Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Амурский государственный университет»

Кафедра информационных и управляющих систем

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

«Сети ЭВМ и телекоммуникации»

Основной образовательной программы по направлению подготовки

230102.65 – Автоматизированные системы обработки и управления информацией

Благовещенск 2012

УМКД разработан канд. техн. наук, доцентом Т.А. Галаган

Рассмотрен и рекомендован на заседании кафедры

Протокол заседания кафедры от «__» _____201_ г. № _____

Зав. кафедрой _____ / А.В. Бушманов /
(подпись) (И.О. Фамилия)

УТВЕРЖДЕН:

Протокол заседания УМСС _____
(указывается название специальности (направления подготовки)

от «____» _____ 201__ г. № _____

Председатель УМСС _____ /_____ /
(подпись) (И.О. Фамилия)

РАБОЧАЯ ПРОГРАММА

1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

Цель дисциплины – обучение студентов принципам построения, назначения, функционирования и практического использования компьютерных сетей как эффективного средства управления процессами обработки данных в современных ЭВМ.

Задачи дисциплины:

Изучение требований и спецификаций отдельных компонентов сетей ЭВМ и возможностей их технических средств;

Изучение этапов проектирования архитектуры компонентов аппаратно-программных комплексов.

По завершению изучения курса «Сети ЭВМ и телекоммуникации» студент должен:

знать

назначение и функции компонентов сетей ЭВМ, алгоритмы их функционирования;

современные средства вычислительной техники, коммуникаций и связи;

базовые принципы построения и способы проектирования компьютерных сетей и телекоммуникаций;

уметь использовать современные методы, средства и технологии разработки;

владеть такими понятиями, как модель взаимодействия открытых систем, метод доступа к среде передачи, коммуникационные устройства, среда передачи, протокол, спецификация,

Преподавание курса «Сети ЭВМ и телекоммуникации» связано с изучением курсов государственного образовательного стандарта «Информатика», «Операционные системы» и является основой для изучения дальнейших дисциплин, использующих ЭВМ и проектирование программно-аппаратных комплексов.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВПО

Программа курса " Сети ЭВМ и телекоммуникации " составлена в соответствие с требованиями с требованиями государственного образовательного стандарта направления подготовки специалиста – Информатика и вычислительная техника, специализации – Интегрированные автоматизированные системы, блок общеобразовательных дисциплин ОПД.Ф.11:

Классификация информационно-вычислительных сетей. Способы коммутации. Сети одноранговые и "клиент-серер". Уровни и протоколы. Эталонная модель взаимосвязи открытых систем. Аналоговые каналы передачи данных. Способы модуляции. Модемы. Цифровые каналы передачи данных. Разделение каналов по времени и частоте. Характеристики проводных линий связи. Кодирование информации. Способы контроля правильности передачи информации. Локальные вычислительные сети. Методы доступа. Множественный доступ с контролем несущей и обнаружением конфликтов. Разновидности сетей Ethernet. Маркерные методы доступа. Сети Token Ring и FDDI. Высокоскоростные локальные сети. Организация корпоративных сетей. Алгоритмы маршрутизации. Протоколы TCP/IP. Протоколы управления. Сетевые операционные системы. Web-технологии. Языки и средства создания Web-приложений.

3. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Общая трудоемкость дисциплины составляет 162 часа.

№ п/п	Раздел дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость в часах			Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)
				Лекц.	Лабораторные работы	Сам.	
1	Классификация, требования, основные компоненты сетей ЭВМ	6	1, 2	4		4	
2	Принципы взаимодействия компьютеров в сети	6	3, 4	4		6	
3	Модель OSI	6	5 – 7	6		6	Тест №1
4	Типы и характеристики линий связи	6	8 , 9	4		6	
5	Базовые стандарты и технологии локальных сетей	6	10 - 15	12		6	Тест №2
6	Мосты и коммутаторы в сетях	6	16, 17	3		4	
7	Глобальные сети	6	17, 18	3		6	
8	Языки и средства создания Web-приложений.	6	1 – 14		28	32	Подготовка отчетов по лаб. работам
9	Проектирование ЛВС	6	15 -18		8	20	Подготовка отчетов по лаб. работам

4. СОДЕРЖАНИЕ РАЗДЕЛОВ И ТЕМ ДИСЦИПЛИНЫ

4.1 Лекции 36 часов

4.1.1 Классификация информационно-вычислительных сетей. Требования, предъявляемые к современным сетям. Одноранговые сети и сети «клиент-сервер». Основные характеристики сети и способы их определения. Способы коммутации.

4.1.2 Принципы взаимодействия компьютеров в сети. Функциональные группы устройств в сети. Топологии подключения. Логическая и физическая структуризация сети.

4.1.3. Организации по стандартизации компьютерных сетей. Понятие «открытая система». Эталонная модель взаимосвязи открытых систем. Стеки коммуникационных протоколов. Уровни модели OSI. Протоколы TCP/IP.

4.1.4. Типы и характеристики линий связи. Стандарты и характеристики линий связи. Методы передачи данных. Аналоговые каналы передачи данных. Способы модуляции. Модемы. Цифровые каналы передачи данных. Кодирование информации. Способы контроля передачи данных.

4.1.5. Базовые стандарты и технологии локальных сетей: Ethernet, Token Ring, FDDI. Методы доступа к среде передачи: множественный метод с контролем несущей и обнаружением коллизий, маркерное кольцо. Высокоскоростные локальные сети.

4.1.6. Построение ЛВС с помощью мостов и коммутаторов. Алгоритмы маршрути-

зации.

4.1.7.Глобальные сети. Организация корпоративных сетей. Протоколы управления. Сетевые операционные системы.

4.2. ЛАБОРАТОРНЫЕ РАБОТЫ (36 часов)

4.2.1. Основы программирования на JavaScript. (14 часов)
4.2.2. Создание сетевых приложений средствами Builder C++ (14 часов)
4.2.3. Методика расчета конфигурации сети Ethernet. (8 часов)

5 САМОСТОЯТЕЛЬНАЯ РАБОТА

№ раздела (темы) дисциплины	Форма (вид) самостоятельной работы	Трудоемкость в часах
Классификация, требования, основные компоненты сетей ЭВМ	Изучение теоретических материалов	4
Принципы взаимодействия компьютеров в сети	Изучение теоретических материалов	6
Модель OSI	Изучение теоретических материалов	6
Типы и характеристики линий связи	Изучение теоретических материалов	6
Базовые стандарты и технологии локальных сете	Изучение теоретических материалов	6
Мосты и коммутаторы в сетях	Изучение теоретических материалов	4
Глобальные сети	Изучение теоретических материалов	6
Языки и средства создания Web-приложений.	Работа над созданием программ и отчетов к лабораторным работам	32
Проектирование ЛВС	Подготовка отчетов по лабораторным работам	20

6 ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

К образовательным технологиям, используемым в преподавании данной дисциплины, относятся лекции и лабораторные работы.

В изложении лекционного материала наряду с традиционной лекцией используются такие неимитационные методы обучения, как:

проблемная лекция, начинающаяся с постановки проблемы, которую необходимо решить в ходе изложения материала,

лекция-визуализация, учащая студента преобразовывать устную и письменную информацию к визуальной форме в виде схем, рисунков, чертежей,

лекция с заранее запланированными ошибками, которые студенты должны обнаружить самостоятельно в конце лекции.

На лекциях используются информационные технологии – презентации. Удельный вес занятий, проводимых в интерактивных формах должен составлять не менее 20% аудиторных занятий.

Лабораторные работы проводятся в компьютерных классах и предназначены для решения прикладных задач с использованием современных инструментальных средств.

При проведении лабораторных работ используются неигровые имитационные методы обучения:

контекстное обучение, направленное на решение профессиональных задач,

работа в команде – совместная деятельность студентов в группе, направленная на решение общей задачи с разделением ответственности и полномочий.

При оценивании результатов обучения используется балльно-рейтинговая технология.

7 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ И УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

7.1. Экзаменационные вопросы

1. Классификация компьютерных сетей
2. Структура взаимодействия устройств в сети
3. Модель взаимосвязи открытых систем (история создания, основные идеи)
4. Функции канального уровня модели OSI
5. Функции сеансового уровня модели OSI
6. Функции канального уровня модели OSI
7. Транспортный уровень модели OSI
8. Иерархическая сеть
9. Сети клиент-сервер
10. Одноранговые сети
11. Методы передачи данных
12. Аналоговые каналы передачи данных
13. Цифровые каналы передачи данных
14. Модемы. Способы модуляции
15. Кодирование информации
16. Способы контроля передачи информации
17. Функциональные группы устройств сети (узел, рабочая станция, сервер (группы серверов), повторитель, мост, мультиплексор, маршрутизатор, шлюз)
18. Среда передачи данных (виды, характеристики, стандарты)
19. Топологии подключения устройств в сети
20. Структура стандартов IEEE
21. Технология Ethernet. Метод множественного доступа с контролем несущей и обнаружением коллизий
22. Высокоскоростные варианты Ethernet
23. Сети Token Ring. Метод доступа, способы подключения устройст.
24. Распределенный волоконно-оптический интерфейс передачи данных (FDDI)
25. Основные направления развития современных сетевых операционных систем
26. Протоколы TCP/IP
27. Мосты в локальных сетях. Алгоритмы работы мостов
28. Маршрутизаторы. Алгоритмы маршрутизации.
29. Глобальные сети. Организация корпоративных сетей.
30. Протоколы управления.
31. Сетевые операционные системы.

7.2. Примеры вопросов теста

1. ***Выделите из ниже перечисленного функции, выполняемые на транспортном уровне эталонной модели взаимодействия открытых систем (OSI):***

   а) определение начала и окончания сеанса связи;

   б) контроль последовательности передачи данных;

в) определение маршрутизации в сети и связь между сетями;
г) установка соответствия между транспортными (логическими) и сетевыми адресами абонентов;
д) определение метода доступа к среде передачи данных;
е) определение времени, длительности и режима сеанса связи;
ж) определение логической топологии сети передачи данных;
з) обнаружение и обработка ошибок передачи данных;
и) обеспечение независимости высших уровней от используемой для передачи информации физической среды;
к) определение точек синхронизации для промежуточного контроля и восстановления при передаче данных;
л) определение физической адресации.

2. Какую функцию выполняет повторитель?

а) организация обмена данными между сетевыми объектами, использующими различные протоколы обмена данными;
б) расширение сети подключением дополнительных сегментов кабеля;
в) объединение несколько сегментов, так что передача данных между станциями внутри одного сегмента не будет влиять на передачу данных в других сегментах;
г) соединение сетей разного типа, использующие одну сетевую операционную систему или протокол обмена данными.

3. Какая из перечисленных ситуаций называется коллизией?

а) ситуация, когда станция, желающая передать пакет, обнаруживает, что в данный момент другая станция уже заняла передающую среду;
б) ситуация, когда две рабочие станции одновременно передают данные в разделяемую передающую среду.

4. Поставьте соответствие между типом коммуникационного устройства и уровнями модели OSI, на котором оно функционирует?

1) повторитель	а) физический, канальный, сетевой, транспортный
2) мост	б) физический, канальный
3) маршрутизатор	в) физический, канальный
4) шлюз	г) над уровнями модели OSI

5. Какой метод доступа поддерживает технология Gigabit Ethernet?

а) множественный доступа с контролем несущей и обнаружением коллизий
б) маркерное кольцо
в) раннее освобождение маркера
г) маркерная шина

6. Стек коммуникационных протоколов _____ является лидером среди протоколов вычислительных сетей и поддерживает все популярные стандарты физического и канального уровня локальных сетей.

7. В каких сетях для работы мостов не используется алгоритм с маршрутизацией от источника?

а) Ethernet
б) Token Ring
в) FDDI

8. При каком способе передачи обмен информацией между узлами сети выполняется устройствами поочередно?

а) симплексной б) дуплесной
в) полудуплексной г) полусимплексной

9. Широкополосные модемы работают только по _____ окончаниям в дуплексном _____ режиме.

10. Поставьте соответствие между названиями стандартов и технологиями, которые они поддерживают.

а) Ethernet 1) 802.1
б) Token Ring 2) 802.2
в) общие определения локальных сетей 3) 802.3

8 УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

ОСНОВНАЯ ЛИТЕРАТУРА

1 Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы. Учеб.: рек. Мин. образования РФ / В.Г. Олифер, Н.А. Олифер. – СПб: Питер, 2005, 2009, 2010. – 994с.

2 Таненбаум, Э. Компьютерные сети. – СПб: Питер, 2006, 2009, 2010. – 992с.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

1 Олифер, В.Г Сетевые операционные системы. Учеб.:доп. Мин. образования РФ /В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2009. – 669 с.

2 Якубайтис, Э.А. Информационные сети и системы. Справочная книга. М: "Финансы и статистика" , 1996. – 368 с

ИНТЕРНЕТ-РЕСУРСЫ

	Наименование ресурса	Характеристика
1	http://www.intuit.ru	ИНТУИТ - сайт, который предоставляет возможность дистанционного обучения по нескольким образовательным программам, касающимся, в основном, информационных технологий. Содержит несколько сотен открытых образовательных курсов.
2	http://ru.wikipedia.org	Википендия – свободная общедоступная мультиязычная универсальная интернет-энциклопедия. Поиск по статьям, написанным на русском языке. Избранные статьи, ссылки на тематические порталы и родственные проекты.

ПЕРИОДИЧЕСКИЕ ИЗДАНИЯ

Журналы «Информационные технологии и вычислительные системы», «Компьютер-Пресс», «Программные продукты и системы»

9 МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Мультимедийная лекционная аудитория (331)
Лаборатория микропроцессорной техники и компьютерных сетей (333)

10 РЕЙТИНГОВАЯ ОЦЕНКА ЗНАНИЙ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

Балльная структура оценки за семестр

Семестровый модуль дисциплины					
Учебные модули	Виды контроля	Сроки	Макс.	По-	Макс.

