е. сообщение - приватное), а адресат в списке пользователей не выделен - выдаем ошибку. Далее, формируем в s команду для отправки сообщения: сначала тип ("#P" или "#M". Если "#P", то плюс имя адресата из списка юзеров), разделитель (";"), затем - имя отправителя, разделитель, и непосредственно сам текст сообщения. Далее идет отправка строки s по сокету, очистка поля ввода сообщения, и перевод в него фокуса (курсора).

В заключении определим отправку сообщения при нажатии клавиши Enter. Для этого создадим обработчик события onKeyDown для компонента SentEdit (поле ввода Текст сообщения) и напишем следующий текст программы:

```
procedure TForm1.SentEditKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  { Отправка сообщения клавишей Enter }
  if Key = VK_RETURN then
    SendBtn.Click;
end;
```

Скомпилируйте приложение.

Проверка функциональности системы Клиент – Сервер.

Для проверки функциональности приложения Клиент необходимо запустить сервер из Вашей рабочей папки и нажать кнопку Старт. В диалоговом окне оставить без изменения предложенный номер порта. Далее из среды Delphi запустить приложение Клиент. В поле Nick Необходимо ввести Ваш псевдоним и нажать кнопку Connect (Значения полей Host и Port оставьте без изменения). При правильной работе приложений в списке сервера должен появиться Ваш псевдоним, а в окне сообщений клиента логин соединения (рис.6).

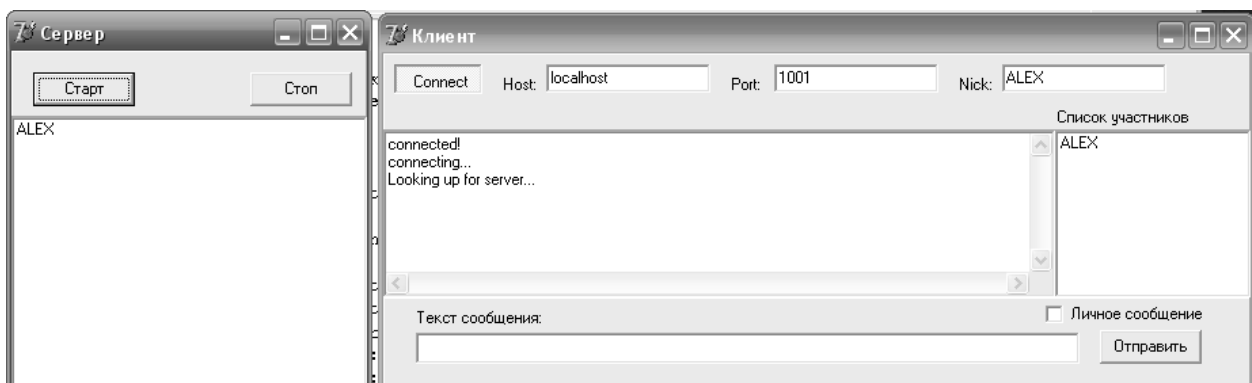


Рис. 6 Взаимодействие приложений Клиент - Сервер

Далее введите в поле ввода “Текст сообщения” Ваше сообщение и отправьте его командной кнопкой отправить. Введите другое сообщение и отправьте его нажатием на клавишу Enter. В окне обмена приложения Клиент Вы должны получить текст, подобный приведенному на рисунке 7.

