МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

«СОГЛАСОВАНО»	«УТВЕРЖДАЮ»
Руководитель ОП	Заведующая (ий) кафедрой
д.ф.-м.н., профессор, академик РАН, Гузев М.А.	информатики, математического и компьютерного моделирования
	(название кафедры)
_____	_____ Чеботарев А.Ю.
(подпись) (Ф.И.О. рук. ОП)	(подпись) (Ф.И.О. зав. каф.)
«23» июня 2017 г.	«23» июня 2017 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ (РПУД)
Сетевые технологии и системное администрирование

Направление подготовки 09.03.03Прикладная информатика

Форма подготовки очная

курс 3 семестр 5,6
лекции 36,18 час.
практические занятия36, 72 час.
лабораторные работы _____час.
в том числе с использованием МАО лек._____/пр._____/лаб._____ час.
всего часов аудиторной нагрузки 72, 90 час.
в том числе с использованием МАО _____ час.
самостоятельная работа 72, 27 час.
в том числе на подготовку к экзамену 27 час.
контрольные работы (количество)
курсовая работа / курсовой проект _____ семестр
зачет 5семестр
экзамен 6 семестр

Рабочая программа составлена в соответствии с требованиями образовательного стандарта, самостоятельно установленного ДВФУ, принятого решением Ученого совета Дальневосточного федерального университета, протокол от 28.01.2016 № 01-16, и введенного в действие приказом ректора ДВФУ от 18.02.2016 № 12-13-235.

Рабочая программа обсуждена на заседании кафедры информатики, математического и компьютерного моделирования, протокол №22 «23» июня 2017 г.

Заведующий кафедрой Информатики, математического
и компьютерного моделирования, д.ф.-м.н., профессор Чеботарёв А.Ю.
Составитель ст.преподаватель кафедры Жандармова И.В.

Оборотная сторона титульного листа РПУД

I. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____
(подпись) (И.О. Фамилия)

II. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____
(подпись) (И.О. Фамилия)

Аннотация

Рабочая программа дисциплины«Сетевые технологии и системное администрирование»разработана для студентов 3-го курса по направлению 09.03.03 «Прикладная информатика» для профиля «Прикладная информатика в компьютерном дизайне» в соответствии с требованиями федерального государственного образовательного стандарта высшего образования, утвержденного приказом Министерства образования и науки РФ от 17 февраля 2014 г. № 124.

Общая трудоемкость освоения дисциплины составляет 8 зачетные единицы. Учебным планом предусмотрены лекционные занятия, практические занятия и самостоятельная работа студента. Дисциплина реализуется на 3 курсе в 5, 6 семестрах.

Содержание дисциплины «Сетевые технологии и системное администрирование» связано с дисциплинами «Основы информатики и программирования», «Основы современных информационных технологий» учебного плана по направлению 09.03.03 «Прикладная информатика».

Целью освоения дисциплины являются формирование понятий о построении сетей на основе принципов открытости, о функциях, реализуемых на каждом уровне семиуровневой модели взаимодействия открытых систем OSI, о принципах передачи данных на физическом уровне, методах доступа к единой среде передачи данных, способах сжатия данных, о технологиях локальных сетей Ethernet, TokenRing, FDDI, о глобальных сетях с коммутацией каналов и коммутацией пакетов, о разработке сетевых приложений наязыке высокого уровня.

Задачи:

1. овладеть системой знаний по информатике и её технологиям,
2. приобрести навык выбора информационных технологий для решения конкретной задачи,

3. исходя из особенностей информации, оптимизировать её обработку,

4. понимать влияние компьютера на эффективность выполнения программ, а также понимать особенности выполнения программ на компьютере в зависимости от реализации языка,

Для успешного изучения дисциплины ««Вычислительные системы, сети и телекоммуникации» у обучающихся должны быть сформированы следующие предварительные компетенции:

5. умение строить алгоритмы и программировать

В результате изучения данной дисциплины у обучающихся формируются следующие части общих профессиональных компетенций:

Код и формулировка компетенции	**Этапы формирования компетенции**	
ПК-28 Способностью готовить обзоры научной литературы и электронных информационно-образовательных ресурсов для профессиональной деятельности	Знает	как применять к решению прикладных задач базовые алгоритмы обработки информации
	Умеет	выполнять оценку сложности алгоритмов
	Владеет	программированием и тестированием программ
ПК-17 Способностью осуществлять презентацию информационной системы и начальное обучение пользователей	Знает	Как и где извлекать полезную научно-техническую информацию
	Умеет	извлекать полезную научно-техническую информацию из электронных библиотек, реферативных журналов, сети Интернет
	Владеет	способами сортировки и быстрого поиска
ПК-15 аналитическая деятельность: способен проводить оценку экономических затрат на проекты по информатизации и автоматизации решения прикладных задач	Знает	Как проводить оценку экономических затрат на проекты
	Умеет	проводить оценку экономических затрат на проекты по информатизации и автоматизации решения прикладных задач
	Владеет	некоторыми способами аналитической деятельности

I. СТРУКТУРА И СОДЕРЖАНИЕ Теоретической ЧАСТИ КУРСА

МОДУЛЬ 1. Основы теории вычислительных сетей

Раздел I. Концепция архитектуры открытых систем (8 час.)

Тема 1. Базовые топологии сетей (4 час.)

История развития сетей. Системы пакетной обработки (50-е годы). Многотерминальные системы (60-е годы). Появление глобальных сетей (60-е годы – начало 70-х годов). Первые локальные сети (70-е годы). Создание стандартных технологий локальных сетей (80-е годы). Современные тенденции. Преимущества использования сетей. Классификация сетей. Глобальные сети. Региональные сети. Локальные сети. Характеристики сетей. Базовые топологии сетей. Шина. Звезда. Кольцо.

Тема 2. Концепция архитектуры открытых систем (4 час.)

Понятие «открытая система». Модель OSI. Семиуровневая сетевая архитектура. Уровни и протоколы. Характеристика уровней семиуровневой модели. Сетезависимые и сетенезависимые уровни.

Раздел II. Физический и канальный уровни модели OSI(12 час).

Тема 3. Методы передачи данных на физическом уровне (4 час.)

Физическое кодирование. Потенциальный код без возвращения к нулю. Метод биполярного кодирования с альтернативной инверсией. Потенциальный код с инверсией при единице. Биполярный импульсный код. Манчестерский код. Потенциальный код 2B1Q. Логическое кодирование. Избыточные коды. Логический код 4B/5B. Скремблирование.

Тема 4. Методы доступа к среде передачи данных (4 час.)

Классификация методов доступа. Схемы с состязаниями ALOHA. Метод доступа CSMA/CD. Схемы с резервированием. Метод циклического опроса. Гибридные схемы. Схемы с маркерами. Маркерная схема с приоритетами. Области применения сетей с различными методами доступа.

Тема 5. Способы контроля правильности передачи информации (4 час.)

Обнаружение и коррекция ошибок. Методы обнаружения ошибок. Контроль по паритету. Вертикальный и горизонтальный контроль по паритету. Циклический избыточный контроль. Методы восстановления искажённых и потерянных кадров. Метод с простоями. Метод «скользящего окна». Сжатие данных. Понятие энтропии данных. Методы сжатия данных. Адаптивное сжатие. Десятичная упаковка. Относительное кодирование. Символьное подавление. Коды переменной длины.

МОДУЛЬ 2. Локальные сети

Раздел I. Стандарты Ethernet (12 час).

Тема 6. Стандарты IEEEна 10 Мбит/с (2 час.)

Архитектура локальных сетей. Технология Ethernet. Формат кадра Ethernet. Спецификация 802.3. Стандарт 10 BASET. Стандарт 10 BASE 2. «Тонкий» Ethernet. Правило 5-4-3. Стандарт 10 BASE 5. «Толстый» Ethernet.Комбинирование «толстого» и «тонкого» Ethernet. Стандарт 10 BASEFL.

Тема 7. Стандарты IEEEна 100 Мбит/с (4 час.)

Технология FastEthernet. Аппаратура 100BASE-TX. Сдвоенная витая пара.Типы сред передачи. Аппаратура 100BASE-TX. Аппаратура 100BASE-

Т4. Аппаратура 100BASE-FX. Выбор конфигурации FastEthernet. Зона конфликта.

Тема 8. GigabitEthernet (2 час.)

GigabitEthernet в качестве магистрали**.**Типы сред передачи. Время двойного оборота. Максимальная производительность сети Ethernet. Кадры минимальной и максимальной длины. Полезная пропускная способность протокола. Коэффициент использования сети.

Тема 9. Сети TokenRing и FDDI (4 час.)

Форматы кадров. Маркер. Кадр данных. Прерывающая последовательность. Поля маркера. Поля кадра данных и прерывающей последовательности. Особенности сетей FDDI.

Раздел II. Аппаратура локальных сетей(16час).

Тема 10. Характеристики проводных линий связи для локальных сетей(4 час.)

Электрические кабели с витыми парами сетей Ethernet и FastEthernet. Коаксиальные кабели. Волоконно-оптический кабель.

Тема 11. Аппаратура локальных сетей(4 час.)

Сетевые адаптеры. Функции и характеристики сетевых адаптеров. Авточувствительность. Классификация сетевых адаптеров. Поколения сетевых адаптеров. Концентраторы. Конструктивное исполнение концентраторов.

Тема 12. Логическая структуризация сети(8 час.)

Ограничения сети, построенной на общей разделяемой среде. Преимущества логической структуризации сети. Структуризация с помощью мостов и коммутаторов. Алгоритм работы прозрачного моста. Затопление сети. Широковещательный шторм. Мосты с маршрутизацией от источника. Ограничения топологии сетей, построенной на мостах. Алгоритм SpanningTree. Коммутаторы локальных сетей. Ограничения мостов и коммутаторов. Принципы объединения сетей на основе протоколов сетевого уровня.

МОДУЛЬ 3. Глобальные сети и телекоммуникации

Раздел I. Управление обменом информацией в глобальных сетях (8 час.)

Тема 13. Коммутация каналов (4 час).

Частотное мультиплексирование. FDM- коммутатор. Мультиплексирование с разделением времени.КоммутаторTDM. Общие свойства сетей с коммутацией каналов. Особенности построения сетей на основе коммутации пакетов.

Тема 14. Структура глобальной сети(4 час).

Транспортные функции глобальной сети. Высокоуровневые услуги глобальных сетей. Пример структуры глобальной сети. Типы глобальных сетей.

Раздел II. Глобальные сети на основе выделенных линий (14 час).

Тема 15. Технологии плезиохронной цифровой иерархии PDH (4 час).

Аналоговые выделенные линии. Новые выделенные линии. Технологии PDH. Иерархия скоростей. Аппаратура T1, T2, T3. Каналы типа E1, E2, E3.

Тема 16. Технология синхронной цифровой иерархии(4 час).

Технология SONET/SDH. Терминальные устройства. Мультиплексоры. Мультиплексора «ввода-вывода». Цифровые кросс-коннекторы. Регенераторы сигналов. Стек протоколов технологии SONET/SDH. Физический уровень (фотонный). Уровень секции. Уровень линии. Уровень тракта. Отказоустойчивость сети SONET/SDH. Управление, конфигурирование и администрирование сети SONET/SDH. Применение цифровых первичных сетей.

Тема 17. Программное обеспечение сетей (6 час).

Структура стека TCP/IP. Характеристика протоколов. Адресация в IPсетях. Три основных класса IP– адресов. Использование масок в IP-адресации. Отображение физических адресов на IP- адреса: протокол ARP. Автоматизация процесса назначения IP- адресов узлами сети- протокол DHCP. Протокол IP. Формат пакета IP. Маршрутизация. Виды и алгоритмы маршрутизации. Алгоритм поиска маршрута в таблице маршрутизации. Протокол динамической маршрутизации RIP.

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Лабораторные работы (36 час.)

Занятие 1. Исследование функциональных характеристик локальной вычислительной сети (2 час.)

1. Проверка сетевых соединений, определение сетевых установок компьютера, передача сообщения по сети.
2. Получение ARP – таблицы.
3. Просмотр статистики работы протоколов Ethernet и TCP.

Занятие 2. Работа с почтовыми ящиками Windows (2 час.)

1. Создание серверного приложения с использованием почтовых ящиков Windows.
2. Создание клиентского приложения с использованием почтовых ящиков Windows.
3. Передача сообщения

Занятие 3. Исследование возможностей сети при выполнении операций коллективного доступа к сетевым ресурсам (2 час.)