			выполнения (недели)	кол-во баллов	сещение занятий, активность	кол-во баллов за уч. модуль
1	Классификация, требования, основные компоненты сетей ЭВМ	Тест №1	1 - 7	10	3	13
2	Принципы взаимодействия компьютеров в сети					
3	Модель OSI					
4	Типы и характеристики линий связи	Тест №2	8 – 18	10	3	13
5	Базовые стандарты и технологии локальных сете					
6	Мосты и коммутаторы в сетях					
7	Глобальные сети					
8	Языки и средства создания Web-приложений.	Отчеты по лабораторным работам	1 – 14	24	3	27
9	Проектирование ЛВС	Отчеты по лабораторным работам	15 – 18	6	1	7
	Экзамен					40
	Итого					100

ПЛАН-КОНСПЕКТ ЛЕКЦИЙ

Лекция 1

Тема Понятие и классификация вычислительных сетей.

План лекции

1. Определение «вычислительная сеть»
2. Классификация по территориальному признаку
3. Классификация по масштабу
4. Одноранговые (равноправные) сети и сети клиент-сервер

Цель - организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных понятий и классификаций современных вычислительных сетей.

Ключевые вопросы – признаки классификации сетей, особенности каждого типа

Вычислительной сетью называют совокупность компьютеров, соединенную линиями связи и коммуникационными устройствами. Все оборудование сети функционирует под управлением системного и прикладного программного обеспечения.

Информационно-вычислительные системы принято делить на три основных типа по территориальному признаку: LAN – *локальная сеть,* MAN – городская или региональная сеть, WAN – глобальная сеть.

Еще одним популярным способом классификации сетей являет их классификация по масштабу: локальные *сети рабочих групп,* локальные *сети отделов, сети кампусов, корпоративные сети*.

В зависимости от способа организации обработки данных и взаимодействия пользователей выделяют два типа сетей: иерархические сети; сети клиент-сервер;

В иерархических системах задачи хранение данных; организация доступа пользователей или программ к данным; обработка данных; передача данных или результатов обработки данных пользователям или программам; выполняются аппаратными и программными средствами одного основного компьютера. В архитектуре клиент/сервер часть этих задач решает сервер, а часть компьютер или программа, выступающая в качестве клиента. Сервер – это устройство или компьютер, выполняющий обработку поступившего от клиента запроса.

По организации взаимодействия принято выделять два типа систем, использующих метод клиент-сервер: равноправная сеть; сеть с выделенным сервером.

Равноправная сеть – сеть, в которой нет единого центра управления взаимодействием рабочих станций, нет единого устройства хранения данных. Операционная система такой сети распределена по всем рабочим станциям, поэтому каждая рабочая станция одновременно может выполнять функции как сервера, т.е. обслуживать запросы от других рабочих станций, так и клиента - направляет свои запросы другим рабочим станциям.

В сети с выделенным сервером один из компьютеров (сервер сети) выполняет функции хранения данных общего пользования, организации взаимодействия между рабочими станциями, выполнения сервисных услуг. На таком компьютере, как правило, выполняется сетевая операционная система, и все разделяемые устройства подключаются непосредственно к нему.

Достоинства сети с выделенным сервером: выше скорость обработки данных; надежная система зашиты информации и обеспечения секретности; простота в управлении по сравнению с равноправными сетями. Недостатки: быстродействие и надежность сети зависят от компьютера, используемого в качестве сервера сети; сеть с выделенным сервером менее гибкая по сравнению с равноправной.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 2

Тема Основные характеристики и требования современной вычислительной сети.

План лекции

1. Требования к современным сетям
2. Основные показатели сети
3. Способы определения характеристик
4. Способы коммутации

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных характеристик современных вычислительных сетей, овладение специальной терминологией.

Ключевые вопросы – производительность, надежность, управляемость, совместимость, интегрируемость, прозрачность, коммутация сообщений, каналов, пакетов.

Вычислительная сеть создается для обеспечения потенциального доступа к любому ресурсу сети для любого пользователя сети. Качество доступа к ресурсу может быть описано многими показателями, выбор которых зависит от задач, стоящих перед сетью.

Производительность вычислительной сети может быть оценена с разных позиций. С точки зрения пользователя, важным числовым показателем производительности сети является время реакции системы, т.е. время между моментом возникновения запроса и моментом получения ответа. Пропускная способность определяется количеством информации, переданной через сеть или ее сегмент в единицу времени и измеряемая в битах в секунду. Пропускная способность может быть мгновенной, максимальной и средней.

Задержка передачи – задержка между моментом поступления пакета на вход какого-либо устройства и моментом появления его на выходе.

Надежность работы вычислительной сети определяется надежностью работы всех ее компонентов. Для технических устройств используются такие показатели надежности, как среднее время обработки на отказ, вероятность отказа, интенсивность отказов. Для оценки надежности сложных систем применяют *готовность* или *коэффициент готовности*.

Для оценки надежности передачи пакетов используются показатели вероятности потери пакета при его передаче, либо вероятности доставки пакета. В современных сетях важна и другая сторона надежности – безопасность – способность сети обеспечить защиту информации от несанкционированного доступа. Еще одной характеристикой надежности является *отказоустойчивость.*

Любая вычислительная сеть является развивающимся объектом, и не только в плане модернизации ее элементов, но и в плане ее физического расширения, добавления новых элементов сети (пользователей, компьютеров, служб). Существование таких возможностей, трудоемкость их осуществления входят в понятие *расширяемости.* Другой похожей характеристикой является масштабируемость сети, которая определяет возможность расширения сети без существенного снижения ее производительности.

Прозрачность сети предполагает скрытие (невидимость) особенностей сети от конечного пользователя. Другой важной стороной прозрачности сети является возможность распараллеливания работы между разными элементами сети.

Интегрируемость (или *совместимость)* означает возможность подключения к вычислительной сети разнообразного и разнотипного оборудования, программного обеспечения от разных производителей.

Под коммутацией данных понимают их передачу, при которой канал данных может использоваться попеременно. Различают коммутация сообщений, каналов, пакетов.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 3

Тема Компоненты вычислительной сети и способы их взаимодействия**.**

План

1. Интерфейсы взаимодействие устройств
2. Взаимодействие двух компьютеров и программных компонентов
3. Функциональные группы устройств сети

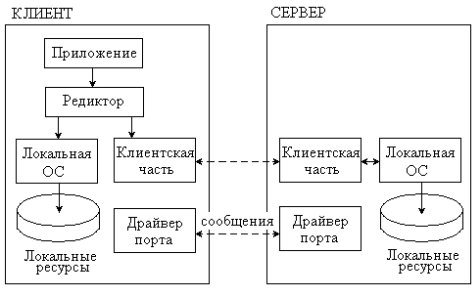
Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных компонентов вычислительных сетей, овладение специальной терминологией.

Ключевые вопросы – интерфейс, протокол, рабочая станция, узел сети, сервер, сегмент сети, коммуникационные устройства.

Изучение сети в целом предполагает знание принципов работы ее отдельных компонентов: компьютеров, коммуникационного оборудования, операционных систем, сетевых приложений.

Для обмена данными между компьютером и периферийным устройством в компьютере предусмотрен внешний интерфейс, то есть набор проводов, соединяющих компьютер и периферийное устройство, а также набор правил обмена информацией по этим проводам (интерфейс или протокол). Примерами интерфейсов, используемых в компьютерах, являются параллельный интерфейс Centronics, предназначенный, как правило, для подключения принтеров, и последовательный интерфейс RS-232C, через который подключаются мышь, модем и много других устройств. Со стороны ПУ интерфейс чаще всего реализуется аппаратным устройством управления, хотя встречаются и программно-управляемые периферийные устройства. Схема взаимодействие программных компонентов при связи двух компьютеров приведена на рис.



Узел – любое устройство, подключенное к сети. Рабочая станция – это персональный компьютер, подключенные к сети, на котором пользователь сети выполняет свою работу. Удаленная рабочая станция – рабочая станция, подключенная к локальной сети через медленную линию связи, отличную от используемой в локальной сети.

Сервер сети – компьютер, подключенный к сети и выполняющий для пользователей сети определенные услуги. Часть сети, в которую не входит устройство расширения, принято называть сегментом сети.

Повторитель (хаб, концентратор) – устройство, позволяющее расширить сеть подключением дополнительных сегментов кабеля.

Мост – устройство, позволяющее объединить несколько сегментов, так что передача данных между станциями внутри одного сегмента не будет влиять на передачу данных в других сегментах. Коммутатор он является своего рода коммуникационным мультипроцессором, так как каждый его порт оснащен специализированным процессором, который обрабатывает кадры по алгоритму моста независимо от процессоров других портов.

Мультиплексор – устройство, позволяющее мультиплексировать данные, приходящие одновременно от различных станций или сегментов и передавать их через передающую среду.

Маршрутизатор – устройство, соединяющее сети разного типа, но использующие одну сетевую операционную систему или протокол обмена данными.

Шлюз – устройство, позволяющее организовать обмен данными между сетевыми объектами, использующими различные протоколы обмена данными.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 4

Тема Топологии подключения. Логическая и физическая структуризация сети.

План

1. Понятие «топология». Логическая и физическая топология.
2. Виды топологии сети
3. Использование структурообразующего оборудования.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

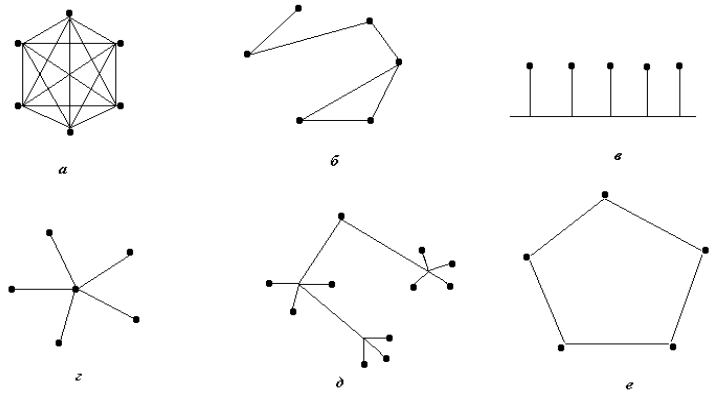
Задача – изучение основных принципов построения вычислительных сетей, овладение специальной терминологией.

Ключевые вопросы – ячеистая, шинная, кольцеобразная, звездообразные, сотовая топологии, методы логической и физической структуризации.

Под топологией вычислительной сети понимается конфигурация графа, вершинам которого соответствуют компьютеры сети (иногда и другое оборудование, например концентраторы), а ребрам – физические связи между ними. Необходимо различать физическую и логическую топологии сети.

Конфигурация физических связей определяется электрическими соединениями компьютеров между собой. Логические связи представляют собой маршруты передачи данных между узлами сети и образуются путем соответствующей настройки коммуникационного оборудования.

При полносвязной топологии (рис. а) каждый компьютер сети связан со всеми остальными. Ячеистая топология получается из полносвязной путем удаления некоторых возможных связей (рис. б). Шина (рис. в) – передаваемая информация может передаваться в обе стороны. В топологии звезда (рис. г) каждый компьютер подключается отдельным кабелем к общему устройству, называемому концентратором. Иногда имеет смысл строить сеть с использованием нескольких концентраторов, иерархически соединенных друг с другом связями типа звезда (рис. д). В сетях с кольцевой конфигурацией (рис. е) данные передаются по кольцу, как правило, в одном направлении. В то время как небольшие сети, как правило, имеют типовую типологию, для крупных сетей характерно наличие произвольных связей между узлами. Их называют сетями смешанной топологии.



Сотовая топология метод разделения географической области на зоны, в каждой из которых обеспечивается обмен информацией между станциями сети, находящимися внутри зоны.

Физическая структуризация сети с помощью концентраторов полезна не только для увеличения расстояния между узлами сети, но и для повышения ее надежности.

Сеть с типовой топологией (шина, кольцо, звезда), в которой все физические сегменты рассматриваются в качестве одной разделяемой среды, оказывается неадекватной структуре информационных потоков в большой сети. Решение проблемы состоит в отказе от идеи использования единой однородной разделяемой среды. Логическая структуризация сети – это процесс разбиения сети на сегменты с локализованным трафиком. Для логической структуризации сети используются такие коммуникационные устройства, как мосты, коммутаторы, маршрутизаторы и шлюзы.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 5

Тема Стандартизация компьютерных сетей.

План

1. Виды стандартов сетей
2. Ведущие организации по стандартизации компьютерных сетей
3. Основные понятия открытых систем

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных принципов построения открытых сетей, овладение специальной терминологией.

Ключевые вопросы – стандарт, открытая система, спецификация, организации по стандартизации сетей.

Работы по стандартизации вычислительных сетей ведутся большим количеством организаций. В зависимости от их статуса различаются и виды стандартов: стандарты отдельных фирм, стандарты специальных комитетов и объединений, национальные стандарты, международные стандарты.

К организациям наиболее активно и успешно занимающихся разработкой стандартов в области вычислительных сетей относятся:

Международная организация по стандартизации (International Organization for Standardization (ISO);

Международный союз электросвязи (International Telecommunications Union, ITU) – организация, являющаяся специализированным органом ООН. Наиболее важную роль играет постоянно действующий в ней Международный комитет по телефонии и телеграфии (Consultative Committee on International Telegraphy and Telephony, CCITT). В 1993 году в результате реорганизации ITU он сменил название на сектор телекоммуникационной стандартизации ITU (ITU Telecommunications Standardization Sector – ITU-T).

Институт инженеров по электротехнике и радиоэлектронике – Institute of Electrical and Electronics Engineers, IEEE – национальная организация США, определяющая сетевые стандарты.

Ассоциация электронной промышленности (Electronic Industries Association, EIA) – промышленно-торговая группа производителей электронного и сетевого оборудования, национальная коммерческая ассоциация США.