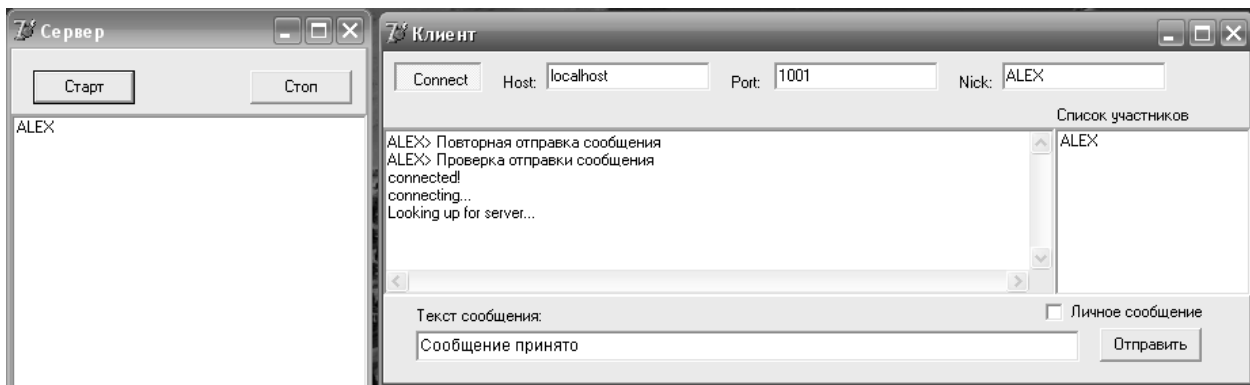


Рис. 7 Тест передачи сообщений

Комплексная проверка работы

- Выйдете из среды Delphi.
- Запустите из рабочей папки Сервер приложение Сервер и активируйте его путем нажатия кнопки Старт.
- Запустите из рабочей папки Клиент три копии приложения клиент. Укажите различные псевдонимы для каждой копии и установите соединение с сервером (рис. 8).

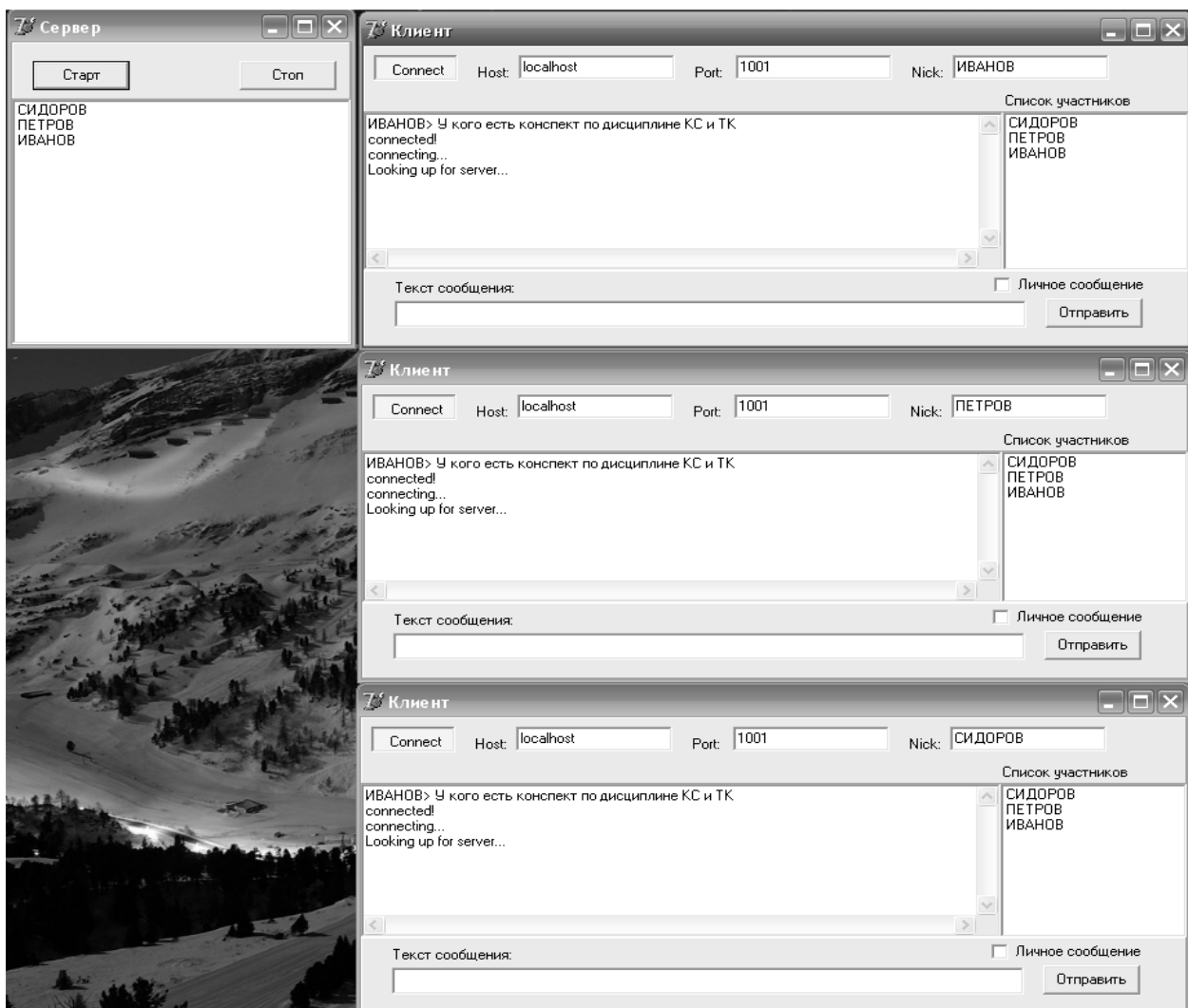


Рис. 8 Чат трех пользователей

- Отошлите всем клиентам из первого клиента некоторое публичное сообщение, например “У кого есть конспект по дисциплине КС и ТК”.
- Создайте ответы из клиентов 2 и 3 клиенту 1. Например, клиент 2 отвечает, что у него нет конспекта, а клиент 3 отвечает положительно.