1. Разработка приложения, создающего файл по UNC-соединению. Создание файла, установка атрибутов совместного доступа.
2. Исследование возможности для записи в файл какой-либо текстовой строки.
3. Исследование возможности считывания содержимого файла на своём ПК и на сервере.

Занятие 4. Работа с именованными каналами WINDOWS в блокирующем режиме (2 час.)

1. Создание серверного приложения, работающего в блокирующем режиме.
2. Создание клиентского приложения, работающего в блокирующем режиме.
3. Расширение возможностей именованных каналов для передачи пакетов различной длины.

Занятие 5. Работа с именованными каналами WINDOWSв неблокирующем режиме (2 час.)

1. Создание серверного приложения, работающего в неблокирующем режиме.
2. Создание клиентского приложения,работающего в неблокирующем режиме.
3. Исследование возможностей разработанных приложений по отправке сообщений.

Занятие 6. Поиск доступных сетевых ресурсов(2 час.)

1. 1.Разработать приложение, выполняющее поиск доступных сетевых ресурсов (Создать проект и перенести на него необходимые компоненты).
2. Создать структуру «сетевое окружение» (Создать форму, в разделе public объявите используемые в программе дополнительные процедуры и функции, нанести на форму необходимые компоненты).

Занятие 7. Освоение методов работы с протоколом NETBIOS (2 час.)

1. Разработка приложения, получающего список сетевых адаптеров, установленных на компьютере.
2. Доработка приложения с целью вывода информациио адаптерах на экран.

Занятие 8. СокетыWINDOWS. Создание клиент-серверного приложения работающего по протоколуUDP (2 час.)

1. Инициализация интерфейса сокетов. Создание сокета.
2. Определение номера порта и IP-адреса, по которому сервер будет принимать данные. Передача данных.
3. Получение данных по сети. Закрытие сокета. Деинициализация интерфейса сокетов.

Занятие 9. Исследование возможностей использования сокетов WINDOWSдля организации обмена информацией между клиентом и сервером по протоколу TCP (2 час.)

1. Разработка серверного приложения, которое должно выполнять следующие действия:

-подключить библиотеку Winsock;

-инициализировать интерфейс сокетов с помощью функции WSAStartup;

-создать сокет с помощью функции socket;

-настроить сокет с помощью функции setsockopt;

- объявить с помощью функции Bind, на каком порту сервер будет принимать данные и от кого;

- перевести сокет в режим пассивногоожидания с помощью функции listen;

-блокировать выполнение программы до тех пор, пока не поступит запрос на установление соединения с помощью функции accept;

-получить данные от любого клиента с помощью функции recv;

-вывести на экран полученное сообщение;

-возвратить полученное сообщение клиенту в эхорежиме с помощью функции send;

-закрыть сокет с помощью функции CloseSocket;

-прекратить работу с интерфейсом сокетов с помощью функции WSACleanup.

2.Разработка клиентского приложения, которое должно выполнять следующие действия:

- инициализировать интерфейс сокетов с помощью функции WSAStartup;

-создать сокет с помощью функции socket;

-установить соединение с сервером с помощью функции connect;

-передать данные на сервер с помощью функции send;

-получить ответ от сервера в эхорежиме с помощью функции recv;

-вывести эхоответ на экран;

-закрыть сокет с помощью функции CloseSocket;

-прекратить работу с интерфейсом сокетов с помощью функции

WSACleanup.

Занятие 10. Исследование методов сканирования сети(2 час.)

1. Создание многопоточного приложения для сканирования сети.
2. Исследование различных методов сканирования сети

Занятие 11. Применение сокетного соединения для обеспечения сетевого взаимодействия клиента и сервера(2 час.)

1.Создать серверное приложение, которое должно выполнять следующие действия:

1. Задавать номер сокета, на котором сервер будет принимать запросы клиентов.
2. Переводить компонент TServerSocket в активное состояние.
3. Устанавливать соединение с клиентом.
4. Получать сообщения от любого клиента.
5. Выводить на экран полученные сообщения.

2.Создать клиентское приложение, которое должно выполнять следующие действия:

1. задавать номер IP-адрес и номер сокета сервера;
2. устанавливать соединение с сервером;
3. передавать на сервер сообщение.

Занятие 12. Исследование работы сервера времени INTERNET(2 час.)

1.Создать серверное приложение, которое должно выполнять следующие действия:

1. Задавать номер сокета, на котором сервер будет принимать запросы клиентов.
2. Переводить компонент TServerSocket в активное состояние.
3. Устанавливать соединение с клиентом.

4. Периодически с интервалом в 1 секунду передавать всем подсоединенным клиентам информацию о статусе виртуальной памяти и показания системных часов на сервере.

2.Создать клиентское приложение, которое должно выполнять следующие действия:

1. задавать IP-адрес сервера;
2. устанавливать соединение с сервером;
3. принимать от сервера показания системных часов и информацию о статусе виртуальной памяти.
4. Исследовать возможности сервер-приложения.

Занятие 13. Создание сетевых приложений, обеспечивающих обмен данными(2 час.)

1.Создать приложение, работающее следующим образом:

1. передача данных является односторонней: от клиента к серверу;
2. если данное приложение работает как клиент, то при нажатии кнопки «Передать» приложение считывает содержимое многострочного компонента Memo2 и передает все строки серверу;
3. если данное приложение работает как сервер, то для обработки получаемой or клиента информации создается дополнительный поток TclientDataThread.

2.Откомпилировать программу и проверьте ее работу для различных адресов и номеров портов.

Занятие 14. Исследование возможностей поддержки протокола TCP(час.)

1. Создать серверное приложение:

- перенести на форму необходимые компоненты;

-для события OnClick кнопки «Запустить сервер» записать соответствующий программный код (Первая команда определяет номер порта, по которому сервер будет принимать данные от клиента.Вторая команда переводит сервер в состояние ожидания сообщений от клиента);

-для события OnExecute компонента IdTCPServer1и для события OnClick кнопки «Завершение работы» записать соответствующий программный код;

- откомпилировать приложение.

2. Создатьклиентское приложение в соответствии с нижеследующими пунктами:

- перенесите на форму необходимые компоненты;

- настроить панели группы радиокнопок;

- для событийOnActivateи OnClick формы Form1 записать соответствующий программный код;

- откомпилировать приложение.

3. Используя созданные приложения осуществить передачу данных.

Занятие 15. Программирование клиентских приложений для работы с серверами INTERNET (2 час.)

1.Создать клиентское приложение:

- перенести на форму компоненты в соответствии с заданием;

- подключить библиотеку WinInet (сделайте ссылку на библиотеку в списке подключаемых модулей - uses);

- в области глобальных переменных определите дескрипторы в соответствии с заданием;

- для событийOnCreale, OnClick кнопки «Установление соединения с WEB-сервером», OnClick кнопки «Установление соединения сFTP - сервером» и OnCIick кнопки «Вывод каталога» формы Form1 записать программный код в соответствии с заданием;

- откомпилировать приложение.

2. Используя разработанное приложение выполнить чтение большого файла с Web-сервера.

Занятие 16. Приём и передача файлов по сети с использованием протокола TFTP (2 час.)

1Создатьсерверное приложение:

- перенести на форму компоненты в соответствии с заданием;

- для событий формы OnCreate,OnChange компонента DriveComboBox1,OnChange компонента DriveComboBox1, OnReadFile компонента IdTrivialFTPServer1, OnWriteFile компонента IdTrivialFTPServer1 и OnTransferComplete компонента IdTrivialFTPServer1 написать программный код в соответствии с заданием;

- откомпилировать созданное приложение.

2. Создать клиентское приложение:

- перенести на форму компоненты в соответствии с заданием;

- для событийOnClick кнопки «Выбор файла», OnClick кнопки «Запись файла на сервер» иOnClick кнопки «Чтение файла с сервера» написать программный код в соответствии с заданием.

- откомпилировать созданное приложение.

3. Проверить совместную работу серверного и клиентского приложений.

Занятие 17. Создание HTTP- клиента (4 час.)

1.СоздатьHTTP-клиент:

- для создания приложения перенести на форму компоненты в соответствии с заданием;

- определить события в соответствии с заданием;

2. Запрограммируйте метод TRACE как функцию.

3. Запрограммируйте метод PUT как процедуру.

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Вычислительные системы, сети и телекоммуникации» представлено в Приложении 1 и включает в себя:

1. план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию;
2. характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению;
3. требования к представлению и оформлению результатов самостоятельной работы;
4. критерии оценки выполнения самостоятельной работы.

1. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
1	Модуль 1	ПК-28	знает	УО-1	Тестирование
		ПК-15	умеет	ПР-7	Опрос
		ПК-17	владеет	ПР-1	Практическое задание
2	Модуль 2	ПК-15	знает	УО-1	Тестирование
		ПК-28	умеет	ПР-2	Опрос
			владеет	ПР-12	Практическое задание
3	Модуль 3	ПК-28	знает	УО-4	Тестирование
		ПК-17	умеет	ПР-2	Опрос
		ПК-15	владеет	ПР-12	Практическое

					задание

2. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

1. Абросимов Л.И. Базисные методы проектирования и анализа сетей ЭВМ [Электронный ресурс] : учебное пособие / Л.И. Абросимов. — Электрон.текстовые данные. — М. : Логос, Университетская книга, 2015. — 248 c. — 978-5-98699-153-5. — Режим доступа: http://www.iprbookshop.ru/33078.html

2. Зиангирова Л.Ф. Вычислительные системы, сети и телекоммуникации [Электронный ресурс] : учебно-методическое пособие / Л.Ф. Зиангирова. — Электрон.текстовые данные. — Саратов: Вузовское образование, 2015. — 150 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/31942.html

3. Куроуз, Джеймс. Компьютерные сети : Нисходящий подход / Джеймс Куроуз, Кит Росс. 6-е изд. — Москва: Издательство «Э», 2016. — 912 с. https://vk.com/doc306640305_440290462

4. Лабораторный практикум по дисциплине Методы и средства защиты информации в компьютерных сетях [Электронный ресурс] / . — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2015. — 58 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61742.html

5. Мухутдинов Э.А. Некоторые проблемы в сетях и способы их решения [Электронный ресурс] : учебное пособие / Э.А. Мухутдинов, С.П. Плохотников. — Электрон.текстовые данные. — Казань: Казанский

национальный исследовательский технологический университет, 2015. — 109 c. — 978-5-7882-1766-6. — Режим доступа: http://www.iprbookshop.ru/62206.html

6. Нужнов Е.В. Компьютерные сети. Часть 2. Технологии локальных и глобальных сетей [Электронный ресурс] : учебное пособие / Е.В. Нужнов. — Электрон.текстовые данные. — Таганрог: Южный федеральный университет, 2015. — 176 c. — 978-5-9275-1691-9. — Режим доступа: http://www.iprbookshop.ru/78675.html

7. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 5-е изд. — СПб.: Питер, 2016. — 992 с. https://proglib.io/p/network-books/

8. Практикум по выполнению лабораторных работ по дисциплине Методы и средства защиты компьютерной информации [Электронный ресурс] / . — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2015. — 58 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61743.html

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Курс структурирован по тематическому, что позволяет систематизировать учебный материал.

В процессе изучения материалов учебного курса предлагаются: чтение лекций, лабораторные работы.

Лекционные занятия призваны ориентировать студентов в предлагаемом материале, заложить научные основы для дальнейшей самостоятельной работы студентов.

Лабораторные работы акцентированы на приобретения навыка работы с программными средствами.

Особо значимой для профессиональной подготовки студентов является *самостоятельная работа* по курсу. В ходе этой работы студенты применяют знания, полученные на лекциях и лабораторных работах.

Освоение курса должно способствовать развитию навыков самостоятельного принятия решения.

Методические указания по сдаче зачета

Зачеты принимаются ведущим преподавателем. При большом количестве групп у одного преподавателя или при большой численности потока по распоряжению заведующего кафедрой (заместителя директора филиала по учебной и воспитательной работе) допускается привлечение в помощь ведущему преподавателю других преподавателей. В первую очередь привлекаются преподаватели, которые проводили практические, лабораторные или семинарские занятия по соответствующей дисциплине в группах.

В исключительных случаях, по согласованию с заместителем директора Школы по учебной и воспитательной работе, заведующий кафедрой имеет право принять экзамен или зачет в отсутствие ведущего преподавателя.

Форма проведения зачета и экзамена (устная, письменная и др.) утверждается на заседании кафедры по согласованию с руководителем в соответствии с рабочей программой учебной дисциплины.