Американский институт национальных стандартов (American National Standards Institut, ANSI) – представляет США в ISO.

В широком смысле открытой системой может называться любая система (компьютер, вычислительная сеть, аппаратные или программные продукты), которая построена в соответствии с открытыми спецификациями. Термин «спецификация» означает формализованное описание аппаратных или программных компонент, способ их функционирования, взаимодействия с другими компонентами, условий эксплуатации, ограничений и особых характеристик. Не всякая спецификация является стандартом. Открытые спецификации – опубликованные, общедоступные, соответствующие стандарту и принятые после всестороннего обсуждения.

Для обеспечения обмена данными между компьютерными сетями ISO совместно с CCITT разработала многоуровневый комплект протоколов, известный как эталонная модель взаимосвязи открытых систем (Open System Interconnection – OSI) (1977 г). Одна из основных ее идей – обеспечение относительно легкого и простого обмена информацией при использовании изготовленных разными фирмами аппаратных и программных средств, соответствующих стандартам OSI. Модель OSI касается только одного аспекта открытости – открытости взаимодействия устройств, связанных в вычислительную сеть.

Ярким примером открытой системы является Internet.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 6

Тема. Эталонная модель взаимосвязи открытых систем.

План

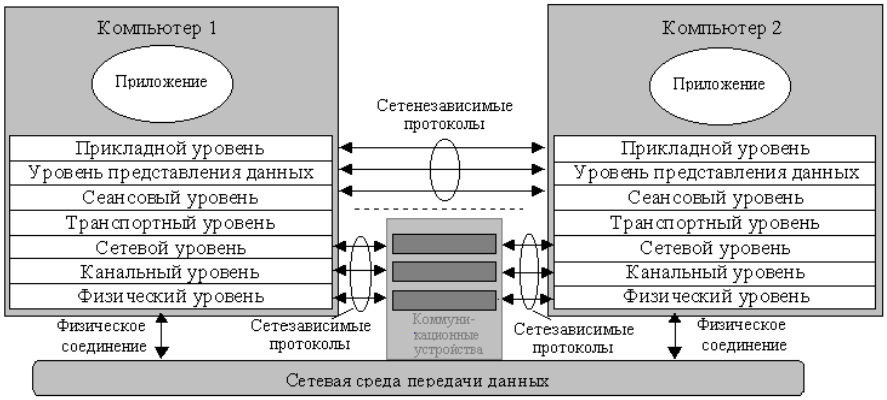
1. История создания модели OSI.
2. Уровни модели OSI.
3. Функции уровней, примеры протоколов

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных принципов построения открытых сетей, овладение специальной терминологией.

Ключевые вопросы – сетезависимые и сетезависимые уровни, единицы передаваемых данных.

Все задачи, которые необходимо решить для организации взаимодействия между объектами информационной системы, разделены на семь отдельных процедур или уровней, представленных на рисунке. Каждый уровень выполняет определенную логическую функцию и обеспечивает определенный набор услуг для расположенного над ним уровня.



Отдельные уровни модели OSI удобно рассматривать как группы программ, предназначенных для выполнения конкретных функций. При обращении некоторого сетевого приложения к прикладному уровню на основании запроса программное обеспечение формирует сообщение стандартного формата, состоящее из заголовка и поля данных. После формирования сообщения прикладной уровень направляет его вниз. Протокол каждого уровня на основании информации, полученной из заголовка вышележащего уровня, выполняет требуемые действия и добавляет к сообщению собственную служебную информацию. Когда сообщение по сети поступает на станцию-адресат, оно принимается физическим уровнем и перемещается вверх. Последовательно каждый уровень анализирует и обрабатывает заголовок своего уровня, выполняя соответствующие данному уровню функции, а затем его удаляет.

Три нижних уровня – физический, канальный, сетевой - являются сетезависимыми, то есть протоколы этих уровней тесно связаны с технической реализацией сети и используемым коммуникационным оборудованием.

Три верхних уровня – прикладной, представительский и сеансовый – ориентированы на приложения и мало зависят от технических особенностей построения сети. На протоколы этих уровней не влияют изменения в топологии сети, замена оборудования или переход на другую сетевую технологию.

Транспортный уровень является промежуточным, он скрывает все детали функционирования нижних уровней от верхних, что позволяет разрабатывать приложения, независящие от технических средств непосредственной транспортировки сообщений.

В стандартах ISO для обозначения единиц данных, с которыми имеют протоколы разных уровней, используется общее название протокольный блок данных (*Protocol Data Unit, PDU*). Для обозначения блоков данных определенных уровней используются специальные названия: кадр (frame), пакет (packet), дейтаграмма (datagram), сегмент (segment).

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 7

Тема Стеки коммуникационных протоколов.

План

1. Виды стандартных сетевых протоколов.
2. Соответствие протоколов уровням модели OSI.
3. Протокол TCP/IP – популярнейший протокол компьютерных сетей.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных принципов построения открытых сетей, изучение стандартов, овладение специальной терминологией.

Ключевые вопросы – протоколы TCP/IP, IPX/SPX, NetBIOS/SMB, DECnet, SNA.

Важнейшим направлением стандартизации в области вычислительных сетей является стандартизация коммуникационных протоколов. Наиболее популярными являются стеки: TCP/IP, IPX/SPX, NetBIOS/SMB, DECnet, SNA, OSI. Все стеки, кроме SNA, на физическом и канальном уровнях используют одни и те же хорошо стандартизированные протоколы Ethernet, Token Ring, FDDI и позволяют использовать во всех сетях одну и ту же аппаратуру. На верхних уровнях все стеки работают по своим собственным протоколам, которые часто не соответствуют рекомендуемому моделью OSI разбиению на уровни. В таблице показано соответствие популярных стеков протоколов модели OSI.

модель OSI	IBM/ Microsoft	TCP/IP	Стек OSI
Прикладной	SMB	telnet FTP SMPT SNPT WWW	X.400 X.500 FTAM
Представительский			Представительный протокол модели OSI
Сеансовый	NetBIOS	TCP	Сеансовый протокол модели OSI
Транспортный			Транспортный протокол модели OSI
Сетевой		IP RIP OSPF	ES-ES IS-IS
Канальный	Ethernet, Token Ring, FDDI, Fast Ethernet, SLIP, X.25, ATM, PPP, 100VG-AnyLAN		
Физический	Коаксиальный кабель, экранированная и неэкранированная витая пара, волоконно-оптический кабель, радиоволны		

Стек TCP/IP – один из самых распространенных стеков транспортных протоколов вычислительных сетей. Основными протоколами стека являются, давшие ему название, протоколы IP и TCP. Они относятся соответственно к сетевому и транспортному уровням. IP обеспечивает продвижение пакета по составной сети, TCP гарантирует надежность его доставки. TCP/IP вобрал в себя популярные протоколы прикладного уровня – протокол пересылки файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMPT. Особенностью технологии TCP/IP является гибкая система адресации, позволяющая достаточно просто включать в интерсеть сети других технологий. Однако мощные функциональные возможности стека TCP/IP требуют для своей реализации высоких вычислительных затрат и предъявляют высокие требования к администрированию IP-сетей.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 8

Тема Линии связи

План

1. Типы линий связи.
2. Характеристики кабельных линий связи и способы их определения.
3. Стандарты кабельных линий связи.
4. Стандарты беспроводной связи.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основных типов кабельной системы сетей, изучение их стандартов, овладение специальной терминологией.

Ключевые вопросы – среда передачи данных, кабельные системы, стандарты физического уровня.

Линия связи в общем случае состоит из физической среды, по которой передаются электрические информационные сигналы, аппаратуры передачи данных и промежуточной аппаратуры. Синонимом термина линия связи является термин канал связи.

В зависимости от среды передачи данных линии связи разделяют на проводные, кабельные (медные, волоконно-оптические), радиоканалы наземной и спутниковой связи.

В компьютерных сетях применяются типы кабеля: на основе скрученных пар медных проводов, коаксиальные с медной жилой и волоконно-оптические. Основные характеристики кабельных линий связи: амплитудно-частотная характеристика, полоса пропускания, затухание, помехоустойчивость, перекрестные наводки на ближнем конце линии, пропускная способность, достоверность передачи данных, удельная стоимость.

Новый стандарт EIA/TIA-568A коаксиальные кабели не описывает, как морально устаревшие.

Медный неэкранированный кабель UTP делится на 5 категорий (Category 1 - Category 5).

Основным стандартом неэкранированной витой пары STP является фирменный стандарт IBM, в котором кабели делятся не на категории, а на типы: Type 1, Type 2, . . . , Type 9.

В зависимости от распределения показателя преломления и от величины диаметра сердечника различают: многомодовое волокно со ступенчатым изменением показателя преломления, многомодовое волокно с плавным изменением показателя преломления, одномодовое волокно.

В спутниковых системах используются антенны СВЧ-диапазона частот для приема радиосигналов от наземных станций и ретрансляции этих сигналов обратно на наземные станции.

LMDS (Local Multipoint Distribution System) - это стандарт сотовых сетей беспроводной передачи информации для фиксированных абонентов.

Радиоканалы WiMAX (Worldwide Interoperability for Microwave Access) в отличие от традиционных технологий радиодоступа работают и на отраженном сигнале, вне прямой видимости базовой станции.

Радиоканалы MMDS (Multichannel Multipoint Distribution System). Эти системы способна обслуживать территорию в радиусе 50—60 км, при этом прямая видимость передатчика оператора является не обязательной.

Стандартом беспроводной связи для локальных сетей является технология Wi-Fi. Wi-Fi обеспечивает подключение в двух режимах: точка-точка (для подключения двух ПК) и инфраструктурное соединение (для подключения несколько ПК к одной точке доступа). Радиоканалы Bluetooht – это технология передачи данных на короткие расстояния (не более 10 м) и может быть использована для создания домашних сетей.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 9

Тема Методы и каналы передачи данных

План

1. Методы передачи данных.
2. Аналоговые каналы передачи данных.
3. Способы модуляции. Модемы.
4. Цифровые каналы передачи данных.
5. Кодирование информации. Способы контроля передачи данных.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение основ передачи данных в компьютерных сетях.

Ключевые вопросы – аналоговые и цифровые каналы, потенциальное и импульсное кодирование, симплексная, дуплексная, полудуплексная передача, методы кодирования.

Так как аппаратура передачи и приема данных работает с данными в дискретном виде (т.е. единицам и нулям данных соответствуют дискретные электрические сигналы), то при их передаче через аналоговый канал требуется преобразование дискретных данных в аналоговые (модуляция).

При приеме таких аналоговых данных необходимо обратное преобразование – демодуляция. Модуляция/демодуляция – процессы преобразования цифровой информации в аналоговые сигналы и наоборот. При модуляции информация представляется синусоидальным сигналом той частоты, которую хорошо передает канал передачи данных. К способам модуляции относятся: амплитудная модуляция; частотная модуляция; фазовая модуляция.

При передаче дискретных сигналов через цифровой канал передачи данных используется кодирование: потенциальное; импульсное. Потенциальное или импульсное кодирование применяется на каналах высокого качества, а модуляция на основе синусоидальных сигналов предпочтительнее в тех случаях, когда канал вносит сильные искажения в передаваемые сигналы.

При обмене данными между узлами вычислительных сетей используются три метода передачи данных: симплексная (однонаправленная) передача (телевидение, радио); полудуплексная (прием/передача информации осуществляется поочередно); дуплексная (двунаправленная), каждый узел одновременно передает и принимает.

коды включают в себя также коды проверки ошибок и другую информацию.

Существует три принципиально различные схемы коммутации в вычислительных сетях: коммутация каналов; коммутация пакетов; коммутация сообщений.

При цифровом кодировании дискретной информации применяют потенциальные и импульсные коды.

В потенциальных кодах для представления логических единиц и нулей используется только значение потенциала сигнала, а его перепады, формирующие законченные импульсы, во внимание не принимаются. Импульсные коды позволяют представить двоичные данные либо импульсами определенной полярности, либо частью импульса - перепадом потенциала определенного направления.

Для обнаружения искажений наиболее популярны методы, основанные на циклических избыточных кодах (CRC), которые выявляют многократные ошибки. Для восстановления кадров используется метод повторной передачи на основе квитанций. Этот метод работает по алгоритму с простоями источника, а также по алгоритму скользящего окна. Для повышения полезной скорости передачи данных в сетях применяется динамическая компрессия данных на основе различных алгоритмов. Коэффициент сжатия зависит от типа данных и применяемого алгоритма и может колебаться в пределах от 1:2 до 1:8.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной и дополнительной литературы, интернет-источники.

Лекция 10 Стандарты и технологии локальных сетей.

План

1. История создания стандартов
2. Структура и содержание стандартов IEEE 802.x

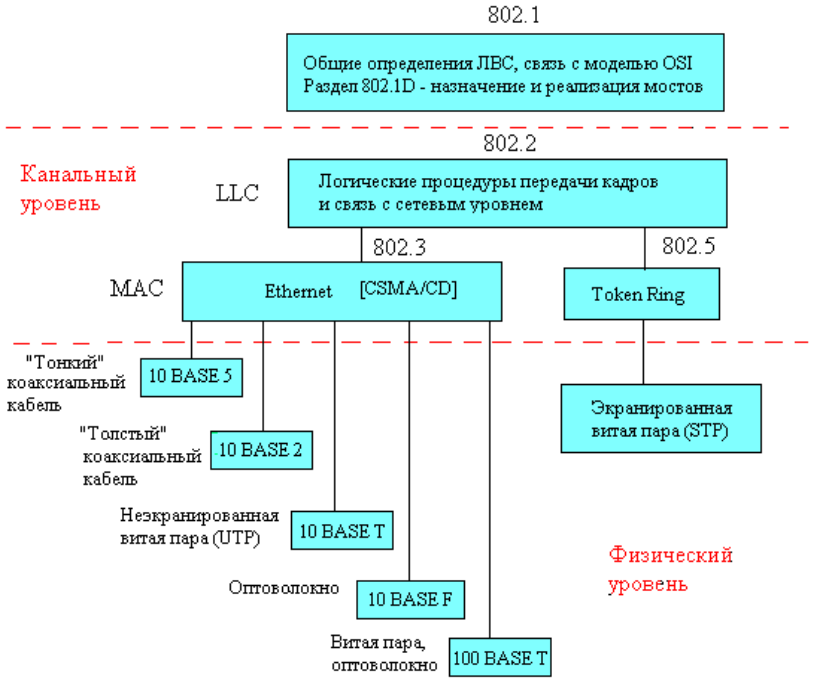
Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение стандартов и технологий организации локальных вычислительных сетей.