Проверьте частную беседу клиентов, выбрав опции личное сообщение, например: для того чтобы Иванов послал частное сообщение Петрову (рис. 9) достаточно выбрать опцию “Личное сообщение” и перед его отправкой дважды щелкнуть мышкой в списке участников по псевдону Петров.

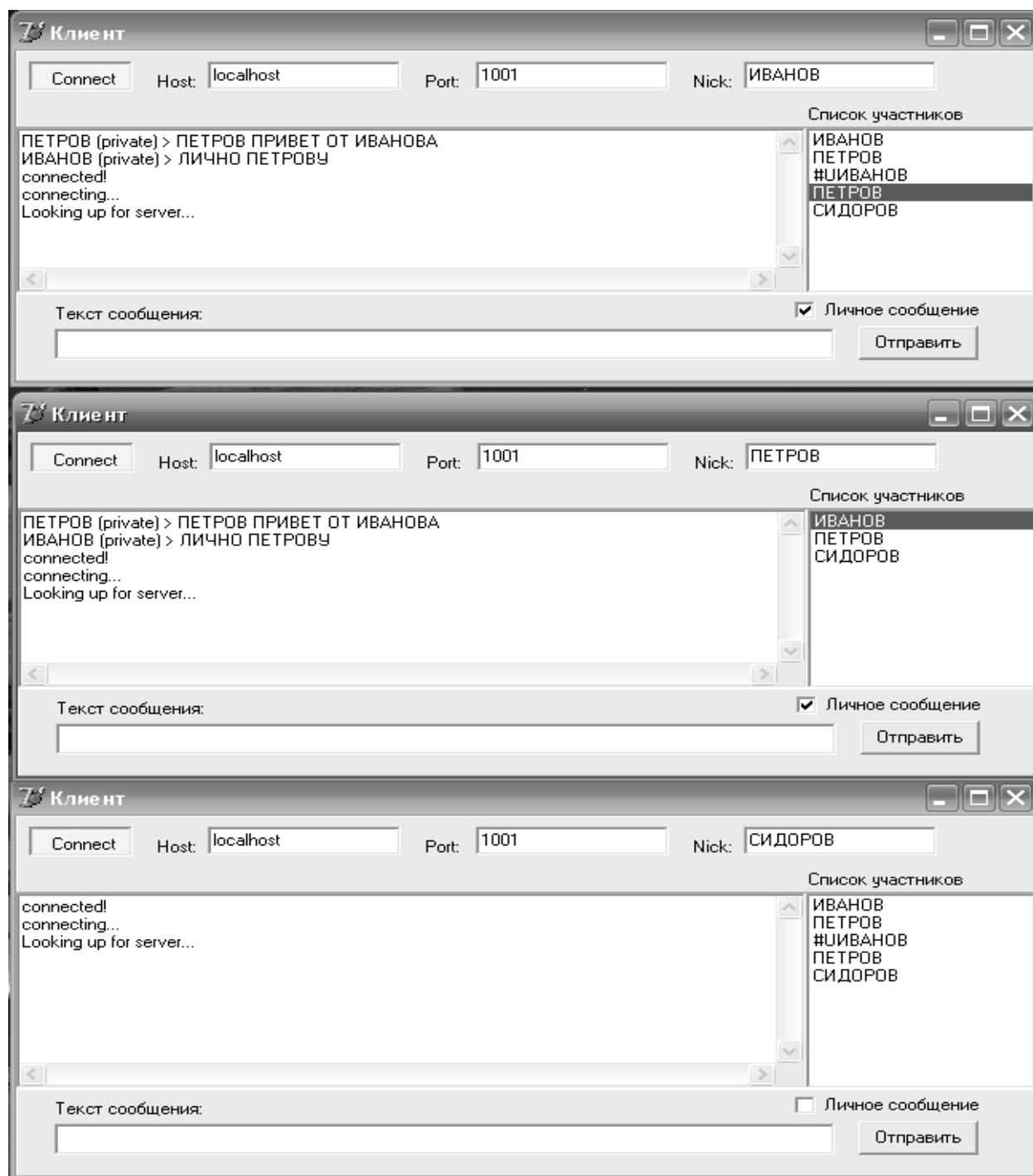


Рис. 9 Обмен частными сообщениями