При подготовке студенту разрешается оформлять ответы на вопросы в письменной форме полностью или тезисно. Оценка студенту объявляется после окончательного ответа по билету, в том числе и по дополнительным вопросам.

10. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Компьютерный класс со стандартным набором программ из установленной операционной системой.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Сетевые технологии и системное администрирование

Направление подготовки 09.03.03 Прикладная информатика
Прикладная информатика в компьютерном дизайне
Форма подготовки очная

г. Владивосток

2018

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Вид самостоятельной работы	Примерные нормы времени на выполнение	Форма контроля
1.	Сентябрь,октябрь,ноябрь	Задание 1	месяц	Сдача на консультации
2.	Декабрь, январь	Задание 2	месяц	Тестирование
3.	Февраль, март	Задание 3	месяц	Сдача на практике
4.	Апрель, май	Задание 4	месяц	Сдача на практике Тестирование

1. ЗАДАНИЕ № 1

СОЗДАНИЕ WEB-БРАУЗЕРА

Цель работы: практически освоить приемы работы с компонентом WebBrowser для программирования клиентских приложений при работе с серверами Internet.

1. ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

Web-браузер должен обладать следующими основными возможностями:

1. заносить URL, для связи с Internet;
2. переключаться в ранее открытые страницы и документы;
3. иметь строку состояния, отображающую процесс загрузки документа.

Для построения Web-браузера средствами Delphi используется компонент WebBrowser, расположенный на странице Internet*Палитры компонентов*.

Основным методом компонента WebBrowser является метод Navigate, например, WebBrowser1.Navigate('D:\tests\file.html').

С помощью метода Navigate ищется указанные файл или адрес URL и загружается в браузер затребованный ресурс.

Для создания Web-браузера, обладающего требуемыми функциями необходимо использовать обработчики следующих событий компонент WebBrowser:

1. OnBeforeNavigate2 (функция-обработчик данного события имеет имя -WebBrowserlBeforeNavigate2) - наступает перед переходом браузера на новый документ. Это событие наступает независимо от того, чем вызван этот переход: выполнением метода Navigate или переходом по ссылке в окне браузера. Передаваемый в обработчик параметр «URL» является адресом, по которому будет осуществляться переход. Параметр «Cancel», если задать ему значение false, прервет переход на новый документ;
2. OnProgressChange - (функция-обработчик этого события имеет имя WebBrowserlProgressChange). Это событие происходят во время загрузки страницы в браузер. Для определения имени загружаемого документа используется свойство «LocationName» компонента WebBorowserl. В обработчик события передаются параметры Progress - объем загружаемого документа и ProgressMax - полный объем документа. Таким образом, появляется возможность отображать ход загрузки больших документов. Когда загрузка завершена, значение Progress становится равным – 1.

2. ЗАДАНИЕ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ

Для создания приложения WEB-БРАУЗЕРА, изображенного па рис. 1 выполните следующие действия.

1) Создайте проект.

2) Добавьте на форму меню MainMenu со страницы Standart*Палитры компонентов*. Меню должно иметь вид, изображенный на рис. 2.

3) Добавьте на форму панель CoolBar (страница Win32 *Палитры компонентов*). Перенесите на панель CoolBar инструментальную панель ToolBar (страница Win32 *Палитры компонентов*).

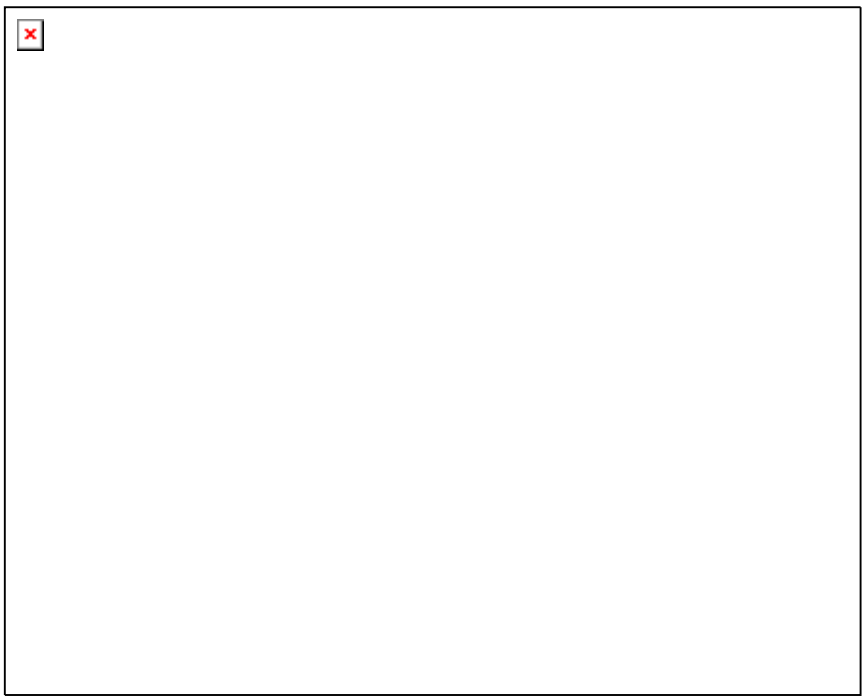

Рис. 1

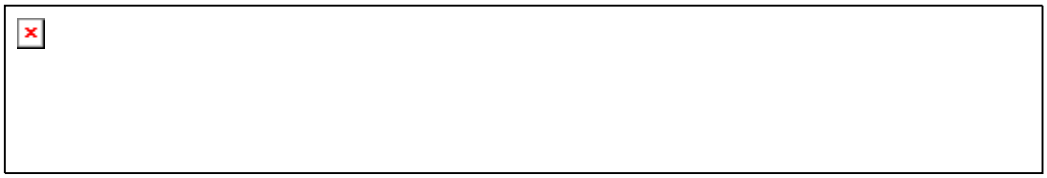

Рис. 2

4) Установите в true ее свойство ShowCaptions, чтобы можно было видеть надписи на кнопках.

a) Введите на панель ToolBar щелчками Правой кнопки «мыши» три кнопки - «Назад», «Вперед» и «Домашняя страница». Установите для каждой из них соответствующую надпись в поле *Caption*.

b) Для первых двух кнопок («Назад» и «Вперед») задайте свойство *Enabled*равным *false*, так как в момент открытия приложения перемещаться будет некуда.

c) Для кнопок можно задать пиктограммы. Для этого перенесите на форму список пиктограмм ImageList (страница Win32 *Палитры компонентов*). Щелкните два раза на списке ImageList для вызова диалога

добавления пиктограмм. Для панели ToolBar установите свойство *Images*равным ImageList1 для связи панели ToolBar со списком пиктограмм.

4) Перенесите на панель CoolBar компонент выпадающего списка ComboBox (страница Standart*Палитры компонентов*). В окне редактирования этого списка пользователь может набирать URL или имя файла документа HTML, а из выпадающего списка выбирать один из ранее просмотренных документов.

5) Установите для панели Coolbar ее свойства *AutoSize*и *ShowText*в true. Первое из них обеспечит автоматическое изменение размеров панели при перемещении пользователем расположенных на ней компонентов. А второе обеспечит видимость текстов полос.

6) Около свойства Bands панели CoolBar щелкните на кнопке с многоточием и в открывшемся окне редактора полос для второй полосы, на которой расположено поле со списком (ComboBox), задайте свойство *Text*равным «Адрес».

7) Перенесите на форму панель состояния StatusBar (страница Win32 *Палитры компонентов*). Установите ее свойство *SimplePanel*в *True*.Это означает, что панель не будет делиться на несколько панелей.

8) Перенесите на форму компонент WebBrowser (страница Internet*Палитры компонентов*). Установите свойство *Align*равным *alClient*, для того, чтобы окно просмотра HTML-документов занимало всю свободную часть формы. Теперь форма будет иметь вид, показанный на рис. 1.

9) Далее необходимо запрограммировать:

а) для события OnCreate формы Form1 запишите следующий программный код:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
// В ComboBox 1 записывается домашняя страница
ComboBox1.Text:='http://myself/www/Main1.html';
WebBrowser1.Navigate( ComboBox1.Text);
```

```
end;
```

b) для события OnClick компонента ComboBox запишите следующий программный код:

```
procedure TForm1.ComboBox1Click(Sender: TObject);
begin
// Web-браузер читает страницу,
//адрес которой указан в текущей строке ComboBox
WebBrowser1.Navigate(ComboBox1.Text);
CoolBarl.Update;
end;
```

c) для события OnkeyDown компонента ComboBox запишите следующий программный код:

```
procedure TForml.ComboBox1KeyDown(Sender:  TObject; var Key:
Word; Shift: TShiftState);
Begin
//Еслинажатаклавиша Enter, то Web-браузерсчитываетстраницу
If Key=VK_Return then
begin
WebBrowserl.Navigate(ComboBoxl.Text);
end;
end;
```

d) для события OnCiick кнопки «Назад» запишите следующий программный код:

```
procedure TForml.ToolButton1Click(Sender: TObject);
begin
// Выбираетсяпредыдущийадрес
ComboBox1.Text := ComboBox1.Items[ComboBox1.itemIndex+1];
WebBrowserl.Navigate(ComboBox1.Text);
end;
```

e) для события OnCiick кнопки «Вперед» запишите следующий программный код:

```
procedure TForm1.ToolButton2Cleck( Sender: TObject);
begin
// Выбираетсяследующийадрес
ComboBox1.Text := ComboBox1.Items[ComboBox1.ItemIndex-1];
WebBrowserl.Navigate(ComboBoxl.Text);
end;
```

f) для события OnClick кнопки «Домашняя страница» запишите следующий программный код:

```
procedure TForm1.ToolButton3Click(Sender: TObject);
begin
// определение страницы но умолчанию
```

```
ComboBox1.Text := 'http://myself/www/Main1.html';
WebBrowserl.Navigate(ComboBox1.Text);
end;
```

g) для события OnBeforeNavigate2 компонента WebBrowserl запишите следующий программный код:

```
procedure TFortn1.WebBrowser1BeforcNavigate2(Sender: TObject;
constpDisp: IDispatch; var URL, Flags, TargetFrameName, PostData,
Headers: OleVariant; var Cancel: WordBool);
// Обработка события перед загрузкой
Var
Index: Integer;
begin
// Управление списком  ComboBox1
Index:=ComboBoxl.Items.IndexOf(URL);
If Index=1 then
begin
ComboBox1.Items.Insert(0,URL);
ComboBo1.ItemIndex :=0;
end
elseComboBoxl.ItemIndex := Index;
// Заданиедоступностикнопок
IfComboBox1.ItemIndex>0
then  ToolButton2.Enabled := True
else ToolButton2.Enabled := False;
```

```
If ComboBox1.ItemIndex<ComboBoxl.Items.Count-1
then ToolButton1.Enabled := True
else ToolButton1.Enabled := False;
end;
```

Вобработчике WebBrowserlBeforeNavigate2 преждевсегоопределяетсяметодомIndexOfкомпонентаComboBox 1 индекснового URL всписке. Если метод IndexOf вернул – 1, значит данного адреса ранее в списке не было. В этом случае он включается методом Insert компонента ComboBox 1 в первую (с индексом 0) позицию списка и индекс списка устанавливается на 0. Если новый адрес уже был в списке, то индекс списка устанавливается равным этому значению. В обоих случаях изменение индекса списка обеспечивает отображение в его окне нового адреса. Далее в обработчике WebBrowserlBeforeNavigale2 устанавливается доступность или недоступность кнопок навигации TooButton2 (Вперед) и ToolButton1 (Назад) в зависимости оттого, каково значение индекса и есть ли в списке предшествующие и последующие документы;

h) для события OnProgressChange компонента WebBrowser1 запишите следующий программный код:

```
procedure TForml.WebBrowser1ProgressChange(Sender: TObject; Pro-
gress,
ProgressMax: Integer);
// Занесение текста в панель состояния
begin
If Progress>0 then
Statusbar1.SimpleText: Format('Документа %s: прочитано %d Кбайт
из %d', [WebBrowser1.LocationName,Progress div l024, Progressmax
div 1024]);
end;
```

ВобработчикеWebBrowserlProgressChangeиспользованосвойствокомпонентаWebBrowserl «LocationName» имязагружаемогодокумента.

n) для щелчка на разделе меню «Открытие файла» запишите следующий программный код:

```
procedure TForm1.N10Click(Sender: TObject);
begin
// Вызовдиалогаоткрытияфайла
If OpenDialogl.Execute then
begin
ComboBox1.Text := OpenDialogl.FileName;
WebBrowser1.Navigate(ComboBox1.Text);
end;
end;
```

Ими выбранного пользователем файла загружается в свойство *Text*компонента ComboBoxl.

12) Самостоятельно запрограммируйте все остальные пункты меню, изображенные на рис. 2.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Из каких частей состоит и для чего используется URL?

2. Перечислите основные свойства компонента WebBrowser.

3. Перечислите основные события компонента WebBrowser.

ЗАДАНИЕ № 2

ИССЛЕДОВАНИЕ TELNET-КЛИЕНТА

Цель работы: исследовать простейшей Telnet-клиент, с помощью которого можно подключиться к любому серверному порту и выполнять команды прямо на сервере.

1. ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

Для создания Telnet-клиента будет использоваться компонент IdTelnet расположенный на закладке IndyClients*Палитры компонентов*.

При создании Telnet-клиента будут использоваться следующие свойства компонента:

1. Terminal - вид терминала;
2. Host - IP-адрес иди имя сервера;
3. Port - номер порта сервера;
4. Connected - признак установленного соединения. Для взаимодействия с удаленным сервером Telnet-клиент примени следующие методы:
5. Connect - установление соединения;
6. Disconnect - разрыв соединения;
7. SendCh - передача очередного символа на сервер. При получении ответа от сервера компонент Idtelnet обрабатывает следующие события;
8. OnConnected - событие происходит при установлении соединения с сервером;
9. OnDataAvailable - событие происходит при получении данных сервера;
10. OnStatus событие происходит при изменении состояния канала связи с сервером.

2. ЗАДАНИЕ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ

2.1. Создание приложения Telnet-клиента

Создайте приложение Telnet- клиента, изображенное на рис. 1, для чего необходимо выполнить следующие пункты.

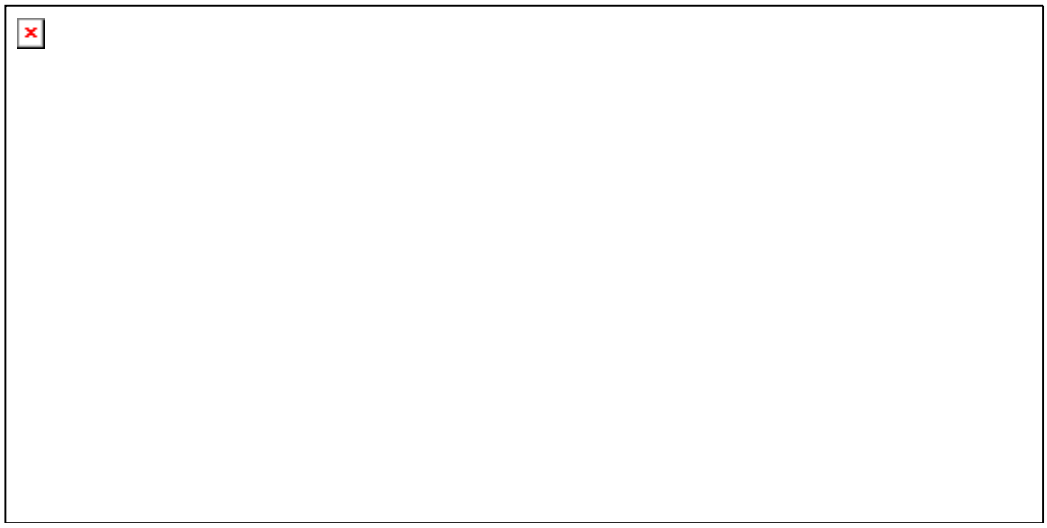

Рис. 1

1. Перенесите на форму компоненты, перечисленные в табл.

Таблица 1

Компоненты, устанавливаемые на форме

Компонент	Класс	Описание
IdTelnet1	TIdTelnet	Компонент telnet-клиент. Закладка IdClients*Палитры компонентов*
StatusBarl	TSlatusBar	Панель состояния. Закладка Win32 *Палитры компонентов*
Panel1	TPanel	Панель, на которой размешаются командные кнопки и окна ввода приложения
Memo1	TMemo	Многострочное окно ввода. В данном окне отображаются команды, вводимые пользователем, и выводятся ответы серверов
Label1	TLabel	Метка «Сервер»

Label2	TLabel	Метка «Порт»
Labe3	TLabel	Метка «Терминал»
Edit1	TEdit	Окно ввода IP-адреса или имени сервера
SpinEdit1	TSpinEdit	Окно ввода номера порта. Закладка Samples*Палитры компонентов*
ComboBoxl	TComboBox	Список выбираемых терминалов
Buttonl	TButton	Кнопка «Connect»
Button2	TButton	Кнопка «Disconnect»

2) Для компонента Editl (Сервер) установите в свойстве TextIP-адрес или имя сервера. В данном примере стоит адрес- 127.0.0.1.

3) Для компонента SpinEdit1 (Порт) установите в свойстве Value значение порта сервера по умолчанию. В данном примере стоит номер порта 21 (FTP -сервер).

4) Для компонента ComboBox 1 (Терминал) установите в свойстве Text значение - VT100, а в свойстве Items - список терминалов, как показано на рис. 2.

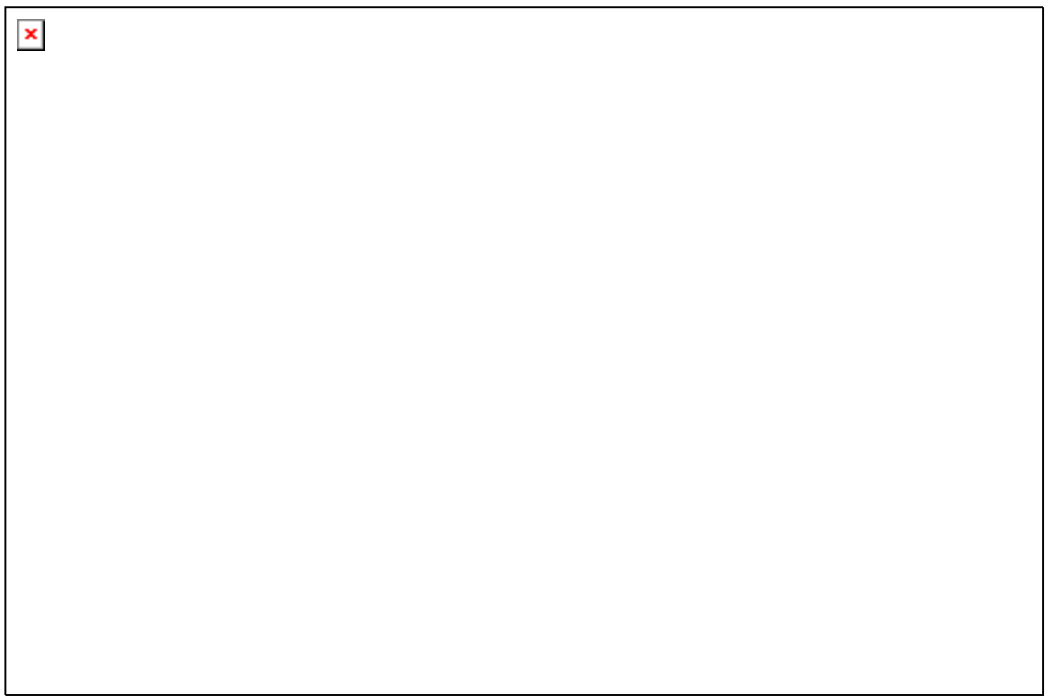

Рис. 2

5) Для события OnCiick кнопки «Conned» запишите следующий программный код:

```
procedure TTelnetForm.Button1Click(Sender: TObject);
begin
IdTelnetl.Terminal := ComboBox1.Text;
IdTelnetl.Host := Edit1.Text;
IdTelnetl.port := SpinEditl.Value;
IdTelnetl.Connect;
end;
```

Этот программный код записывает для компонента IdTelnet1 тип пользуемого терминала, адрес сервера, номер порта сервера и выполняет соединение с сервером.

6) Для события OnCiick кнопки «Disconnect» запишите следующий программный код:

```
procedureTTelnetForm.Button2Click(Sender: (Object);
begin
If IdTelnet1.Connected then IdTelnetl.Disconnect;
end;
```

Здесь вызывается метод Disconnect компонента IdTelnetl, что приводит к отключению от сервера.

7) Для отправки команд на сервер для компонента Memo1 следует создать обработчик события OnKeyPress. В этом обработчике напишите следующий программный код:

```
procedureTTelnetForm.Memo1KeyPress(Sender: TObject: varKey: Char);
begin
if IdTelnetl.Connected then
begin
IdTelnetl.SendCh(Key);
end;
end;
```

Здесь происходит проверка: если компонент IdTelnetl подключен к серверу, то символ нажатой клавиши нужно передать на сервер. Для этого используется метод SendCh компонента IdTelnetl, а в качестве параметра этому методу передается нажатый символ. Теперь при нажатии любой клавиши для ввода символа в компонент Memol этот символ сразу же передается на сервер.

8) Для события OnConnected компонента IdTelnetl запишите следующий программный код:

```
procedure TTelnetForm.IdTelnetlConnected(Sender: TObject);
begin
Memol.Lines.Add('Клиентподключен');
Memol.Lines.Add('Может выполнять команды');
Memol.Lines.Add(‘’);
end;
```

Работа этого обработчика направлена на то, чтобы проинформировать пользователя о том, что соединение произошло успешно.

9) Для события OnDataAvailable компонента IdTelnetl запишите следующий программный код:

```
procedure TTelnetForm.Id'T'elnet1 DataAvailable(Sender: TldTelnet;
constBuffer: String);
const
// коды символов возврата строки и перевода каретки
CR=#13;
LF=#10;
var
Start, Stop: Integer;
Begin
Memol.Lines. Add(");
Start := 1;
```

```
// Поиск символа возврата каретки
Stop :=Pos(CR, Buffer);
// Если символ возврата каретки не найден, то в поле Memo1 выводится
// содержимое всего полученного буфера данных
ifStop = 0 then
Stop :=Length(Buffer) + 1;
// Цикл, пока не встретится конец буфера
while Start <= Length(Buffer) do
begin
//Запись в поле Memo1 очередной строки
Memo1.Lines.Strings[Memo1.Lines.Count-1] :=
Memo1.Lines.Strings[Memo1.Lines.Count - l]+Copy(Buffer, Start, Stop -
Start);
if Buffer[Stop]=CR then
begin
Memol.Lines.Add('');
end;
Start:=Stop + 1;
//Обработка строки в полученном от сервера буфере данных
if Start>Length(Buffer) then
Break;
If Buffert[Start]=LF then
Start:=Start + 1;
Stop := Start;
while (Buffer[Stop]<>CR) and (Stop<= Length(Buffer)) do
```

```
Stop:=Stop + 1;

end;

end;
```

Этот обработчик вызывается каждый раз, когда с сервера поступаю данные.

Весь написанный в листинге код направлен на вывод пришедшего текста с помощью компонента Memol. Для этого ищутся символы конца строки и перевода каретки, и если они найдены, то в компонент добавляется новая строка. Для облегчения поиска заведены две константы CR и LF с шестнадцатеричными значениями #13 и #10, которые являются кодами символов конца строки и перевода каретки.

10) Для события OnStatus компонента IdTelnet1 запишите следующий программный код:

```
procedureTTelnetForm.IdTelnetl Status(axSender: TObject; constaxStatus: TIdStatus: constasStatusText: Siring); begin

if axStatus =hsDisconnected then

Memo1.Lines.Add(‘Сессиязакончена');

end;
```

11) Для события OnCloseQuery формы запишите следующий программный код:

```
procedure TTelnetForm.FormCloseQuery(Sender: TObject;

varCanClose: Boolean);

begin

if ldTelnetl.Connected then
```

```
IdTelnetl.Disconnect;
end;
```

Здесь при попытке закрыть приложение выполняется проверка наличия соединения с каким-либо сервером, и если соединение установлено, то оно разрывается.

2.2. Исследование Telnet-клиента

1) Выполните проверку и исследуйте соединение данного Telnet-клиента с сервером Echo (порт 7). Сервер возвращает все данные, которые ему послали, то есть любой переданный на сервер символ будет им возвращен.

2) Выполните проверку и исследуйте соединение данного telnet-клиента с сервером времени (порт 13). Сервер времени должен переслать клиенту точные дату и время.

3) Выполните проверку и исследуйте соединение данного telnet-клиента с сервером FTP (порт 21). После того как соединение установлено, выполните следующие команды:

1. User anonymous;
2. Pass;
3. Help;
4. List;

ит. д.;

5. Quit.

Последняя команда разрывает соединение с FTP-сервером.

4) Выполните проверку и исследуйте соединение данного telnet-клиента с сервером электронной почты SMTP (порт 25). После того как соединение установлено, выполните следующие команды:

1. Echo;

ит. д.

2. Quit.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение Telnet-клиента.

2. С помощью каких команд компонент IdTelnet устанавливает и разрывает соединение?

3. Как передать данные на сервер с помощью компонента IdTelnet?

4. Какое событие компонента IdTelnet происходит, когда поступают данные от сервера?

ЗАДАНИЕ №3

РАЗРАБОТКА CGI-ПРОГРАММ

Цели работы: практически освоить методы создания CGI-программ на основе консольных приложений Delphi, разработать исполняемый файл, устанавливаемый на Web-сервере и выполняющий определенные действия в зависимости от получаемых параметров.

1. ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

В теоретическом введении рассматриваются следующие вопросы.

3. Принцип работы CGI-программ.
4. Описание тега FORM.
5. Описание тега FORM.
6. CGI-переменные окружения.

1.1. Принцип работы CGI-программ

CommonGatewayInterface - это спецификация интерфейса взаимодействия Web-сервера с внешними прикладными программами. Главное назначение CGI - обеспечение единообразного потока данных между сервером и работающим на нем приложением. Программы, написанные в соответствии со спецификацией CGI, называются CGI-скриптами (CGI-программами).

Обычно гипертекстовые документы, возвращаемые по запросу клиента WWW-сервером, содержат статические данные. CGI обеспечивает средства создания динамических Web-страниц на основе данных, полученных от пользователя. CGI-программа запускается Web-сервером, возвращает результаты работы серверу и завершает свое выполнение.

Схема работы CGI-программы приведена на рис. 1.

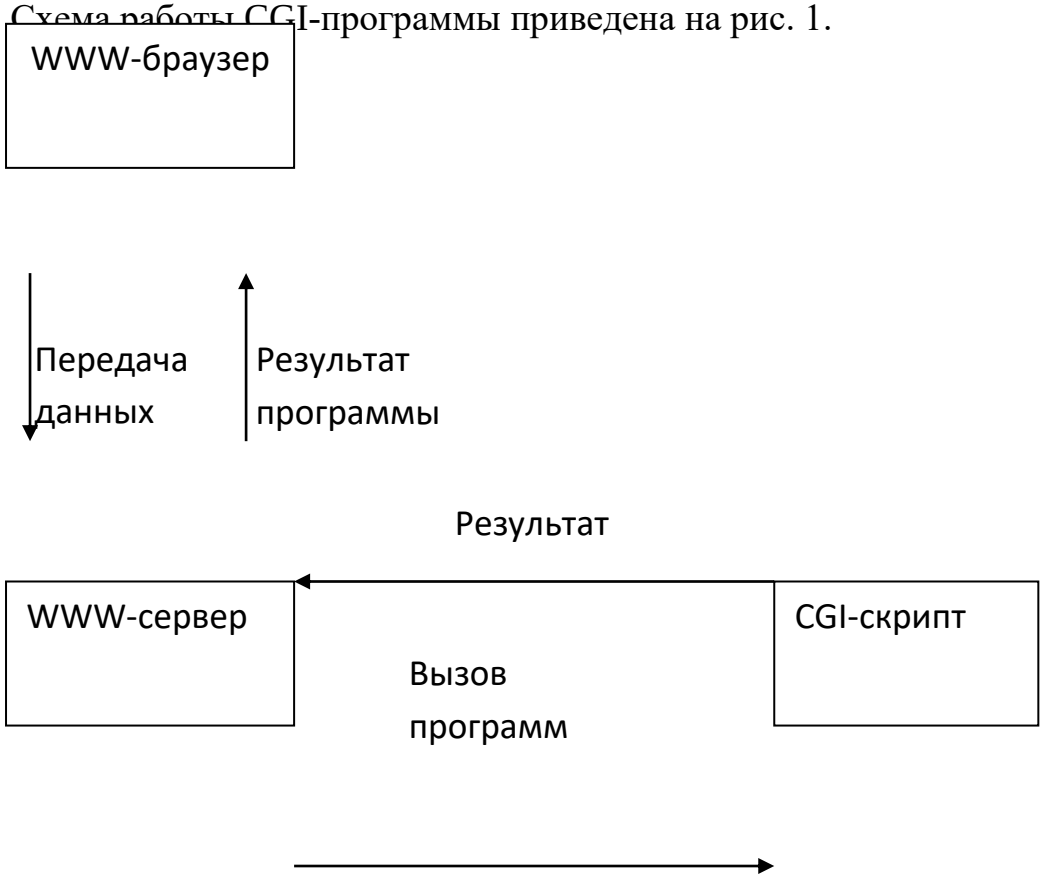

Рис. 1. Схема взаимодействия CGI-программы

CGI-программа работает по следующему принципу.

1) CGI-программа устанавливается на Web-сервере.

2) Пользователь запускает HTML-страницу, содержащую тег FORM.

3) С помощью тега FORM пользователь вводит параметры запроса и передает запрос на Web-сервер.

4) Web-сервер передает полученные данные CGI-программе, указанной в запросе.

5) CGI-программа обрабатывает запрос и формирует новую HTML-страницу, которую возвращает Web-серверу.

6) Web-сервер передает данную страницу клиенту.

Обмен данными по спецификации CGI реализуется обычно через переменные окружения и стандартный ввод-вывод. Выбор механизма передачи параметров определяется методом доступа, который указывается в форме в атрибуте METHOD. Если используется метод GET, то передача параметров происходит с помощью переменных окружения, которые сервер создает при запуске внешней программы. Через них передается приложению как служебная информация (версия программного обеспечения, доменное имя сервера и др.), так и сами данные (в переменной QUERY_STRING). При методе POST для передачи используется стандартный ввод. А в переменных окружения фиксируются тип и длина передаваемой информации (CONTENT_TYPE и CONTENT_LENGTH).

Стандартный вывод используется CGI-скриптом для возврата данных серверу. При этом вывод состоит из заголовка и собственно данных. Результат работы скрипта может передаваться клиенту без каких-либо преобразований со стороны сервера, если скрипт обеспечивает построение полного HTTP-заголовка, в противном случае сервер модифицирует заголовок в соответствии со спецификацией HTTP.

1.2. Описание тега FORM

Тег FORM определяет форму для заполнения в HTML-документе. В одном документе может быть определено несколько форм для заполнения, но вложенные теги FORM запрещены. Формат тега FORM выглядит следующим образом:

<FORMACTION="url" METHOD ="POST”>...</FORM> .

Его атрибуты следующие:

ACTION - URL сервера запросов, куда будет отослано содержание формы после подтверждения. Если это поле отсутствует, будет использовано URL текущего документа. Данное поле указывает на CGI-программу.

METHOD - метод, используемый для посылки содержания заполненной формы на сервер. Этот метод зависит от того, как работает конкретный сервер запросов. Настоятельно рекомендуется использование метода POST.

Возможные варианты следующие:

• GET - это метод по умолчанию, который приводит к добавлению содержимого заполненной формы к URL, как и в нормальном запросе;

• POST - при использовании этого метода содержимое заполненной формы пересылается не как часть URL, а как содержимое тела запроса;

• ENCTYPE - задает тип кодирования содержимого заполненной формы. Этот атрибут действует только когда используется метод POST и даже в этом случае имеет только одно возможное значение (которое является значением но умолчанию) - application/x-www-form-urlencoded.

Внутри FORM оператора может находиться все, что угодно, кроме другого оператора FORM. Согласно спецификации, для задания интерфейсных элементов внутри оператора FORM используются теги INPUT, SELECT и TEXTAREA.

1.2.1. Тег INPUT

Тег INPUT используется для задания простого элемента ввода внутри FORM. Это одиночный тег, его ничего не окружает, и он не имеет закрывающего тега.

Атрибуты для тега INPUT перечислены ниже.

TYPE должен быть один из:

1. «hidden» - пользователю не предлагаются поля для ввода, но содержимое тега передается при подтверждении и посылке формы. Это значение может быть использовано для передачи информации состояния при взаимодействии клиента и сервера;

2. «image» - картинка, по которой вы можете сделать щелчок «мышью» или другим указывающим устройством, что приводит к немедленному подтверждению и отсылке формы. Координаты выбранной точки измеряются в пикселях от верхнего левого угла и возвращаются (наряду с другими компонентами формы);

3. «text» - поле ввода текста, значение по умолчанию;

4. «password» - поле ввода пароля; вводимые символы представляются как звездочки;

5. «checkbox» - кнопка, принимающая положения on (включено) и off (выключено);

6. «radio» - кнопка, принимающая положения on и off; причем остальные кнопки с тем же параметром NAME ведут себя по принципу «одна из многих»;

7. «submit» - кнопка, действие которой сводится к отсылке содержимого заполненной формы на сервер запросов;

8. «reset» - кнопка, которая устанавливает во всех интерфейсных элементах значения по умолчанию.

NAME - символическое имя (оно не показывается) для этого поля ввода. Это поле должно присутствовать для всех полей ввода кроме «submit»

и «reset», т. к. оно используется в строке запроса при идентификации поля ввода при посылке ее на сервер после подтверждения формы.

VALUE - для полей ввода текста или пароля, может быть использовано для задания начального содержания поля. Для checkbox или radioVALUE задает значение кнопки, когда она находится в отмеченном состоянии (неотмеченные кнопки опускаются при посылке запроса); значение по умолчанию для checkbox или radio - «on». Для типов «submit» и «reset» VALUE может быть использовано для задания надписи на этих кнопках.

CHECKED - значение не требуется. Указывает, что данная кнопка типа checkbox или radio отмечена по умолчанию. Это поле работает только для кнопок типа checkbox и radio.

SIZE - физический размер поля ввода в символах; это поле действует только для элементов ввода текста или пароля.

MAXLENGTH - максимальное количество введенных символов, которые будут приниматься для ввода, верно только для полей ввода текста и пароля (и только в однострочных элементах). По умолчанию - неограниченно. Подразумевается, что поля ввода должны прокручиваться.

1.2.2. Тег SELECT

Внутри <FORM>... </FORM> может присутствовать любое количество тегов SELECT, свободно перемешанных с другими HTML-элементами (включая INPUT и TEXTAREA) и текстом (но не дополнительных элементов FORM). Тег SELECT во многих графических клиентах представляется как меню или список.

В отличие от INPUT, SELECT имеет открывающий и закрывающий теги. Внутри оператора SELECT разрешена только последовательность тегов OPTION, за каждым из которых следует некоторое количество простого текста (без HTML-выражений), например:

<SELECT NAME="a-menu">

<OPTION> First option.

<OPTION> Second option.

</SELECT>.

Атрибутыоператора SELECT следующие:

NAME - символическое имя для этого SELECT-элемента. Это поле должно присутствовать, т. к. оно используется при посылке запроса (аналогично оператору INPUT).

SIZE - если SIZE равен 1 или если этот атрибут опущен, по умолчанию SELECT будет представлен как меню опций Motif. Если SIZE=2 или более, SELECT будет представлен как окно выбора; значение SIZE тогда будет определять, сколько элементов списка будут видны.

MULTIPLE - если присутствует, то задает, что SELECT должен позволять множественный выбор из списка. Наличие MULTIPLE принуждает SELECT быть представленным как список выбора, вне зависимости от значения SIZE.

Атрибутом OPTION является SELECTED, который задает, что эта опция выбрана по умолчанию. Если SELECT позволяет множественный выбор

(с помощью атрибута MULTIPLE), то может быть помечено несколько опций.

1.2.3. Тег TEXTAREA

Тег TEXTAREA может быть использован для расположения многострокового поля ввода с необязательным содержимым по умолчанию в форме. Атрибуты тега TEXTAREA следующие:

NAME - символическое имя поля ввода;

ROWS - число строк в поле ввода (высота);

COLS - число столбцов в поле ввода (ширина).

TEXTAREA имеет полосы прокрутки, так что может быть введено любое количество текста. Элемент TEXTAREA требует и открывающий и закрывающий теги. TEXTAREA без содержания по умолчанию выглядит примерно так:

<TEXTAREA NAME-"foo" ROWS=4 COLS=40>

</TEXTAREA>

TEXTAREA с содержанием по умолчанию выглядит так:

<TEXTAREA NAME=”foo" ROWS=4 COLS=40>

Default contents go here.

</TEXTAREA>.

Содержание по умолчанию должно быть строгим ASCII-текстом. Символы перевода строки воспринимаются (в примере до и после текста «Defaultcontentsgohere» будет пустая Строка).

13. Подтверждение и посылка запроса CGI-программе

Для метода GET

Когда нажимается кнопка submit, содержимое формы будет добавлено к URL в следующей форме: action?name=value&name=value&name=value, где «action» - URL, заданное атрибутом ACTION в теге FORM, или URL текущего документа, если атрибут ACTION не был задан.

Для полей ввода текста и пароля значением будет то, что введет пользователь. Если пользователь оставит это поле пустым, значение value также будет пустым, но в строке запроса будет присутствовать фрагмент «name=».

Для кнопок типа checkbox и radio значение value определяется атрибутом VALUE в том случае, когда кнопка отмечена. Неотмеченные кнопки при составлении строки запроса игнорируются целиком. Несколько кнопок типа checkbox могут иметь один атрибут NAME (и различные VALUE), если это необходимо. Кнопки типа radio предназначены для того, чтобы вести себя по принципу «одна из всех» и должны иметь одинаковый атрибут NAME и различные атрибуты VALUE.

Для метода POST

Содержимое формы кодируется точно как для метода GET, но вместо добавления содержимого формы к URL, заданной атрибутом формы ACTION в качестве запроса, содержимое посылается блоком данных как часть операции POST. Если присутствует атрибут ACTION, то значение URL, которое там находится, определяет, куда послать этот блок данных. Этот метод особенно рекомендуется при посылке больших блоков данных

1.4. CGI-переменные окружения

Следующие переменные окружения CGI-программ устанавливаются для всех запросов.

SERVER_SOFTWARE - название и версия информационного сервера, который отвечает на запрос (и запускает шлюз). Формат: имя/версия.

SERVER_NAME - имя хоста, на котором запущен сервер; DNS-имя или IP-адрес в том виде, в котором он представлен в URL.

GATEWAY_INTERFACE - версия CGI-спецификации на тот момент, когда компилировался сервер. Формат: CGI/версия.

Следующие переменные окружения являются специфичными для разных запросов и заполняются Web-сервером перед вызовом CGI-программы:

SERVER_PROTOCOL - имя и версия информационного протокола, в котором пришел запрос. Формат: протокол/версия;

SERVER_PORT - номер порта, на который был послан запрос;

REQUEST_METHOD - метод, который был использован для запроса. Для HTTP - это «GET», «HEAD», «POST» ит.д.;

PATH_INFO - дополнительная информация о пути, которую передал клиент. Другими словами, доступ к шлюзу может быть осуществлен по виртуальному пути, за которым следует некоторая дополнительная информация. Эта информация передается в PATH_INFO;

PATH_TRANSLATED - сервер передает преобразованную версию PATH_INFO, которая включает путь, преобразованный из виртуального в физический;

SCRIPT_NAME - виртуальный путь к CGI-программе, которая должна выполняться (содержится в URL);

QUERY_STRING - информация, следующая за «?» в URL, к которое относится данная CGI-программа. Это информации представляет собой строку запроса. Она не должна быть декодирована никоим образом. Вне зависимости от командной строки эта переменная всегда должна быть установлена при наличии такой информации;

REMOTE_HOST - имя хоста, производящего запрос. Если сервер не имеет такой информации, он должен установить REMOTE_ADDR, а это поле оставить не установленным:

REMOTE_ADDR - IP-адрес хоста, производящего запрос;

AUTH_TYPE - если сервер поддерживает идентификацию пользователя и шлюз является защищенным от постороннего доступа, этот специ-

фичный для протокола метод идентификации используется для проверки пользователя;

REMOTE_USER - используется в ситуациях, аналогичных предыдущему случаю, для хранения имени пользователя;

REMOTE_IDENT- если HTTP-сервер поддерживает идентификацию пользователя согласно RFC 931, то эта переменная будет содержать имя пользователя, полученное от сервера;

CONTENT_TYPE - для запросов, которые содержат дополнительную добавочную информацию, такие, как HTTPPOST и PUT, здесь содержится тип данных этой информации;

CONTENT_LENGTH -длина данных, которую передает клиент.

2. ЗАДАНИЕ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ

2.1. Создание простейшего серверного приложения

Разработайте консольное приложение. Для этого выполните следующие действия.

1) Выберите в главном меню Delphi команду *File/ New*, затем выберите в открывшемся окне хранилища объектов значок ConsoleApplication и щелкните на кнопке *OK*.

ПРИМЕЧАНИЕ. Если в окне хранилища объектов отсутствует значок ConsoleApplication, то выберите значок Application, в главном меню Delphi выберите команду *View/ Units*и откройте файл проекта. Затем создайте шаблон консольного приложения, которой имеет следующий вид:

```
program Project 1;

{@APPTYPE CONSOLE}

uses

SysUtils:
```

```
Begin
// Здесь будет располагаться CGI-программа
end.
```

2) Введите следующий текст CGI-программы (вместо комментария):

```
// Заголовокответа
Writeln('Content-Type:text/html');
//Пустая строка, которая отделяет заголовок от тела ответа
Writeln;
Writeln('<HTML>');
Writeln('<HEAD>');
Writeln('<TITLE->ПримерCGI-приложения-</TITLE>');
Writeln('</HEAD>');
Writeln('<BODY>');
Wrileln('<H2 ALIGN=CENTER> Hello, World! </H2>');
Writeln('</BODY>');
Writeln('</HTML');.
```

3) Откомпилируйте полученное приложение и запишите полученный исполняемый файл в каталог Web-сервера, предназначенный для размещения исполняемых файлов и сценариев.

4) Для полученного приложения необходимо создать HTML-документ, из которого будет производиться вызов CGI-приложения. Так как в данном случае не требуется получать какие-либо данные от пользователя, то можно использовать вызов приложения с помощью тега «FORM» и как обычную ссылку (тег «A HREF=...»). Создайте следующий HTML-документ:

```
<HTML>
<HEAD>
<TITLE> Пример CGI-приложения</TITLE>
</HEAD>
<BODY>
<H2> Работа с тегом Form. Метод GET </H2>
<FORM METHOD="GET" ACT1ON=http://myself/temp/hello.exe>
<INPUT TYPE="SUBMIT">
</FORM>
<HR>
<BR>
<A HREF="http://myself/temp/hello.exe">
Вызов приложения с помощью тега "AHREF"
</A>
</BODY>
</HTML>.
```

5) Запустите созданный HTML-документ с помощью InternetExplorer (IE). В окне IE отобразится документ, содержаний одну кнопку и одну ссылку (рис. 2). При щелчке на любом из этих элементов в окне IE отобразится документ, соответствующий ответу запущенного CGI-сценария (рис. 3).

Рис. 2

Рис. 3

2.2. Передача CGI-приложению параметров, введенных пользователем, и получение дополнительной информации из переменных окружения

Создайте консольное приложение, результатом работы которого будет вывод используемого метода и передаваемой строки параметров.

1) Запишите в файл проекта следующий программный код:

```
program param;
{@APPTYPE CONSOLE}
uses
SysUtils,
Windows;
Var
buff: PChar;
ContentLength, i: Integer;
St1,St2: String;
C: Char;
Begin
// Выделение памяти под строку параметров
GetMem(buff,50);
// Получение строки параметров
GetEnvironmentVariable('REQUEST_METHOD', buff, 50):
//Преобразование строки PChar в паскалевскую строку
St1 := StrPas(buff);
// Освобождениепамяти
Freemem(buff);
i:=length(St1);
While i>0 do
begin
St1[i] :=Upcase(St1[i]);
dec(i);
end;
```

```
If St1='GET’ then
begin
GetMem(buff, 200);
GetEnvironmentVariable(‘QUERY_STRING', buff 200);
St2 :=StrPas(buff);
Freemem(buff);
end;
If St1=’POST’ then
begin
GetMem(buff, 50);
GetEnvironmentVariable('CONTENT_LENGTH', buff 50);
St2 := StrPas(buff);
Freemem(buff);
ContentLength :=StrToInt(St2);
St2 :='';
For i :=1 to ContentLength do
begin
Read(C);
St2 := St2+C;
end;
end;
St1:='Method'+St1;
St1 := '<H2 ALIGN=CENTER>'+St1+’</H2>';
Writeln('Content-Type: text/html');
Writeln;
```

```
Writeln('<HTML>');
Writeln('<HEAD>');
Writeln('<TITLE>Пример CGI-приложения</TITLE>');
Writeln(‘<</HEAD>');
Writeln('<BODY>');
Writeln('<H2 ALIGN=CENTER>METHOD’+St1+'</H2>');
Writeln('<BR>');
Writeln('<H2 ALIGN=CENTER>PARAMETRS:'+St2+'</H2>');
Writeln('</BODY>');
Writeln(‘</HTML>');
end.
```

2) Откомпилируйте полученное приложение и запишите полученный исполняемый файл в каталог Web-сервера, предназначенный для размещения исполняемых файлов и сценариев.

3) Для проверки работоспособности полученного приложения необходимо создать следующий HTML-документ:

```
<HTML>
<HEAD>
<TITLE> Пример CGI-приложения</T1TLE>
</HEAD>
<BODY>
<H2>Метод GET </H2>
<FORM METHOD="GET"
ACTION=http://myself/temp/param.exe>
```

<INPUT TYPE="TEXT" NAME="Edit1" VALUE="test">

<INPUT TYPE="SUBMIT">

</FORM>

<HR>

<H2>Метод POST </H2>

<FORM METHOD="POST”

ACTION="http://myself/temp/param.exe">

<INPUT TYPE="TEXT" NAME="Edit1" VALUE="test">

<INPUT TYPE="SUBMIT">

</FORM>

<BODY>

<HTML>.

4) Запустите созданныйHTML-документ с помощью IE. В окне IE отобразится документ, изображенный на рис. 4. При выборе метода GET

отобразится документ, изображенный на рис. 5, при выборе метода POST отобразится документ, изображенный на рис. 6.

Рис. 4

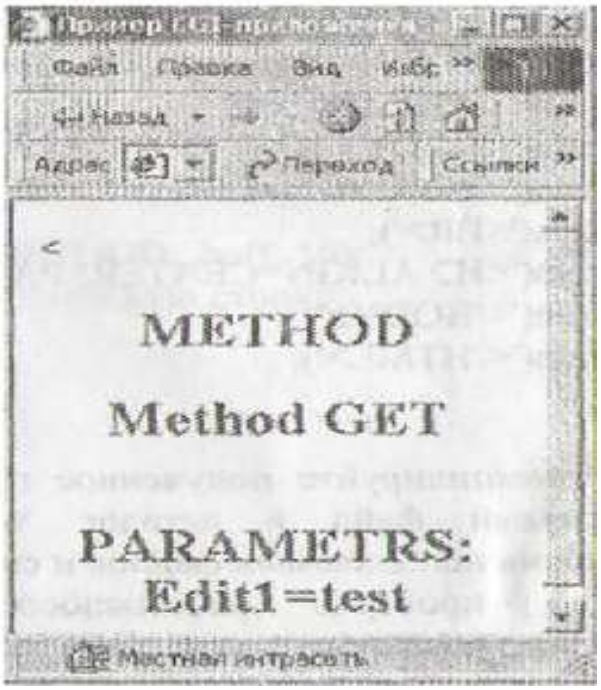

Рис. 5

Рис. 6

5) Измените программный код приложения таким образом, чтобы CGI-программа выводила на экран переменные окружения, указанные в табл. 1,

Таблица 1

Переменные окружения

Название	Описание
REQUEST_METHOD	Метод передачи информации от пользователя (GET или POST)
SERVER_NAME	IP-адрес или имя сервера
SERVER_PORT	Номер порта, используемый при обращении к серверу
SERVER_PROTOCOL	Название и версия протокола, по которому был передан запрос
PATH_INFO	Строка параметров, расположенная в запросе после имени приложения
REMOTE_ADDR	IP-адрес узла, с которого был послан запрос
REMOTE_HOST	Доменное имя узла, с которого поступил запрос

2.3. Разработайте CGI-приложение, которое проверяет результат умножения

1) Создайте HTML-документ, с помощью которого пользователь вводит число, которое, по его мнению, должно получиться в результате умножения 2x2. При нажатии на кнопку «SUBMIT» результат передается CGI-приложению.

2) CGI-приложение сравнивает полученное число с эталонным значением и возвращает результат пользователю.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные элементы тега FORM.
2. Чем отличаются методы GET и POST?
3. Перечислите особенности CGI-приложений.
4. Какие Вы знаете переменные окружения CGI-программ?

ЗАДАНИЕ № 4

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ

Цель работы: практически освоить специальные средства Delphi для разработки Web-приложений.

В теоретическом введении рассматриваются следующие вопросы.

• Общее описание компонента TWEBMODULE.

• Параметр REQUEST.

• Параметр RESPONSE.

• События TWEBMODULE.

1. ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

1.1. Общее описание компонента TWEBMODULE

Компонент TWebModule является основой любых Web-приложений, разрабатываемых в Delphi. С помощью этого компонента приложение осуществляет интерпретацию HTTP-запросов.

Основное свойство компонента TWebModule - свойство Action, которое содержит список действий, являющихся обработчиками запросов, поступающих от клиентов.

Каждый элемент этого списка имеет тип TWebActionItem и характеризуется следующими свойствами.

• PathInfo: String - указывает, при какой строке параметров (расположенной в запросе после имени программы, но до данных запроса) будет вызываться данное действие.

• MethodType: TMethodType - указывает метод, используемый клиентом, для передачи запроса, на который данное действие может ответить. Возможны следующие значения: mtGet, mtPost, mtHead, mtPut, mtAny.

• Default: Boolean - используется для задания обработчика по умолчанию. Если это свойство установлено равным true, то действие будет обрабатывать запросы со строками параметров, для которых не заданы обработчики.

• Enabled: Boolean - указывает, может (true) или нет (false) данное действие обработать HTTP-запрос с параметрами PathInfo и MethodType, соответствующими свойствам данного действия.

• Producer: TCustomContentProducer - указатель на специальный компонент, используемый для формирования ответа web-приложения. В данной работе не используется.

Каждый элемент списка Actions может обрабатывать всего одно событие OnActions. ОбработчиксобытияOnActionsимеетследующийвид:

procedure TWebModule1.WebModule1 WebActionItem1 Action (Sender: TObject; Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);.

С помощью параметра Request передается запрос, полученный от клиента. Параметр Response используется для формирования ответа. Параметр Handled применяется в том случае, когда требуется указать, что запрос не обработан. Для этого параметру следует присвоить значение *false*.

1.2. Параметр REQUEST

Параметр Request является экземпляром класса TWebRequest - базового класса для передачи информации Web-приложениям. Основные свойства данного класса:

1. PropertyContent: String - строка параметров, переданная с помощью метода POST. Фактически эта строка содержит тело HTTP-запроса, полученного от клиента;
2. PropertyContentFields: TStrings - «разобранная» строка параметров, переданная с помощью метода POST. Каждый элемент коллекции ContentFields представляет собой строку, соответствующую одному элементу управления, расположенному в теге FORM, и представляет собой имя управляющего элемента и его значение, разделенные знаком равенства;
3. PropertyQuery: String - строка параметров, переданная клиенту с помощью метода GET;
4. PropertyQueryFields: TStrings - «разобранная» строка параметров. переданная с помощью метода GET;
5. PropertyRemoteAddr: String - IP-адрес клиента, пославшего запрос;
6. PropertyRemoteHost: String – доменное имя клиента, пославшего запрос;
7. PropertyMethod: String – метод, используемый для передачи данных

серверу.

1.3. Параметр RESPONSE

Параметр Response является экземпляр класса TWebResponse - базового класса, предназначенного для формирования ответа на HTTP-запрос. Основные свойства данного класса:

1. PropertyContentType: String тип данных, содержащих в теле ответа;
2. PropertyContentLength: Integer – число символов, содержащихся в теле ответа;
3. Properly Content: String - содержимоетелаответа;
4. PropertyContentStream: TStream - определяет объект TStream, который будет передан клиенту. Данное свойство обычно используется для передачи клиенту бинарных файлов.

1.4. События TWEBMODULE

Для выполнения обработки запросов и изменения содержимого ответа можно использовать действия, задаваемые в свойстве Actions, а также события самого компонента TWebModule. В этом компоненте предусмотрена возможность обработки четырех событий: AfterDispatch, BeforeDispatch, OnCreate и OnDestroy.

Описание данных событий:

1. AfterDispatch: THTTPMethodEvent - вызывается после того, как HTTP-ответ был успешно сформирован (в обработчике OnActions какого-либо действия из списка Actions), но еще не передан клиенту;
2. BeforeDispatch: THTTPMethodEvent - вызывается перед тем, как диспетчер устанавливает соответствие HTTP-запроса с каким-либо действием. Может использоваться для предварительной обработки HTTP-запроса;
3. OnCreate: TNotifyEvent - вызывается при создании экземпляра TWebModule. Данное событие обычно вызывается для инициализации переменных и, объектов, содержащихся в приложении. Например, сайт модуль содержит базу данных, в обработчике этого события можно выполнить подключение базы данных;
4. OnDestroy: TNotifyEvent - вызывается перед уничтожением TWebModule. Обычно используется для освобождения объектов, созданных динамически. При использовании баз данных в обработчике данного события можно, например, разрывать соединение с базой данных.

2. ЗАДАНИЕ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ

2.1. Создание WEB-приложения с помощью компонента Delphi - WEBMODULE

Создайте Web-приложение, выполняющее проверку знаний таблицы умножения. Серверное приложение выполняет два действия:

1. первое - анализирует ответ клиента;
2. второе - создает основной документ HTML.

Для создания данного приложения выполните следующие действия.

Этап 1: *создание нового Web-приложения.*

1) Выберите в главном меню Delphi команду *File/New*, затем выберите е открывшемся окне диалога значок WebServerApplication и щелкните на кнопке *ОК*.

2) В открывшемся окне диалога выберите необходимый тип приложения. В данном случае следует использовать вариант «CGIStand-aloneexecutable». В результате будет создано новое CGI-приложение, содержащее компонент TWebModule.

Этап 2: *задание действий, выполняющих обработку запросов клиентов.*

1) Выберите в инспекторе объектов компонент TWebModule и щелкните на кнопке с многоточием в поле ввода свойства Action этого компонента. При этом откроется окно редактора действий (рис. 1).

2) Создайте два новых действия. Для этого следует воспользоваться либо кнопками на панели инструментов редактора действий (кнопка AddNew), либо командой Add контекстного меню редактора действий, которое открывается при щелчке правой кнопкой «мыши» в окне редактора действий.

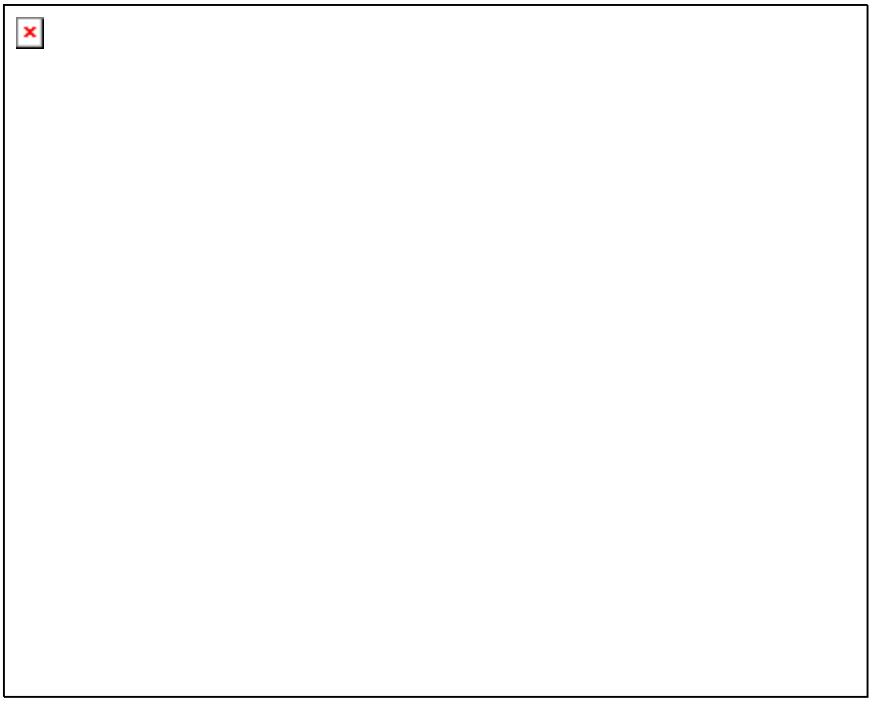

Рис. 1

3) Запрограммируйте первое действие. Для этого в окне редактора действий (рис. 1) выберите первое действие. Установите свойство PahtInfo равным «/1». Это приведет к тому, что данное действие будет обрабатывать HTTP-запрос только в том случае, если при вызове CGI-программы после ее имени будет указана строка «/1», например: http://nts/tmp/mult.exe/l. Значение всех остальных параметров оставьте без изменения.

4) Задайте обработчик запроса первого действия. Для этого выберите в инспекторе объектов вкладку Events и дважды щелкните «мышью» в поле ввода события OnAction. Введитеследующийпрограммныйкод:

```
procedure TWebModule1 .WebModule1 WebAclionItem1 Action(Sender: TObject;
Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
VAR
Num1, num2, result: integer;
begin
num1 :StrToInt(Request.ContentFields.Values[‘num1']);
num2 :StrToInt(Request.ContentFields.Values['num2']);
result:= num1*num2;
If Request.ContentFields.Values['resuIt']=IntToStr(result)
thenResponse.Content :=
'<b>Поздравляю!!! </b><p> Вы прекрасно знаете таблицу умножения!'
elseResponse.Content :=
'Вы немного забыли таблицу умножения <p>'+
'<a href="http://myself/temp/mult.exe">'+
'<p> Попробуйте еще раз </a></p>';
Handled :=True;
end;.
```

5) Запрограммируйте второе действие. Для второго действия свойство PathInfo оставьте пустым, а свойство Default установите равным True. Это означает, что по умолчанию будет вызываться данное действие. В обработчик OnAction второго действия запишите следующий программный код:

```
procedure TWebModule1.WebModule1 WebActionItem2Action(Sender:
TObject;
Request: TWebRequest; Response: TWebResponse: var Handled: Boolean);
Var
num1, num2: Word;
begin
Randomize;
Num1 := Random(8)+l;
num2 :=Random(8)+1;
Response.Content:='<html><head><Title>Таблицаумножения'+
'</title></head><body>'+
‘<h1>Проверьте свое знание таблицы умножения</h1>';
Response.Content:= Response.Content+
'Запишите в окне, чему равно произведение,‘+
'и нажмите кнопку "Ответ"';
Response.Content := Response.Content+
‘<form method="POST” ‘+
'action="http://myself/temp/mult.exe/1 "<table>'+
'<table><tr><td><input type="text" name="num1" size=" 1"'+
¦value='+IntToStr(num1)+ '></td><td> x'+
```