Ключевые вопросы – комитет 802 по стандартизации локальных сетей, структура его стандартов.

В 1980 году в институте IEEE был организован комитет 802 по стандартизации локальных сетей. Результатом его работы стало семейство стандартов IEEE 802.x, содержащих рекомендации по проектированию нижних уровней локальных сетей.

Стандарты IEEE 802 имеют достаточно четкую структуру, приведенную на рисунке.



Специфика локальных сетей нашла свое отражение в разделении канального уровня на два подуровня:

- логической передачи данных (Logical Link Control – LLC),
- управления доступом к среде (Media Access Control, MAC).

Уровень MAC появился из-за существования в ЛВС разделяемой среды передачи данных. Этот уровень обеспечивает корректное совместное использование общей среды, предоставляя ее в соответствии с определенным алгоритмом в распоряжении той или иной станции сети. После того как доступ к среде получен, ею может пользоваться более высокий уровень – LLC, организующий передачу логических единиц данных с различным уровнем качества транспортных услуг. В современных ЛВС получили распространение несколько протоколов уровня MAC, реализующих различные алгоритмы доступа к передающей среде. Эти протоколы полностью определяют специфику отдельных технологий.

Стандарты 802.3, 802.4, 802.5 и 802.12 описывают технологии локальных сетей, полученные в результате улучшений фирменных технологий, легших в их основу. К други известным стандартам относятся:

802.6 – Metropolitan Area Network , MAN – сети мегаполисов;

802.8 – Fiber Optic Technical Advisory Group – техническая консультационная группа по волоконно-оптическому кабелю;

802.9 – Integrated Voice and data Networks – интегрированная среда передачи голоса и данных;

802.10 – Network Security – сетевая безопасность;

802.11 – Wireless Networks – беспроводные сети.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 10

Тема Локальная сеть Ethernet.

План

1. История создания сети Ethernet.
2. Основные технологии.
3. Принципы подключения устройств

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение стандартов и технологий организации ЛВС.

Ключевые вопросы - стандарт IEEE 802.3, технологии Ethernet.

Ethernet – самый распространенный в настоящее время стандарт локальных сетей.

Ethernet версии 1 появилась в 1980 г. В 1982 году была опубликована спецификация на Ethernet версии 2.0. Обе версии используются до сих пор, причем между ними существуют различия и по интерфейсу, и по уровню сигналов. На базе Ethernet версии 2 институтом IEEE был разработан стандарт IEEE 802.3. В 1995 году был принят стандарт Fast Ethernet, который во многом не является самостоятельным стандартом. Его описание 802.3u является дополнительным разделом к основному стандарту. Аналогично, принятый 1998 году стандарт Gigabit Ethernet описан в разделе 802.z основного документа. В таблице приведены данные «классических» технологий сети Ethernet.

Ethernet	Thick Wire Ethernet	Thin Wire Ethernet	UTP Ethernet	Fiber Optic Ethernet	Broadband Ethernet
IEEE 802.3	10BASE-5	10BASE-2	10BASE-T	10BASE-F	10BROAD36
Скорость передачи данных, Мбит/с	10	10	10	10	10
Метод передачи сигналов	Одно-Полосной	Одно-полосной	Одно-полосной	Одно-полосной	Широко-полосной
Длина сегмента кабеля,м	500	185	100	2000	1800
Максим. расстояние между узлами сети (при использовании повторителей), м	2500	925	500	2500	3600
Максимальное число станций на сегменте	100	30	1024	1024	100
Максим. число повторителей Между любыми станциями сети	4	4	4	4	4
Тип кабеля	коаксиальный, «толстый» RG-8 или RG-11	коаксиальный, «тонкий» RG-58	Неэкранированная витая пара категории 3, 4, 5	Многомодовый волоконно-оптический кабель	75 Омный коаксиальный, «толстый»

Все конфигурации используют одинаковый метод доступа к среде передачи данных и единый формат пакета.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 12

Метод доступа к передающей среде и формат пакета сети Ethernet.

План

1. Основные принципы множественный доступ с контролем несущей и обнаружением коллизий

2. Условия корректной работы сети.

3.Взаимодействие устройств в сети

4.Формат кадра в Ethernet.

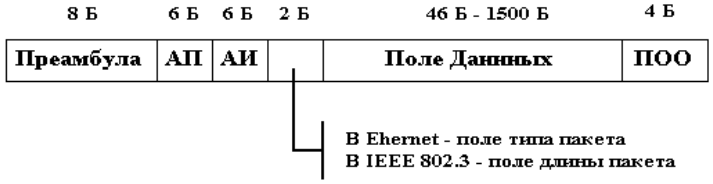
Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов взаимодействия устройств в сети Ethernet.

Ключевые вопросы – множественный доступ, коллизия, формат передаваемых данных.

Спецификация Ethernet и IEEE 802.3 определяют несколько конфигураций подключения оборудования. Они различаются типом кабеля, топологией подключения устройств и т.п. Во всех конфигурациях используется один метод доступа станций к среде – множественный доступ с контролем несущей и обнаружением коллизий – Carrier Sense Multiple Access/Collision Detection (CSMA/CD).

Во время работы станция постоянно проверяет среду передачи. Передающая среда может быть свободна, если ни одна другая станция не передает данные, или занята. Любая станция, обнаружив, что среда свободна, может начать передачу своих данных. Поэтому возможна ситуация, когда одновременно несколько станций начнут передавать данные – коллизия. При этом происходит смешение и искажение сигналов. Четкое распознавание коллизий всеми станциями сети является необходимым условием корректной работы сети Ethernet. Если какая-либо станция не распознает коллизию и решит, что переданный ею кадр верен, то кадр будет утерян или значительно искажен. Для надежного распознавания коллизий должно выполняться соотношение: $T_{min} \geq PDV$, где T_{min} – время передачи кадра минимальной длины, а PDV – время, за которое сигнал коллизии успевает распространиться до самого дальнего узла сети – время двойного оборота.



Минимальная длина пакета в IEEE 802.3 и Ethernet - 64 байта, соответственно длина поля данных - 46 байтов. IEEE 802.3 позволяет использовать символы-заполнители (PAD) в поле данных, если требуется передать сообщение короче, чем 46 байтов. В Ethernet пакет с длиной поля данных менее 46 байтов просто не будет обрабатываться. На рисунке показана структура пакета в сети Ethernet.

Назначение полей:

Преамбула служит для синхронизации работы приемника и передатчика.

АП - Адрес Приемника адрес станции, которой направляется пакет.

АИ - Адрес Источника - адрес передающей станции.

Поле Типа Пакета определяет тип Ethernet пакета.

Поле Длины Пакета определяет в IEEE 802.3 количество байт данных в поле данных. Количество символов-заполнителей (PAD) не входит в это число.

Поле Данных содержит данные и символы-заполнители в IEEE 802.3, данные - в Ethernet.

ПОО Поле Обнаружения Ошибок служит для определения достоверности полученной информации.

IEEE 802.3 определяет, что после поля адреса источника следует двухбайтное поле длины пакета, а в Ethernet - поле типа пакета

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники.

Лекция 13

Тема Высокоскоростные варианты сети Ethernet

План

1. История создания высокоскоростных вариантов
2. Технология Fast Ethernet
3. Технология Гигабит Ethernet
4. Технология 10 Гигабит Ethernet

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов взаимодействия устройств в сети Ethernet.

Ключевые вопросы – история создания и современные высокоскоростные технологии.

К ранним высокоскоростным вариантам относятся: коммутированная Ethernet, дуплексная Ethernet, 100BaseVG AnyLAN. Первые две использовались лишь в качестве временного решения, в 100BaseVG AnyLAN используется другой метод доступа – обработка запросов по приоритету, что усложняет совместимость с существующими сетями Ethernet.

Технология 100BaseX или Fast Ethernet реализуется с помощью дуплексной передачи сигналов по традиционной для Ethernet схеме доступа с контролем несущей и обнаружением коллизий в комбинации с уровнем зависимости от физической среды. Она принята в 1995 году в качестве дополнения 802.3u к существующему стандарту 802.3.

100BASE-T - общий термин для обозначения стандартов, использующих в качестве среды передачи данных витую пару. Длина сегмента до 100 метров. Включает в себя стандарты 100BASE-TX, 100BASE-T4 и 100BASE-T2.

100BASE-TX, IEEE 802.3u – развитие стандарта 10BASE-T для использования в сетях топологии "звезда". Задействована витая пара категории 5, фактически используются только две неэкранированные пары проводников, поддерживается дуплексная передача данных, расстояние до 100 м.

Более сложная структура физического уровня технологии Fast Ethernet вызвана тем, что в ней используется три варианта кабельных систем: волоконно-оптический многомодовый кабель (2 волокна), 2-х парная витая пара категории 5; 4-х парная витая пара категории 3. Fast Ethernet всегда имеет иерархическую древовидную структуру, построенную на концентраторах, подобно технологиям 10Base-T и 10Base-F. Диаметр сети сокращен до 200 метров, что объясняется уменьшением времени передачи кадра минимальной длины.

При использовании коммутаторов протокол Fast Ethernet может работать в полнодуплексном режиме, в котором нет ограничения на длину сети, а существуют лишь ограничения на длину физических сегментов. Формат кадра не отличается от формата 10-мегабитного Ethernet.

Гигабит Ethernet – поддерживает скорость обмена до 1 Гбит/с. Сеть Gigabit Ethernet представляет собой ЛВС, в которой применяется многомодовый оптоволоконный кабель, но IEEE изучает методы ее применения на сетях с UTP 5 категории. Топология звездообразная. Оптоволоконный кабель используется для магистрали, коаксиальный кабель для отводов к концентраторам.

1000BASE-T, IEEE 802.3ab – стандарт, использующий витую пару категорий 5e. В передаче данных участвуют все 4 пары. Скорость передачи данных – 250 Мбит/с по одной паре. Используется метод кодирования PAM5, частота основной гармоники 62,5 МГц.

Новый стандарт 10 Гигабит Ethernet включает в себя семь стандартов физической среды для LAN, MAN и WAN. В настоящее время он описывается поправкой IEEE 802.3ae и должен войти в следующую ревизию стандарта IEEE 802.3.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 14

Тема Сеть Token Ring

План

1. История создания стандарта Token Ring. Основные характеристики
2. Метод доступа станций к передающей среде
3. Взаимодействие устройств в сети, способы их подключения
4. Типы пакетов, форматы

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов взаимодействия устройств в сети Token Ring.

Ключевые вопросы – маркерное кольцо, однократно и двукратно подключаемые устройства.

Крупные предприятия имеют сети смешанной структуры. Доминирует среди них Ethernet, но присутствует и большое количество сетей Token Ring (маркерное кольцо).

Этот стандарт разработан фирмой IBM в 1984 году. В качестве передающей среды применяется неэкранированная или экранированная витая пара (UTP или STP) или оптоволокно. Скорость передачи данных 4 Мбит/с или 16 Мбит/с. В качестве метода управления доступом станций к передающей среде используется метод маркерное кольцо (Token Ring). Основные положения этого метода:

1) устройства подключаются к сети по топологии кольцо;
2) все устройства, подключенные к сети, могут передавать данные, только получив разрешение на передачу (маркер);
3) в любой момент времени только одна станция в сети обладает таким правом.

На базе IBM Token Ring в 1985 году комитетом IEEE 802 был разработан стандарт IEEE 802.5. Этот стандарт в отличие от IBM Token Ring не определяет среду передачи данных и топологии подключения устройств.

В IBM Token Ring используются три основных типа пакетов: пакет управление/данные, маркер, пакет сброса.

Станция, которая приняла маркер, получает право на передачу, и может передавать данные. Для этого станция удаляет маркер из кольца, формирует пакет данных и передает его следующей станции. В сети Token Ring все станции принимают и ретранслируют все пакеты, проходящие по кольцу. При приеме станция сравнивает поле адреса приемника пакета с собственным адресом. Если адреса не совпадают, то пакет передается далее по кольцу без изменений. Если адреса совпадают, или принят пакет с широковещательным адресом, то содержимое пакета копируется, а по результатам приема данных вносятся изменения в поле статуса пакета. Затем пакет передается далее по сети, и таким образом возвращается на станцию - отправитель. Получив пакет, станция-отправитель проверяет поле статуса пакета. Если при приеме пакета станцией-приемником была обнаружена ошибка, выполняется повторная передача пакета. Если ошибок при приеме не было или станция-приемник в данный момент не работает, станция - отправитель удаляет пакет из кольца, формирует маркер и передает его следующей станции. Такой алгоритм доступа применяется в сетях Token Ring со скоростью передачи 4 Мбит/с, описанный в стандарте 802.5. В сетях Token Ring 16 Мбит/с используется алгоритм раннего освобождения маркера.

Логически сеть Token Ring представляет собой кольцо, а физически – звезду, в которой устройства подключаются к сети через специальный концентратор.

Технология позволяет использовать различные типы кабелей: STP Type1, UTP Type 3 и 6, а также волоконно-оптический кабель. Максимальное расстояние от станции до MSAU – 100 м для STP и 45 м для UTP. Возможно также расширение сети с помощью мостов и повторителей. Сеть Token Ring может включать 260 узлов.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 15

Тема Распределенный волоконно-оптический интерфейс передачи данных FDDI.

План

1. История создания стандарта FDDI. Основные характеристики
2. Метод доступа станций к передающей среде
3. Взаимодействие устройств в сети, способы их подключения
4. Типы пакетов, форматы

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов взаимодействия устройств в сети FDDI.

Ключевые вопросы – маркерное кольцо, однократно и двукратно подключаемые устройства.

Стандарт был разработан в середине 80-х годов Национальным Американским Институтом Стандартов (ANSI) и получил номер ANSI X3T9.5.

FDDI обеспечивает передачу данных со скоростью 100 Мбит/с по двойному волоконно-оптическому кольцу на расстояние до 100 км.

Стандарт FDDI определяет кольцевую топологию с маркерным доступом, способную охватить большую площадь. Этот стандарт обеспечивает совместимость с сетями Token Ring, поскольку форматы кадров у них одинаковы.

Стандарт FDDI определяет перечень компонентов сети, который включает однократно подключенную станцию (SAS – Single Attached Station), двукратно подключенную станцию (DAS – Dual Attached Station) и концентраторы проводных линий. Соединения SAS с концентраторами имеют топологию звезды.

Интерфейс двукратного подключения обеспечивает отказоустойчивость системы благодаря своей избыточности. В случае разрыва кабеля сеть выполняет “заворачивание” – включает второе кольцо для обхода отказавшей станции. Сеть продолжает работать, но ее производительность падает.

Существуют также версии FDDI на медных проводах. «Медные» варианты FDDI не являются как таковыми стандартами и не могут гарантировать совместимость. Второе ограничение в “медных” версиях и даже в самом FDDI заключается в том, что здесь используется иная топология(двойное кольцо) и и другой размер пакета, нежели например в Ethernet.

Максимальное количество станций двойного подключения в кольце – 500. Максимальные расстояния между соседними узлами для многомодового кабеля – 2 км, для витой пары UTP категории 5 – 100 м, а для многомодового волокна зависит от его качества.

Максимальный размер кадра – 4500 байтов. Формат кадра FDDI:

Преамбула	Началн. разделитель	Контроли кадра	Адрес получа-те-ля	Адрес отправи-теля	Данные	Посл-ть проверки кадра	Конечный разделитель	Статус кадра

Поле начального разделителя обозначает начало кадра. Поле контроля кадра содержит данные о режиме передачи (синхронная или асинхронная), размере адреса и типе кадра (кадр LLC или управляющий кадр MAC). В поле последовательности проверки кадра для контроля ошибок используется 32-разрядный циклический избыточный код. Поле конечного разделителя означает конец кадра. Поле статуса кадра используется для индикации обнаружения ошибки.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 16

Тема Использование мостов в локальных сетях.

План

1. Построение ЛВС с помощью мостов и коммутаторов.
2. Алгоритмы маршрутизации.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов работы и преимущества использования мостов в локальных сетях.

Ключевые вопросы – алгоритм работы прозрачного моста, алгоритм с маршрутизацией от источника, алгоритм остовного дерева

Мосты функционируют на подуровне управления доступом к среде передачи канального уровня модели OSI.

При проектировании сетей мосты являются необходимыми элементами, с их помощью обеспечивается повышение эффективности, безопасности и дальности.

Чаще всего мосты устанавливаются в целях повышения эффективности. Администратор сети может воспользоваться мостом для уменьшения перегрузки и повышения быстродействия: большая сеть делится на несколько подсетей, соединенные мостами. Две небольшие сети будут работать быстрее, поскольку трафик локализуется в пределах подсети, кроме этого они более просты в управлении и обслуживании. Использование мостов приводит к повышению эффективности работы сети еще и потому, что разработчик может использовать разные топологии среды передачи, а затем соединить эти сети посредством мостов. Поскольку комитет IEEE 802 разработал для различных сетевых архитектур общий уровень управления логическим каналом, то существует возможность объединения, например, двух сетей Token Ring, разделенных ЛВС Ethernet. ЛВС Ethernet может пересылать пакеты так же, как почтальон может доставлять письма, написанные на иностранном языке, если конверты (пакеты) оформлены в соответствии со стандартом.

Мосты часто используют в целях повышения безопасности, т.к. программирую мосты на передачу только тех пакетов, которые содержат определенные адреса отправителя и получателя, можно ограничить круг рабочих станций, которые могут посылать и принимать информацию из другой подсети, тем самым предотвращая несанкционированный доступ. Здесь часто используют интеллектуальные мосты.

Мосты можно также использовать и в целях повышения отказоустойчивости системы, если с помощью внутренних мостов связать два файловых сервера, которые постоянно будут подстраховывать друг друга, причем при этом снизится уровень трафика.

Мосты увеличивают дальность охвата сети. Поскольку мост ретранслирует пакет в широковещательном режиме, то он функционирует как повторитель.

Алгоритмы работы мостов:

алгоритм работы прозрачного моста, в котором мост строит адресную таблицу на основании пассивного наблюдения за трафиком.

алгоритм остовного дерева разработан фирмами DEC и Vitalink, а в последствии принят комитетом IEEE 802.1 как стандарт. Этот алгоритм применяется для объединения мостами нескольких Ethernet сетей при возможности существования более одного непустого пути.

алгоритм с маршрутизацией от источника используется в сетях Token Ring и FDDI, в нем ответственность за разработку для кадра полного маршрута возлагается на рабочую станцию – источник. Станция-отправитель помещает в посылаемый в другое кольцо кадр всю адресную информацию о промежуточных мостах и кольцах. При получении каждого пакета мост просматривает поле маршрутной информации на наличие в нем своего идентификатора.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 17

Тема Глобальные сети.

План

1.Принципы организации глобальных сетей. Классификация.

2. Отличие глобальных сетей от локальных

3. Организация корпоративных сетей.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов работы глобальных сетей.

Ключевые вопросы – технологии глобальных сетей, магистральные сети и сети доступа.

Глобальные сети обычно создаются крупными телекоммуникационными компаниями для оказания платных услуг абонентам. Такие сети называют публичными или общественными. Существуют также такие понятия, как оператор сети и поставщик услуг сети. Гораздо реже глобальная сеть полностью создается какой-нибудь крупной корпорацией (такой, например, как Dow Jones или «Транснефть») для своих внутренних нужд. В этом случае сеть называется частной. Очень часто встречается и промежуточный вариант - корпоративная сеть пользуется услугами или оборудованием общественной глобальной сети, но дополняет эти услуги или оборудование своими собственными. Наиболее типичным примером здесь является аренда каналов связи, на основе которых создаются собственные территориальные сети.

Кроме вычислительных глобальных сетей существуют и другие виды территориальных сетей передачи информации. В первую очередь это телефонные и телеграфные сети, работающие на протяжении многих десятков лет, а также телексная сеть. Ввиду большой стоимости глобальных сетей существует долговременная тенденция создания единой глобальной сети, которая может передавать данные любых типов. Существенного прогресса в этой области не достигнуто, хотя технологии для создания таких сетей начали разрабатываться достаточно давно - первая технология для интеграции телекоммуникационных услуг ISDN стала развиваться с начала 70-х годов. Пока каждый тип сети существует отдельно и наиболее тесная их интеграция достигнута в области использования общих первичных сетей - сетей PDH и SDH, с помощью которых сегодня создаются постоянные каналы в сетях с коммутацией абонентов. Тем не менее, каждая из технологий, как компьютерных сетей, так и телефонных, старается сегодня передавать «чужой» для нее трафик с максимальной эффективностью, а попытки создать интегрированные сети на новом витке развития технологий продолжаются под преемственным названием Broadband ISDN (B-ISDN), то есть широкополосной (высокоскоростной) сети с интеграцией услуг. Сети B-ISDN будут основываться на технологии ATM, как универсальном транспорте, и поддерживать различные службы верхнего уровня для распространения конечным пользователям сети разнообразной информации - компьютерных данных, аудио- и видеоинформации, а также организации интерактивного взаимодействия пользователей.

Хотя в основе локальных и глобальных вычислительных сетей лежит один и тот же метод - метод коммутации пакетов, глобальные сети имеют достаточно много отличий от локальных сетей. Эти отличия касаются как принципов работы (например, принципы маршрутизации почти во всех типах глобальных сетей, кроме сетей TCP/IP, основаны на предварительном образовании виртуального канала), так и терминологии.

Технологии глобальных сетей определяют два типа интерфейса: «пользователь-сеть» (UNI) и «сеть-сеть» (NNI).

Глобальные компьютерные сети работают на основе технологии коммутации пакетов, кадров и ячеек.

Глобальные сети делятся на магистральные сети и сети доступа.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, интернет-источники, периодические издания.

Лекция 18

Тема Управление сетью

План

1. Протоколы управления.
2. Сетевые операционные системы.

Цель – организация целенаправленной познавательной деятельности студентов по овладению программным материалом.

Задача – изучение принципов управления работой вычислительных сетей.

Ключевые вопросы – технологии глобальных сетей, магистральные сети и сети доступа.

Типичная сеть состоит из множества устройств различных производителей. Управлять такой сетью возможно только при наличии стандартного, не зависящего от производителя протокола управления. Самым популярным стандартным протоколом управления в современных сетях является простой протокол управления сетью (Simple Network Management Protocol, SNMP). Широкое распространение он получил в силу своей гибкости и расширяемости — SNMP позволяет описывать объекты для самых разных устройств.

Обычно при использовании SNMP присутствуют управляемые и управляющие системы. В состав управляемой системы входит компонент, называемый *агентом*, который отправляет отчёты управляющей системе. По существу SNMP агенты передают управленческую информацию на управляющие системы как переменные (такие как «свободная память», «имя системы», «количество работающих процессов»).

Управляющая система может получить информацию через операции протокола GET, GETNEXT и GETBULK. Агент может самостоятельно без запроса отправить данные, используя операцию протокола TRAP или INFORM. Управляющие системы могут также отправлять конфигурационные обновления или контролирующие запросы, используя операцию SET для непосредственного управления системой. Операции конфигурирования и управления используются только тогда, когда нужны изменения в сетевой инфраструктуре. Операции мониторинга обычно выполняются на регулярной основе. Переменные, доступные через SNMP, организованы в иерархии. Эти иерархии и другие метаданные (такие, как тип и описание переменной) описываются Базами Управляющей Информации.

Сетевая операционная система обеспечивает поддержку сетевого оборудования, сетевых протоколов, протоколов маршрутизации, фильтрации сетевого трафика, доступа к удалённым ресурсам и пр.

Примерами сетевых операционных систем служат: Novell NetWare, Microsoft Windows (95, NT и более поздние), различные UNIX системы, такие как Solaris, FreeBSD, различные GNU/Linux системы и др. Сетевая ОС предоставляет виртуальную вычислительную систему не полностью скрывающую распределенную природу своего реального прототипа. Пользователю необходимо помнить, что при работе с сетевыми ресурсами необходимо выполнить особые операции. В распределенных ОС сетевые ресурсы предоставляются пользователю в виде единой централизованной виртуальной машины.

Сетевые средства делятся на три компонента: средства предоставления локальных ресурсов и услуг в общее пользование – серверная часть; средства запроса доступа к удаленным ресурсам сети – клиентская часть; транспортные средства, которые совместно с коммуникационной системой обеспечивают передачу сообщений между компьютерами сети. Совокупность серверной и клиентской частей, предоставляющих доступ к конкретному типу ресурса компьютера, называются сетевой службой. В сетях с выделенным сервером используют специальные варианты ОС, оптимизированные для работы либо с серверами, либо с клиентами.

Ссылки на литературные источники, приведенные в рабочей программе дисциплины: пп. 1, 2 основной литературы, п.1 дополнительной литературы, интернет-источники, периодические издания.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Методические указания по изучению дисциплины

Для успешного освоения данной дисциплины студентам предлагаются:

- содержание разделов и тем дисциплины, включающей лекционные и лабораторные занятия;
- контрольные вопросы для самостоятельной работы;
- список рекомендуемой основной и дополнительной литературы;
- вопросы к экзамену.

В течении семестра студент не только должен изучать материал лекций, но и готовить вопросы самостоятельной работы на основе рекомендуемой литературы.

Основными формами самостоятельной (внеаудиторной) работы студентов являются:

подготовка отдельных вопросов по темам программы;

участие в научных и научно-практических студенческих конференциях;

подготовке к лабораторной работе по определенной теме;

подготовка и написание отчетов к лабораторным работам;

подготовка к промежуточному и итоговому контролю.

Самостоятельная работа начинается до прихода студента на лекцию. Весьма эффективно использование «системы опережающего чтения», т.е. предварительного прочитывания лекционного материала, содержащегося в учебниках, учебных пособиях, в результате чего закладывается база для более глубокого восприятия лекции.

В процессе организации самостоятельной работы большое значение имеют консультации преподавателя, в ходе которых решаются многие проблемы изучаемого курса, уясняются наиболее сложные вопросы.

Контроль самостоятельной работы осуществляется тестированием, которое может проводиться в письменной форме, либо с использованием компьютерной системы тестирования. Примерные тестовые задания приведены в рабочей программе. Тест включает в себя вопросы с открытыми и закрытыми вариантами ответов. Результаты тестирования включены в балльно-рейтинговую систему оценки знаний (см. рабочую программу).

Экзамен является завершающим этапом учебного процесса, на котором проводится подведение итогов всей самостоятельной работы студентов.

Подготовку к экзамену требуется начинать с просмотра перечня всех вопросов с целью оценки требуемого объема учебного материала, логики и структуры построения курса. С учетом накопленных за семестр знаний студент должен запланировать распределение времени на подготовку. Желательно зарезервировать время для повторения материала. Работа над каждым из вопросов рекомендуется прочитать конспект лекции, дополнительно прочитать рекомендованный учебник, если материал трудно усваивается. Завершается работа восстановление в памяти прочитанного.