```
'<td><input type="text" name="num2" size="1" value='+
IntToStr(num2)+'></td><td>='+
'<td><input type="text" name="result" size="3"></td>'+
'<td><input type=”submit" name="post" value="Ответ">'+
'</td></tr><table></form>'+
‘<form method="POST"’+
'action="http://myself/temp/mult.exe">'+
'<p><input type="submit" value="Новыечисла" ‘+
'name="New"></p></form></body></html>';
Handled :=True;
end;.
```

6) В результате вызова второго действия (действия по умолчанию) на экране появится HTML-страница, изображенная на рис. 2.

7) При нажатии на кнопку «Ответ» активизируется второе действие. Если пользователь ответил на вопрос правильно, на экране появится HTML-страница, изображенная на рис. 3, если - неправильно, то HTML-сграница, изображенная на рис. 4.

Этап 3: *компиляция и выполнение приложения*

1) Откомпилируйте приложение и запишите его в каталог Web-сервера, допускающий выполнение приложений.

2) Запустите InternetExplorer и в поле «Адрес» укажите URL данного приложение в виде: http://nts/tmp/mult.exe. Так как после имени программы не указано действие, то будет выполняться действие по умолчанию, а именно действие два.

ПРИМЕЧАНИЕ

При повторном открытии созданного приложения компонент TWebModule автоматически не загружается. Для его открытия надо выполнить пункт главного меню *File/Open*и в открывшемся диалоговом не выбрать созданный вами для этого приложения файл с расширением *.pas.

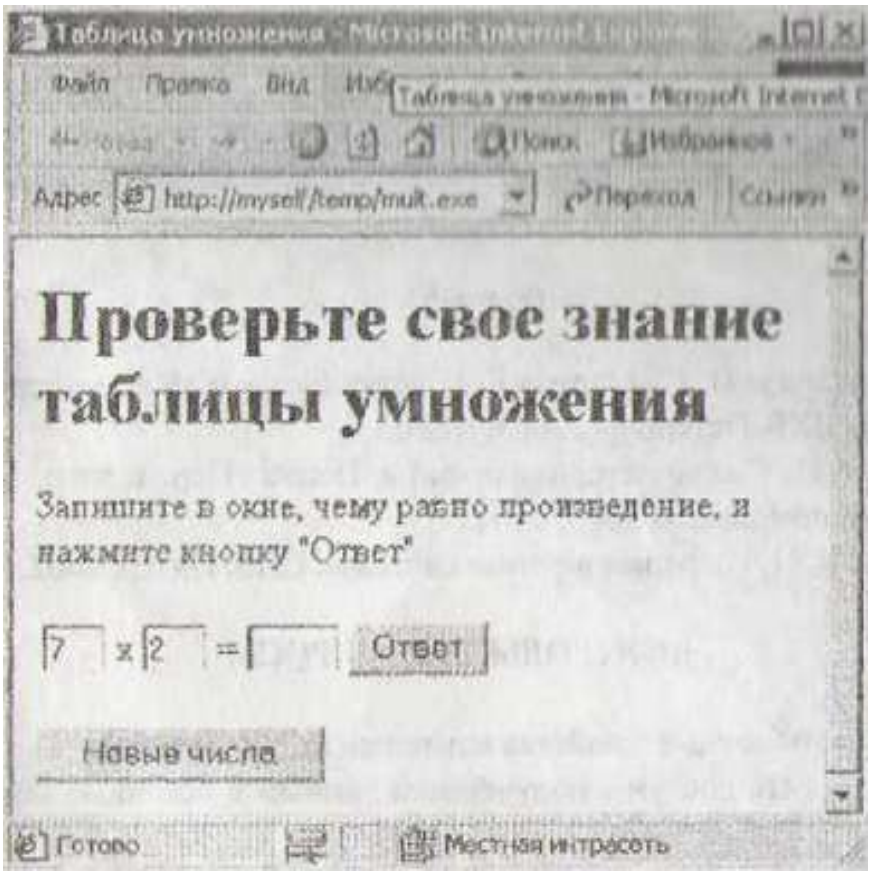

Рис. 2

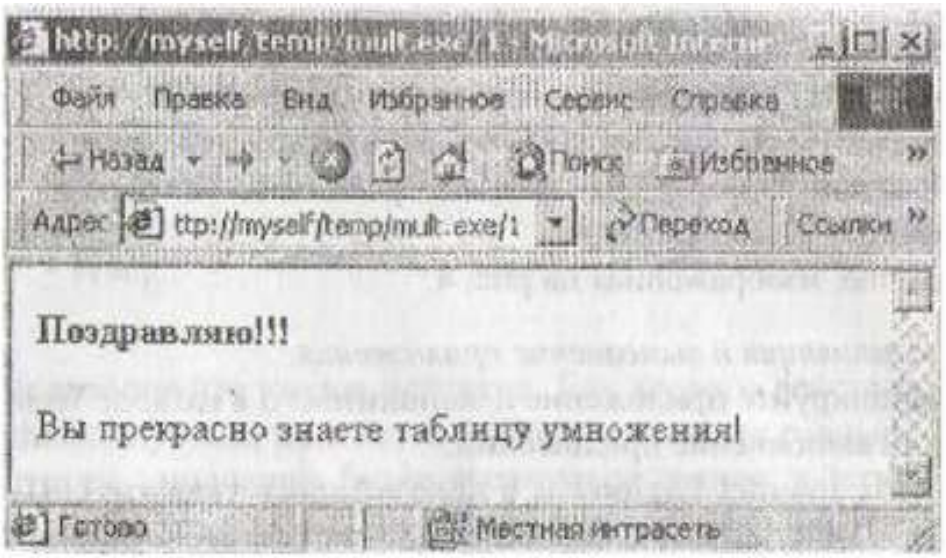

Рис. 3

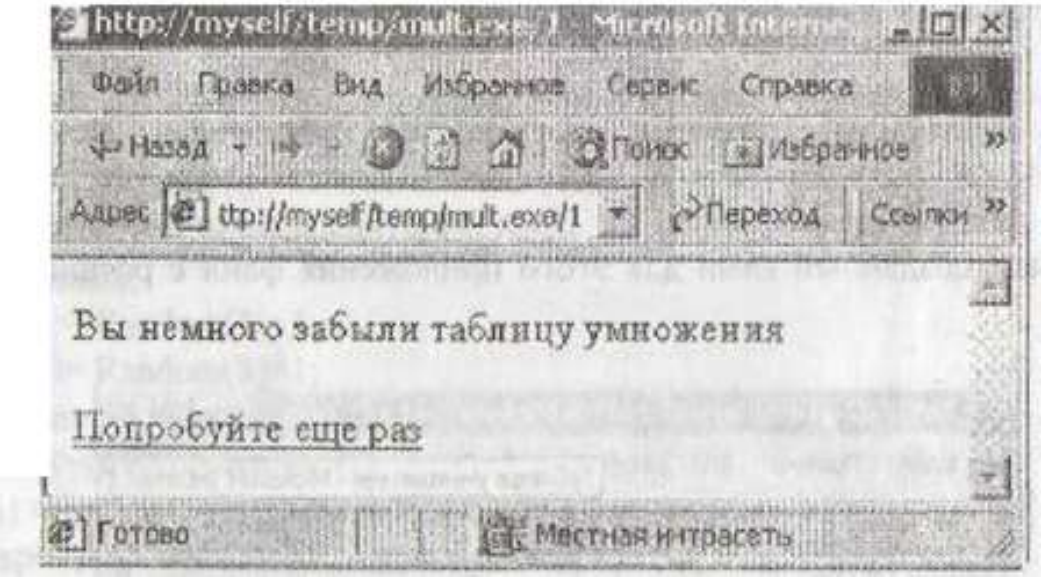

Рис. 4

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные свойства компонента TWebModule.

2. Как получить доступ к полученным данным с помощью компонента TWebModule?

3. Как передать данные с помощью компонента TWebModule?

4. Как задать действие по умолчанию?

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

Сетевые технологии и системное администрирование

Направление подготовки 09.03.03 Прикладная информатика
Прикладная информатика в компьютерном дизайне
Форма подготовки очная

г. Владивосток

2018

1. Контрольно-измерительные материалы

Формы контроля	Модуль 1		Модуль 2					
	ТК1		ТК5-8		РК2		РК3	
	min	max	min	max	min	max	min	max
Основное задание	3	5	3	5	6	10		
Тестирование							3	5
Баллы точки контроля	**6**	**10**	**12**	**20**	**6**	**10**	**6**	**10**
Накопление баллов	**6**	**10**	**36**	**70**	**42**	**70**	**48**	**80**

Формы контроля	Модуль 3				Модуль 4				Экзамен	
	ТК10-12		РК4		ТК13-15		РК5			
	min	max	min	max	min	max	min	max	min	max
Основное задание	6	10			6	10				
Тестирование			6	10			6	10	12	20
Баллы точки контроля	**18**	**30**	**6**	**10**	**18**	**30**	**6**	**10**	**12**	**20**
Накопление баллов	**18**	**30**	**24**	**40**	**42**	**70**	**49**	**80**	**60**	**100**

Порядок начисления баллов

Формы контроля	Границы	Порядок начисления баллов

	min	max	
Задание на практическое занятие	**3(6)**	**5(10)**	Задание выполнено частично или с грубыми ошибками = **3(6)**, задание выполнено с помарками или с опозданием = **4(8),** задание выполнено в полном объёме и в срок = **5(10)**
Тестирование	**6**	**10**	«3» = **6**, «4» = **8**, «5» = **10**

Сокращения и обозначения:

ТК – текущий контроль, РК – рубежный контроль, «Х» - традиционная оценка.

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Вопросы к зачету

1. История развития сетей ЭВМ.
2. Классификация сетей.
3. Характеристики сетей.
4. Базовые топологии сетей.
5. Концепции архитектуры открытых сетей.
6. Семиуровневая сетевая архитектура. Уровни и протоколы.
7. Характеристика уровней семиуровневой модели.
8. Сетезависимые и сетенезависимые уровни.
9. Методы передачи данных на физическом уровне.

10. Потенциальный код без возвращения к нулю.
11. Метод биполярного кодирования с альтернативной инверсией.
12. Потенциальный код с инверсией при единице.
13. Биполярный импульсный код.
14. Манчестерский код.
15. Потенциальный код 2B1Q.
16. Логическое кодирование.
17. Архитектура локальных сетей.
18. Технология Ethernet.
19. Формат кадра Ethernet.
20. Стандарты IEEE на 10 Мбит/с.
21. Технология Fast Ethernet.
22. Сетевые адаптеры.
23. Концентраторы.
24. Ограничения сети, построенной на общей разделяемой среде.
25. Структуризация с помощью мостов и коммутаторов.
26. Управление обменом информацией в глобальных сетях.
27. Коммутация каналов.
28. Коммутация каналов на основе частотного мультиплексирования.
29. Коммутация каналов на основе разделения времени.
30. Общие свойства сетей с коммутацией каналов.
31. Коммутация пакетов.
32. Структура стека TSP/IP. Краткая характеристика протоколов.
33. Адресация в IP- сетях.
34. Три основных класса IP-адресов.
35. Использование масок в IP-адресации.
36. Протокол IP.
37. Маршрутизация. Виды и алгоритмы маршрутизации.
38. Локальные вычислительные сети.
39. Глобальные вычислительные сети.

40. Модель взаимодействия открытых систем.
41. Методы кодирования информации.
42. Методы доступа.
43. Способы контроля.
44. Аппаратура сетей.
45. Управление обменом информацией.
46. Структура глобальной сети.
47. Глобальная сеть на основе выделенной линии.
48. Глобальная сеть с коммутацией каналов.
49. Коммутация пакетов.
50. Стек протоколов TCP/IP.
51. Протоколы сетевого уровня.
52. Маршрутизация.

тематика и перечень курсовых работ и рефератов

Рефераты

Тема 1. Метод доступа CSMA/CA.

1. Беспроводные сети WI-FI.История. Классификация.
2. Метод доступа CSMA/CA. Суть метода. Недостатки метода.
3. Возможные пути устранения недостатков метода.

Тема 2. Разработка драйверов.

1. Понятие драйвера. Классификация драйверов.
2. Структура драйвера. Используемые функции.
3. Коррекция методов доступа CSMA/CD (CSMA/CA).

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕЙ АТТЕСТАЦИИ

Примеры тестовых заданий

МОДУЛЬ 1. Основы теории вычислительных сетей

1. Сетью называют группу …

А) компьютеров;

Б) процессоров;

В) регистров.

2. Компьютерные сети бывают …

А) телефонными;

Б) локальными;

В) глобальными;

Г) электрическими;

Д) силовыми.

3. Сокращенное обозначение локальных сетей :

А) IBM;

Б) GAN;

В) LAN;

Г) MAN.

4. Сокращенное обозначение глобальных сетей:

А) IBM;

Б) GAN;

В) LAN;

Г) MAN.

5. Первые локальные сети появились в …. 20-го века

А) 30-е годы;

Б) 40-е годы;

В) 70-е годы;

Г) 80-е годы.

6. Стандартные технологии локальных сетей были созданы в …. 20-го века

А) 30-е годы;

Б) 40-е годы;

В) 70-е годы;

Г) 80-е годы;

7. Базовые топологии сетей

А) треугольник;

Б) звезда;

В) кольцо;

Г) шина.

8.Модель OSI описывает ….

А) физические процессы в полупроводниках;

Б) организацию приоритетного доступа в сетях

В) взаимосвязи открытых систем;

Г) структуру IP- пакета.

9. Модель OSI разработана

А) SAMSUNG

Б) IBM

В) ISO

Г) INTEL

10 . Модель OSI разработана в ….. годы 20-го века

А) 20-е

Б) 30-е

В) 70-е

Г) 80-е

11. Модель OSI имеет ….архитектуру

А) пятиуровневую

Б) семиуровневую

В) четырёхуровневую

Г) восьмиуровневую

12. Кодирование передаваемой по сети информации производится на….уровне

А) прикладном

Б) сеансовом

В) физическом

Г) сетевом

13. Проверка среды передачи на предмет занятости производится на ….уровне

А) канальном

Б) представительном

В) физическом

Г) сетевом

14. Передача информации по каналу связи производится на….уровне

А) канальном

Б) представительном

В) физическом

Г) сетевом

15. Обнаружение ошибок производится на….уровне

А) канальном

Б) представительном

В) физическом

Г) сетевом