Экзаменационный билет включает в себя два теоретических вопроса из перечня.

При оценке на экзамене учитывается: полнота ответа на поставленный вопрос, точность формулировок, логичность ответа, умение делать выводы, выявлять закономерности, соблюдение норм литературной речи и использования специализированной терминологии. Высшего балла заслуживает ответ, удовлетворяющий всем этим требованиям.

Методические указания к лабораторным работам

Лабораторные работы проводятся по подгруппам в компьютерном классе. Каждый студент получает индивидуальное задание в соответствии с вариантом.

Выполняя задание, студент пользуется материалом, изученным в тексте лабораторной работы.

Перед созданием любой программы требуется точно продумать алгоритм. Записать его блок-схемой или словесно. Надо четко определить, что в нее требуется ввести и что получить в результате, в какой последовательности выполнять действия. В случае необходимости выделить циклические структуры и подпрограммы. В циклах четко определить параметры, задать их начальные значения, определить условия повторения и завершения цикла. В функциях определить количество передаваемых и возвращаемых значений.

При кодировании программы нужно определить тип используемых данных в зависимости от возможного диапазона принимаемых значений. При вводе величины не забывать осведомить об этом пользователя, а иногда сообщить и о типе, диапазоне или порядке ввода значений. Такое сообщение должно быть информативно и коротко. Вывод данных лучше сопровождать текстом и форматированием. Формат вывода можно уточнить при помощи модификаторов.

В именах переменных необходимо отражать их назначение, что повышает читаемость и понимание программы.

При записи сложных выражений нужно обращать внимание на приоритет операций. Текст программы лучше сопровождать краткими и информативными комментариями, что облегчает как понимание программы, так и ее отладку.

Объявление локальных переменных предпочтительнее по сравнению с глобальными.

Для отладки программы нужно запустить ее на выполнение несколько раз, задавая различные значения вводимых величин. Перед запуском необходимо иметь заранее подготовленные тестовые примеры, содержащие исходные данные и ожидаемые результаты. Их количество зависит от алгоритма. Проверьте реакцию программы на заведомо неверные исходные данные.

Для быстрого поиска ошибки в алгоритме рекомендуется выводить промежуточные данные.

При сдаче лабораторной работы студент должен продемонстрировать преподавателю созданную программу, правильно работающую, отлаженную.

Преподаватель, принимая лабораторную работу, тестирует программу студента и задает ему вопросы по конструкциям, используемым в программе и теоретическим основам программирования.

ЗАДАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

Лабораторная работа 1 **Основы JavaScript**

Для выполнения программ JavaScript в качестве интерпретатора (исполнительной системы) можно использовать веб-браузер Internet Explorer 6.0 (5.0). В качестве редактора программ текстовый редактор, например Блокнот. Создавать программы на JavaScript можно и с помощью специальных программ, предназначенных для разработки веб-сайтов, например Microsoft FrontPage.

Можно легко создать и собственный редактор программ, например кодом:

```
<HTML>
<H3>Редактор кодов</H3>
код:<br>
<TEXTAREA id="mycode" ROWS=10 COLS=60></TEXTAREA>
<p>Результат:<br>
<TEXTAREA id="myrezult" ROWS=3 COLS=60></TEXTAREA>
<P>
<BUTTON onclick="document.all.myrezult.value=eval(mycode.value)">
Выполнить</BUTTON>
<BUTTON onclick="document.all.mycode.value=' ';
```

```
document.all.myrezalt.value=' ' ">
Очистить </BUTTON>
<P>
<!Комментарий>
Введите выражения в верхнее поле.
Выражения разделяются точкой с запятой.
Можно также каждое выражение писать в отдельной строке
</HTML>
```

Ввод-вывод данных

Для обеспечения ввода-вывода JavaScript предоставляет несколько методов: alert( ), confirm ( ), prompt( ).

Первый из перечисленных выводит на экран диалоговое окно с заданным сообщением и кнопкой ОК.

Синтаксис данного метода:

alert (сообщение)

Сообщение может представлять собой данные любого типа: последовательность символов, заключенную в кавычки, число, переменную или выражение.

Текст необходимо заключить в кавычки. Например,

alert (“Всем привет! ”)

Для формирования строк используют служебные символы:

\n – новая строка,
\t – табуляция,
\f – новая страница,
\b – забой,
\r – возврат каретки.

Окно, создаваемое alert( ) является модальным (останавливающим все последующие действия программы и пользователя). Его можно убрать, щелкнув по кнопке ОК.

Метод confirm выводит на экран диалоговое окно с сообщением и двумя кнопками – ОК и Отмена. Этот метод возвращает логическую величину, значение которой зависит от того, по какой из кнопок щелкнет пользователь. Возвращаемое значение можно обработать в программе, создавая тем самым интерактивный эффект. Синтаксис применения данного метода аналогичен синтаксису метода alert.

Окно, создаваемое confirm также является модальным.

Метод prompt осуществляет вывод диалогового окна с сообщением и кнопками ОК и Отмена, а также с текстовым полем, в которое пользователь может ввести данные. В отличие от alert( ) и confirm( ) данный метод принимает два параметра: сообщение и значение, которое должно появиться в текстовом поле ввода данных по умолчанию. Если пользователь щелкнет по кнопке ОК, метод вернет содержимое поле ввода данных, если – по кнопке Отмена, то возвращается значение ложь. Возвращаемое значение можно также обработать в программе. Синтаксис метода:

prompt (сообщение, значение_поля_ввода данных)

Оба параметра не являются обязательными. Если они не указаны, на экране появится окно без сообщения, а в поле ввода данных подставлено значение по умолчанию – undefined (не определено). Чтобы значение по умолчанию не появилось, в качестве второго параметра указывается пустая строка (“ ”).

Типы данных

Типы данных языка JavaScript приведены таблице 1. При создании программ за типом данных следит сам программист. Интерпретатор не выдаст ошибки при неверном их использовании. Он просто попытается привести данные к типу, требуемому в данной операции.

Тип данных	*Описание значений*
Строковый или символьный тип (string)	Последовательность символов, заключенная в кавычки, двойные или одинарные
Числовой (number)	Положительное или отрицательное число. Целая и дробная части разделяются точкой
Логический	Два значения: true или false
Null	Отсутствие какого-либо значения
Объект (object)	Программный объект с собственными свойствами. В частности, массив также является объектом.
Функция (function)	Программный код, выполнение которого может возвращать некоторое значение.

Табл.1. Типы данных JavaScript

Для преобразования строк в числа предусмотрены встроенные функции parseInt( ) и parseFloat( ). Синтаксис:

parseInt(строка, основание)

parseFloat(строка, основание)

Если основание не указано, то предполагается 10 – десятеричная система счисления. В качестве основания можно также использовать 8, 10, 16.

При преобразовании строки в целое число округление не происходит – дробная часть просто отбрасывается.

Для определения того, является ли значение выражения числом, служит встроенная функция isNaN(значение). Функция возвращает логический тип. Данная функция считывает числа, данные числового типа и строки, содержащие только числа.

Переменные

Имя переменой представляет собой конечную последовательность символов, содержащую буквы, цифры, символ подчеркивания. Имя переменной не должно начинаться с цифры или содержать пробелы. Для имен переменных нельзя использовать ключевые слова языка.

В отличие от многих других языков программирования переменной не нужно задавать тип при объявлении. Тип переменной определяется типом ее значения. Переменная может принимать значения разных типов и неоднократно его изменять.

Создавать переменную в программе можно несколькими способами. Можно ей просто присвоить значение с помощью оператора присваивания в формате: имя_переменной = значение

Например,

Month= “Январь”

Можно использовать ключевое слово var перед именем переменной. В этом случае переменная не будет иметь первоначальное значение, но в дальнейшем его можно передать с помощью оператора присваивания. Например,

var Month

Month = “Январь”

При использовании var допускается и инициализация переменной, например:

var Month = “Январь”

Для каждой переменной инициализация при объявлении возможна только один раз.

Можно сразу объявить несколько переменных, используя var и разделяя их запятой, при этом возможно инициализировать их все или некоторые:

var Month= “Январь”, day, pi=3.14, x

Комментарии

В JavaScript допустимы два вида операторов комментария:
- одна строка символов, расположенная справа от //
- произвольное количество строк, заключенных между /* и */.

Операции выполняются в соответствии с приоритетами. Приоритет операций аналогичен языку С++. Для изменения порядка выполнения операций используются круглые скобки.

Операторы ветвления

Условный оператор if используется для разветвления процесса вычислений на два направления. Синтаксис оператора if:

```
if ( условие ) оператор1  else оператор 2
```

Пример

```
if ( fvalue>=0.0 )  fvalue = fvalue  else fvalue = -fvalue
```

Условные операторы могут быть вложенными.

```
if ( a<b ) { if ( a<c ) m = a  else m = c }  else { if ( b<c ) m = b  else m = c }
```

Необходимо помнить, что в этом случае else относится к ближайшему if. Операторные скобки после первого if необязательны.

Если требуется проверить несколько условий, их объединяют знакам логических операций.

Оператор switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Синтаксис оператора:

```
switch ( выражение) {
case  константное выражение 1  : операторы1   break
case  константное выражение 2  : операторы2   break
…
case  константное выражение n  : операторыN   break
default : операторы }
```

Параметр выражения оператора switch может принимать строковые, числовые и логические значения. В следующем примере переменная x содержит название языка, который выбрал пользователь. А выражение window.open ( ) открывает новое окно браузера и загружает в него указанный в скобках HTML-документ.

```
 switch (x ) {
        case " английский" : window.open ( "engl.htm") ; break
        case  " французский"  : window.open ( "french.htm") ;     break
        case  " русский" :  window.open ( "russ.htm") ;           break
        default  :  alert(  "Нет документа на таком языке" )
}
```

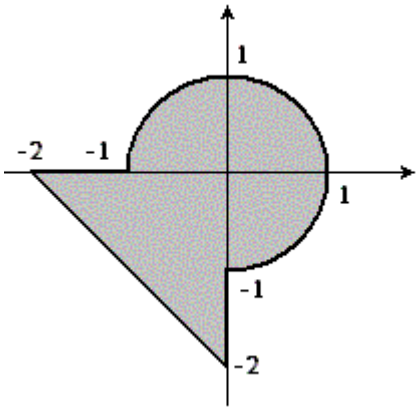
Задание

1. Создать собственный редактор программ, на основе HTML-кода, модифицировав код, приведенный выше. Файл открыть в браузере как веб-страницу.
2. Протестировать редактор, изучив в нем функции вывода и основные операции JavaScript.
3. Составить программу на основе разветвляющего алгоритма для задачи:

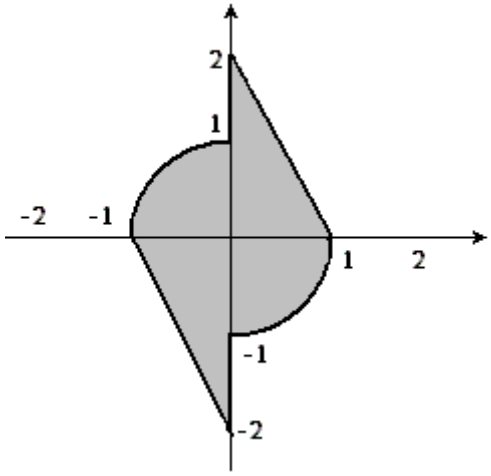
Дана «мишень» в виде закрашенной области, изображенной на рисунке. Создать алгоритм для определения, попадает ли заданная точка с координатами (x, y) в мишень.

4. Для удобства пользователя рисунок мишени отобразить в редакторе.

Вариант 1



Вариант 2



Лабораторная работа №2 **Операторы цикла. Организация пользовательских функций**

JavaScript содержит следующие встроенные функции (некоторые уже были рассмотрены):

parseInt(строка, основание) – преобразует указанную строку в целое число по указанному основанию(8, 10, 16); по умолчанию - десятеричная система.

parseFloat(строка, основание) – преобразует строку в число с плавающей точкой.

isNaN(строка, основание) – возвращает true, если указанное в параметре значение не является числом.

eval(строка) – вычисляет выражение в указанной строке, выражение не должно содержать тегов HTML.

Например,

```
var a = 10;                          // значение a равно 10
var b = “if (a<=25) { a *= 2 }”      // значение b равно строке символов
eval (b)                             // значение a равно 20
```

Другой пример применения данной функции в тексте программы редактора. Там сценарии записаны в качестве значений атрибутов onclik, определяющих событие щелчок кнопкой мыши на HTML-кнопках, заданных тегами <button>.

escape(строка) – возвращает строку в виде %XX, где XX – ASCII-код указанного символа. Такую строку называют escape-последовательностью.

unescape(строка) – обратное преобразование.

type0f (объект) – возвращает тип указанного объекта в виде символьной строки, например “boolean”, “function”.

В программах на JavaScript пользователям также разрешено создавать собственные функции. Объявление функция состоит из заголовка и тела:

```
function имя_функции ( параметры )        //заголовок функции
{ тело функции }
```

Тело функции ограничивается фигурными скобками. Параметры функции, стоящие в круглых скобках перечисляются через запятую.

Возвращение значений из функции происходит с помощью оператора return, за ко-

торым помещается само возвращаемое значение.

Для вызова функции можно воспользоваться выражениями вида:
имя_функции (параметры) или
имя_переменной = имя_функции(параметры)

Параметры в вызове функции должны быть представлены конкретными значениями.

Например, если описание функции дано в виде:

```
function cube (x)
{ return x*x*x }
```

Данная функция вычисляет куб от параметра функции. Ее вызов может быть записан в виде:

```
y=cube(25)
```

В JavaScript функции могут вызываться как после их определения, так и до него. Можно не поддерживать соответствие количества параметров в определении функции и в ее вызове. Если в определение функции параметров больше, чем в вызове, то недостающим параметрам автоматически присваивается значение null. Лишние параметры в вызове функции игнорируются.

Внутри функции можно создавать переменные с помощью оператора присваивания или с помощью ключевого слова var.

Если в теле функции переменная в составе оператора присваивания встречается впервые в программе, или она была определена до этого – она действует как глобальная.

Если в теле функции используется переменная, объявленная только во внешней программе, она также является глобальной.

Если для определения переменной в теле функции используется ключевое слово var, она будет локальной вне зависимости от того определена она во внешней программе или нет.

Операторы цикла

Как и другие языки программирования JavaScript имеет три вида оператора цикла: цикл с предусловием (while), цикл с постусловием (do while), цикл с параметром (for).

Синтаксис цикла с предусловием:

```
while ( условие )
{
операторы
}
```

Выражение, стоящее в круглых скобках, определяет условие повторения тела цикла, представленного простым или составным оператором. Если оператор простой операторные скобки { } могут не ставиться.

Выполнение оператора цикла начинается с вычисления выражения. Если оно истинно, выполняется тело цикла. Если при первой проверке выражение ложно (false), цикл не выполнится ни разу.

Значение выражения вычисляется перед каждой итерацией цикла.

Пример. Программа вычисляет возведение 10 в степень 5.

```
var x =10
y= 2
while ( y <=  5 )
        {
        x*=10;  y++
        }
```

Цикл while обычно используется в тех случаях, когда число повторений заранее не известно. В этом случае используется и оператор цикла с постусловием. Его отличительной чертой является выполнение тела цикла хотя бы один раз. И только после первого его выполнения проверяется, надо ли его выполнять еще раз. Таким образом, даже если условие заведомо ложно цикл выполняется один раз. Если условие истинно, тело цикло выполнится еще раз. Цикл завершается, когда выражение станет равным false или в теле цикла будет выполнен оператор передачи управления.

Цикл for называют также циклом с заданным числом повторений. Он имеет следующий формат:

```
for (инициализация; выражение (условие); модификации )
{
операторы
}
```

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. Инициализация выполняется один раз перед выполнением тела цикла.

Выражение определяет условие выполнения цикла: если его результат равен истине, то цикл выполняется. Цикл с параметром реализуется как цикл предусловием.

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла.

Тело цикла представляет собой простой или составной оператор.

```
for (i = 1, s=1; i<11; i++)
{
s*=i;                    // вычисления факториала 10
}
```

Для принудительного выхода из тела любого цикла используются операторы break и continue. break позволяет переход в точку программы, находящуюся непосредственно за оператором, внутри которого находится, т.е. управление передается первой строке, следующей за телом цикла.

Например,

```
 time=1
 sum=0
while (time<10)
 { sum+=time
                if ( sum>20 )  break;
   time++
                    }
```

Инструкция continue заставляет программу пропустить все оставшиеся строки цикла, но сам цикл при этом не завершается. Для решения некоторых задач удобно комбинировать инструкции break и continue.

Задание

1. Составьте программу на основе циклического алгоритма для вычисления суммы ряда с заданной точностью ε. Определите и выведите на экран значение суммы и число элементов ряда, вошедших в сумму.

Номер варианта	***Задание***	***Точность***

1	$\frac{\pi}{3}+\sum_{n=1}^{\infty}(-1)^n\frac{(\pi/3)^{2n+1}}{(2n+1)!}$	$0.5*\ 10^{-4}$
2	$\sum_{n=1}^{\infty}(-1)^n\frac{(\pi/6)^{2n}}{(2n)!}$	$0.5*\ 10^{-4}$

Лабораторная работа №3 **Встроенные объекты JavaScript. Строки и массивы**

Объекты представляют собой программные единицы, обладающие заданными свойствами. Как любой объект, объект JavaScript, обладает свойствами (данными) и методами (функциями) для их обработки. Программный код встроенных объектов JavaScript недоступен.

Управление web-страницами с помощью сценариев, созданных на JavaScript, заключается в использовании и изменении свойств объектов HTML-документа и самого браузера.

Встроенные объекты имеют фиксированные названия. Наиболее важными объектами в разработке web-сайтов являются String (символьные строки), Array (массивы), Math (математические формулы и константы), Date (работа с датами).

Объекты, названия которых совпадают с их фиксированными названиями, называются статическими. Можно создавать и экземпляры (копии) статических объектов, имеющие собственные имена и наследующие все свойства и методы статических объектов.

Встроенные объекты имеют прототипы (prototype), позволяющие добавлять новые свойства и методы к уже существующим в экземплярах объектов.

Объект String

С помощью объекта String можно создавать строку или строковый объект. Синтаксис:

имя_переменной = new String (“значение”)

Создать строковый объект можно и с помощью обычного оператора присваивания:

имя_переменной = “значение”

или

var имя_переменной = “значение”

Например,

mystring1 = new String (“строка”)

mystring2 = “строка”

Свойство String

length – длина (количество символов), включая пробелы.

Доступ к свойствам и методам объекта осуществляется через операцию точка. Например,

str = “весна”

str.length // значение равно 5

“лето”.length // значение равно 4

Методы String

Как и любые функции, методы могут иметь параметры. Параметры перечисляются через запятую в круглых скобках, стоящих после имени метода.

chartAt (индекс) – возвращает символ, занимающий в строке указанную позицию. Индекс является числом. Необходимо помнить, что нумерация элементов строки начинается с нуля.

Примеры

y = “Осень”.charAt(3) // значение “н”
str = “Зима”
str.charAt (str . length – 2) // значение “м”

charCodeAt(индекс) – преобразует символ в указанной позиции в его код. Поддерживает систему кодов Unicode, NN4 – ISO-Latin1.

fromCharCode(номер [1, номер2[, номер3, …, номерn]]) – возвращает строку символов, числовые коды символов которой указаны в строке параметров.

concat(строка) – конкатенация (слияние) строк.

Синтаксис: строка1.concat(строка2)

Возвращает строку, полученную дописыванием символов строки2 к строке1.

Пример

x = “Петр”
y = x .concat (“ Семенович”) // результат “Петр Семенович”

index0f(строка_поиска [, индекс]) - поиск строки, указанной параметром. Метод возвращает индекс первого вхождения строки. Поиск в пустой строке возвращает -1. Второй параметр, не является обязательным, о чем говоря квадратные скобки. Индекс указывает позицию, с которой начинается поиск.

lastIndex0f (строка_поиска [, индекс]) – поиск первого вхождения строки, указанной параметром. Причем поиск начинается с конца исходной строки, но возвращаемый индекс отсчитывается сначала.

localeCompare (строка) – сравнение строк в кодировке Unicode, то есть с учетом используемого браузером языка общения с пользователем. Синтаксис:

строка1.localeCompare(строка2)

Если строки одинаковы, метод возвращает 0. Если строка1 меньше, чем строка2, метод возвращается отрицательное число, в противном случае - положительное.

slice(индекс1[, индекс2]) – возвращает подстроку исходной строки, начальный и конечный индексы которой указываются параметрами, за исключением последнего символа. Второй параметр не обязателен. Если он не указан – подразумевается до конца строки.

split(разделитель [, ограничитель]) – возвращает массив элементов, полученных из исходной строки. Первый параметр – строка символов, используемая в качестве разделителя строки на элементы. Второй параметр – число, указывающее количество элементов возвращаемого массива из строки, полученной при разделении. Второй параметр необязателен. Если разделитель – пустая строка, возвращается массив символов строки.

Например,

x = “Привет всем!”
x.split(“ ”) // значение – массив из элементов “Привет”, “всем!”
x. split(“е”) // значение – массив из элементов “Прив”, “т вс”, “м!”

substr(индекс[, длина]) – возвращает подстроку исходной строки, начальный индекс и длина, которой указываются параметрами. Если второй параметр не указан, возвращается подстрока с начальной позиции до конца.

substring(индекс1, индекс2) – возвращает подстроку исходной строки, с позиции1 до позиции2.

toLocaleLowerCase( ), toLowerCase( ) – переводят строку в нижний регистр.

toLocaleUpperCase( ), toUpperCase ( ) – переводят строку в верхний регистр.

Тексты web-страниц, как правило, создаются и форматируются с помощью тегов HTML. Это же можно сделать средствами JavaScript.

Например, для вывода строки полужирным шрифтом используют метод bold( ). Данный метод не выводит строку в окно браузера, а лишь форматирует ее. Для вывода

строки в HTML-документе используется метод write( ) для объекта document:

```
<HTML>
<SCRIPT>
st = “Доброе утро!”.bold( )
document. write(st)
</SCRIPT>
</HTML>
```

Методы форматирования строк носят названия, соответствующие тегам HTML:
anchor(“anchor_имя”)
blinc( )
bold( )
fixed( )
fontcolor(значение цвета)
fontsize(число от 1 до 7)
italics( )
link(расположение или URL)
big( )
small( )
strike( )
sub( )
sup( )

Объект Array

Массив представляет собой упорядоченный набор данных. Нумерация элементов массива начинается с нуля. К элементам массива можно обращаться по их порядковому номеру, заключив его в квадратные скобки, расположенные после имени массива. Элементы массива в JavaScript могут быть разного типа.

Существует несколько способов создания массива:

1. имя_массива = new Array ([длина_массива])

Если длина массива не указана, создается пустой массив, не содержащий ни одного элемента. Иначе создается массив указанной длины, все элементы которого имеют значение null. Создав пустой массив, можно присвоить значения его элементам операцией присваивания.

2. инициализация массива при объявлении:

имя_массива = new Array (значение1[, значение2[, … значениен]])

3. инициализация каждого отдельного элемента массива, подобно свойствам объекта

имя_массива = new Array( )
имя_массива. имя_элемента1 = значение1
[имя_массива. имя_элемента2 = значение 2
[… имя_массива. имя_элемента *n* = значение *n*]]
Например,
child = new Array (4)
child[0]= “Женя”
child[1]= 22
child[2]= “ июнь”
child[3]= 1996
child = new Array (“Женя”, 22, “ июнь”, 1996)
y=child.length// y=4

```
child = new Array ( )
child.name = “Женя”
child.day= 22
child.month= “ июнь”
child.year= 1996
```

Свойства объекта Array

Свойство length, возвращает количество элементов объекта Array.

Свойство prototype позволяет добавлять новые свойства и методы для всех созданных массивов.

Например,

```
function SumNegative (massiv)
{var s = 0
  for (i=0; i<=massiv.length-1; i++)
          if (massiv[i]<0 s +=massiv[i]
         return s
}
mass = new Array(1, -9, 7, -23, -3)
Array.protptype.SumN = SumNegative       //добавляем метод к объекту
Massiv.SumN(mass)            // применяем метод SumN к массиву mass
```

Многомерные массивы

Для создания многомерного массива требуется указать все его размерности, заключив каждую из них в квадратные скобки.

Например,

```
matrix = new Array ( )
matrix[0] = new Array ( 2, 3, 8)
matrix[1] = new Array ( 1, -3, 7)
matrix[2] = new Array ( 0, 2, -6)      // массив размерности 3 на 3
```

Методы объекта Array

concat( ) – объединяет два массива в третий и возвращает полученный массив.

Синтаксис: имя_массива3 = имя_массива1.concat(имя_массива2)

join( ) – создает строку из элементов массива с указанным разделителем между ними, возвращает строку символов.

Синтаксис: имя_массива2 = имя_массива1.join(строка)

pop( ) – удаляет последний элемент массива и возвращает его значение.

Синтаксис: имя_массива1.pop( )

push( ) – добавляет к массиву последний элемент, значение которого указанно в качестве параметра и возвращает новую длину массива.

shift( ) – удаляет первый элемент массива и возвращает его значение.

unshift( ) – добавляет к массиву первый элемент, значение которого указанное в качестве аргумента.

reserve( ) – переписывает массив в обратном порядке, возвращает массив.

slise(индекс1[, индекс2]) – создает массив из элементов исходного массива с индексами указанного диапазона. Возвращает массив. Если второй индекс не указан, то новый массив создается из элементов с индекса1 до конца исходного массива.

sort( ) – упорядочивает элементы массива. Если параметр не указан, сортировка производится на основе ASCII-кодов символов значений, что удобно для строк, но не походит для чисел. Параметром может служить имя функции сравнивающей два элемента массива.

Пример,

```
massiv = new Array (7, 1, 34, 5, 63)
function cmp (x, y)
{ return x - y  }
massiv.sort(cmp)                    //массив будет сортироваться по возрастанию
```

Эта функция дает критерий сортировки.

splice(индекс, количество [, элем1[, элем2[, … элемn]]]) – удаляет (заменяет) из массива элементы. Возвращает массив из удаленных элементов. Первый параметр является индексом первого удаляемого элемента, второй - количеством удаляемых элементов. Если указаны необязательные параметры, то происходит замена элементов указанного диапазона на указанные значения параметров. Но это справедливо, если второй параметр не равен нулю.

Пример

```
b = new Array( “один”, 2, 3,  4, “пять”)
c = b.splice(1, 3, “два”,  “три”, “четыре”)
// массив b из элементов «один», «два», «три», «четыре», «пять»
// массив c из элементов 2. 3, 4
```

toLocaleString( ), toString( ) – преобразуют содержимое массива в символьную строку.

Задание

1. Создайте пользовательскую функцию для работы со строками с использованием методов объекта String.

Вариант 1.

Функция вставки строки в исходную строку. Функция должна иметь три параметра: исходную строку, вставляемую строку и позицию вставки.

Вариант 2.

Функция замены в исходной строке все вхождения заданной подстроки на подстроку замены. Функция должна иметь три параметра: исходную строку, заменяемую подстроку и подстроку, которой следует заменить все вхождения заменяемой подстроки.

2. Изучите функции форматирования строк. Продемонстрируйте работу функции из задания 1, отформатировав вновь полученную строку тремя различными способами в HTML-документе.

3. Создайте функцию для работы с объектом Array.

Вариант 1.

Замена минимального элемента значением, заданным как параметр функции.

Вариант 2.

Сортировка по возрастанию элементов массива, расположенных между максимальным и минимальным его элементами.

4. Создайте 2 массива. Добавьте созданную функцию из задания 3 в качестве нового свойства ко всем созданным массивам.

Лабораторная работа № 4 **Работа с окнами**

Главное окно браузера создается автоматически при запуске браузера. С помощью сценария можно создать любое количество окон, а также разбить окно на несколько прямоугольных областей, называемых фреймами. Окну браузера соответствует объект window, а HTML-документу, загруженному в окно, соответствует объект document. Эти объекты могут содержать в себе другие объекты. В частности, объект document входит в состав window.

Доступ к свойствам и методам данного объекта происходит, как и в других объектах, через точку. Поскольку объект document является подобъектом объекта window, ссылка на HTML-документ, загруженный в текущее окно: window.document. Объект document имеет метод write (запись строки в текущий HTML-документ).

Для его применения используют window.document.write(строка). Объект окна window - корневой объект, имеющий свои подобъекты. Например, location хранит информацию об URL-адресе загруженного документа, screen – данные о возможностях экрана монитора пользователя.

В объектной модели документа объекты сгруппированы в *коллекции*. Коллекция – промежуточный объект, содержащий объекты собственно документа. Коллекция является упорядоченным массивом объектов, отсортированных в порядке упоминания соответствующих им элементов в HTML-документе. Индексация объектов в коллекции начинается с нуля. Синтаксис обращения к элементам коллекции аналогичен синтаксису обращению к элементам массива. Коллекция имеет длину – свойство length.

Коллекция всех графических изображений документа называется images, коллекция всех форм – forms, ссылок – links. Коллекция всех объектов документа называется all.

Один и тот же объект может входить в частную коллекцию (например, images), но он обязательно входит в коллекцию all. При этом его индексы могут быть разными в разных коллекциях.

При использовании документа, загруженного в текущее окно, объект window можно не упоминать, а сразу начинать с объекта document.

Например,

document. images(0)

Вместо индекса можно использовать значение атрибута ID в теге, который определяет соответствующий элемент HTML-документа.

Однако универсальный способ обращения к объектам документа – обращение посредством коллекции all.

С помощью сценария можно создавать любое количество окон. Для этого применяется метод open( ):

window. open(параметры)

Данному методу передаются следующие необязательные параметры:

адрес документа, который нужно загрузить в создаваемое окно;

имя окна (как имя переменной);

строка описания свойств окна (features).

В строке свойств записываются пары свойство = значение, которые отделяются друг от друга запятыми.

Свойства, передаваемые в строке features

Свойство	Значения	Описание
channel mode	yes, no, 1, 0	Показывает элементы управления channel
directories	yes, no, 1, 0	Включают кнопки каталога
full1screen	yes, no, 1, 0	Полностью разворачивает окно
height	число	Высота окна в пикселях
left	число	Положение по горизонтали относительно левого края экран в пикселях
location	yes, no, 1, 0	Текстовое поле Address
menubar	yes, no, 1, 0	Стандартное меню браузера
resizeable	yes, no, 1, 0	Возможность пользователя изменять размер окна
scrollbars	yes, no, 1, 0	Горизонтальные и вертикальные полосы прокрутки
status	yes, no, 1, 0	Стандартная строка состояния Включает панели инструментов браузера

toolbar	yes, no, 1, 0	Положение по вертикали относительно верхнего края экрана в пикселях
top	число	Ширина окна в пикселях
width	число	

Примеры

window. open (“mypage.htm”, “NewWin”, “height=150, width=300”)
window. open (“mypage.htm”)
strfeatures = “top=100, left=15, height=250, width=300, location=no”
window. open (“www.amsu.ru”, strfeatures)

Вместо строки strfeatures можно использовать значение true, тогда указанный документ загружается в существующее окно, вытесняя предыдущий документ.

Метод window.open() возвращает ссылку на объект окна, сохранив которую, можно использовать позднее, например, при закрытии окна.

Для закрытия используют метод close(). Однако, выражение window.close() закрывает главное окно. Для закрытия других окон используют ссылки.

Пример

var str = window. open (“mypage.htm”, “моя страница”)
str.close()

Объект document является центральным в иерархической объектной модели. Он предоставляет всю информацию о HTML-документе с помощью коллекций и свойств и множество методов для работы с документами.

Коллекция document

all	все теги и элементы основной части документа
anchor	якоря (закладки) документа
applets	все объекты документа, включая встроенные элементы управления, графические элементы, апплеты, внедренные объекты
embeds	все внедренные объекты документа
forms	все формы на странице
frames	фреймы, определенные в теге <FRAMESET>
images	графические элементы
links	ссылки и блоки <AREA>
plugins	другое название внедренных документов
scripts	все разделы <SCRIPT> на странице
styleSheets	контейнерные свойства стиля, определенные в документе

Методы document

clear	очищает выделенный участок
close	закрывает текущее окно браузера
createElement	создает экземпляр элемента для выделенного тега
elementFromPoint	возвращает элемент с заданными координатами
execCommand	выполняет команду над выделенной областью
open	открывает документ
queryCommandEnabled	сообщает, доступна ли данная команда
queryCommandIndeterm	сообщает, если данная команда имеет неопределенный статус
queryCommandState	возвращает текущее состояние команды
queryCommandSupported	сообщает, поддерживается ли данная команда
queryCommandText	возвращает строку, с которой работает команда
queryCommandValue	возвращает значение команды, определенное для документа или объекта TextRange

write (writeln)	записывает текст и код HTML в документ, находящийся в указанном окне

Свойства document

Свойство	Атрибут	Назначение
activeElement		Активизирует активный элемент
alinkColor	ALINK	Цвет ссылок на странице
bgColor	BGCOLOR	Определяет цвет фона элемента
body		Ссылка только для чтения на неявный основной объект документа, определенный в теге <BODY>
cookie		Строка cokie-записи. Значение этого свойства приводит к записи на диск.
domain		Устанавливает или возвращает домен документа для его защиты или идентификации.
fgColor	TEXT	Устанавливает цвет текста переднего плана
lastModified		Дата последней модификации страницы, доступна как строка
linkColor	LINK	Цвет еще непосещенных гиперссылок на странице
location		Полный URL документа
parentWindow		Возвращает родительское окно для документа
readyState		Определяет текущее состояние загружаемого объекта
referer		URL страницы, которая вызвала текущую
selection		Ссылка только для чтения на дочерний для document объект selection
title	TITLE	Определяет справочную информацию элемента, используемую при загрузке и всплывающей подсказке
url	URL	URL-адрес документа клиента или в теге <META>
vlinkColor	VLINK	Цвет посещенных ссылок на странице

Объект window кроме дочерних объектов имеет свои методы, свойства, события.

Свойства window:

parent	возвращает родительское окно для текущего
self	возвращает ссылку на текущее окно
top	возвращает ссылку на главное окно
name	название окна
opener	окно, создаваемое текущим
closed	сообщает, если окно закрыто
status	текст, показываемый в строке состояния браузера
defaultStatus	текст по умолчанию строки состояния браузера
returnValue	позволяет определить возвращаемое значение для события или диалогового окна
client	ссылка, возвращаемая объект навигатора браузеру
document	ссылка только для чтения на объект окна document
event	ссылка только для чтения на глобальный объект event
history	ссылка только для чтения на объект окна history
location	ссылка только для чтения на объект окна location
navigator	ссылка только для чтения на объект окна navigator
screen	ссылка только для чтения на объект окна screen

Например,

window.status = «работает сценарий»

Свойство parent позволяет обратиться к объекту, расположенному в иерархии на одну ступень выше. Для перемещения на две ступени выше используют parent.parent. Для обращения к самому главному окну – окну браузера, используют свойство top.

Свойство status используют для вывода сообщений во время работы сценария. Например, window. status = “сценарий работает”

Методы window

open( )	открывает новое окно браузера
close( )	закрывает текущее окно браузера
showHelp( )	показывает окно подсказки как диалоговое
showModalDialog( )	показывает новое модальное(диалоговое) окно
alert( )	окно предупреждения с сообщением и кнопкой ОК
prompt( )	окно приглашения с сообщение, текстовым полем и кнопками ОК и Cancel (Отмена)
confirm( )	окно подтверждения с сообщением и кнопками ОК и Cancel
navigate( )	загружает другую страницу с указанным адресом
blur( )	убирает фокус с текущей страницы
focus( )	устанавливает страницу в фокус
scroll( )	разворачивает окно на заданную ширину и высоту
setInterval( )	указывает процедуре выполняться автоматически через заданное число миллисекунд
setTimeout( )	запускает программу через заданное количество миллисекунд после загрузки страницы
clearInterval( )	обнуляет таймер, заданный методом setInterval( )
clearTimeout( )	обнуляет таймер, заданный методом setTimeout( )
execScript( )	выполняет код сценария, по умолчанию Jscript

Рассмотренные выше методы позволяют работать с независимыми (немодальными) окнами. Для создания модального окна используется метод showModalDialog(). В качестве параметра данный метод принимает адрес документа (файла), имя окна, и строку свойств.

При работе с модальными окнами пользователь не может обратиться к другим окнам, в том числе и к главному. Окна, создаваемые методами alert(), promt(), confirm() являются модальными.

Одним из главных назначений сценариев в HTML-документе является обработка событий, таких как щелчок кнопки мыши по элементу документа, помещение указателя мыши на элемент, нажатие клавиши и др. Для одного и того же элемента можно определить несколько событий на которые он будет реагировать.

Сообщение о событии формируется в виде объекта, т.е. контейнера для хранения информации. Объект события в одном из свойств содержит ссылку на элемент, с которым связано данное событие (на кнопку, изображение и т.п.)

Обычно обработчики событий оформляются в виде функций, определения которых помещаются в контейнерный тег <SCRIPT>.

События window

onblur	выход окна из фокуса
onfocus	окно становится активным
onhelp	нажатие пользователем клавиши F1
onresize	изменение пользователем размеров окна
onscroll	прокрутка окна пользователем
onerror	ошибка при передаче

onbeforeunload	для сохранения данных перед выгрузкой страницы
onload	страница полностью загружена
onunload	непосредственно перед выгрузкой страницы

В случае открытия нескольких окон браузера, пользователь может переключаться между ними, переводя фокус с одного окна на другое. Эти действия инициируются программными событиями onblur и onfocus. Эти же действия можно вызвать, используя методы blur и focus.

Событие onerror происходит при ошибки загрузки страницы или ее элемента. Его можно использовать в программе при попытке вновь загрузить страницу. Например,

```
<SCRIPT>
function window.onerror( ) {
alert (“ Ошибка! Повтори попытку!”)
}
</SCRIPT>
```

События document

onafterupdate	окончание передачи данных
onbeforeupdate	перед выгрузкой страницы
onclick	при щелчке левой кнопкой мыши
ondblclick	при двойном щелчке левой кнопкой мыши
ondragstart	при возникновении перетаскивания
onerror	ошибка при передаче
onhelp	нажатие клавиши F1
onkeydown	нажатие клавиши
onkeypress	возникает при нажатии клавиши и продолжается при удержании клавиши в нажатом состоянии
onkeyup	пользователь отпускает клавишу
onload	при полной загрузке документа
onmousedown	при нажатии кнопки мыши
omousemove	при перемещении указателя мыши
onmouseout	когда указатель мыши выходит за границы элемента
onmouseover	когда указатель мыши входит на документ
onmouseup	пользователь отпускает кнопку мыши
onreadystatechange	возникает при изменении свойства readystate
onselectstart	когда пользователем впервые запускается выделенная часть документа

Динамическое изменение элементов документа

Элементы HTML-документа задаются тегами, большинство из которых имеют параметры (атрибуты). В объектной модели документа тегам соответствуют объекты, а атрибутам – свойства этих объектов. Названия свойств объектов, как правило, совпадают с названиями атрибутов, но записываются в нижнем регистре.

Наиболее удобный способ динамического изменения HTML-документа основан на использовании свойств innerText, outerText, innerHTML и outerHTML. С их помощью можно получить доступ к содержимому элемента. Изменяя значения перечисленных свойств можно частично или полностью изменить сам элемент. Например, можно изменить только надпись на кнопке, а можно превратить кнопку в изображение или Flash-анимацию.

Значением свойства innerText является все текстовое содержимое между открывающим и закрывающим тегами элемента. Внутренние теги игнорируются. Данные открывающего и закрывающего тегов соответствующего элемента также не входят.

В отличие от предыдущего свойство outerText включает в себя данные открывающего и закрывающего тегов. Таким образом, outerTextесть весь текст, содержащийся в контейнере, включая его внешние теги. Например, задан HTML-код:

<DIV ID = “my” >
<A HREF = ‘raznoe.htm’>
<IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B>
</A>
<\DIV>

Здесь свойства innerText и outerText для элемента, заданного контейнерным тегом <DIV>, совпадают:

document.all.my.innerText //значение равно – «Ссылка на раздел Разное»

При присвоении свойствам innerText и outerText новых значений нужно помнить, что если значения содержат теги, то они не интерпретируются, а воспринимаются как обычный текст.

Свойство innerHTML содержит внутренний HTML-код контейнера элемента. Присвоение этому свойству нового значения, содержащего HTML-код, приводит к интерпретации кода. Свойство outerText дополнительно включает внешние открывающие и закрывающие теги элемента.

Для приведенного HTML-кода значение document. all. my. innerHTML равно “<A HREF = ‘raznoe.htm’> <IMG SRC = ‘picture.jpg’> Ссылка на раздел <B> Разное <\B> </A>”/

Значение document. all. my. outerHTML – “<DIV ID = “my” > <A HREF = ‘raznoe.htm’> <IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B>
</A> <\DIV>”.

Если в сценарии выполнить выражение

document.all.my.innerHTML = “<BUTTON>Щелкни здесь</BUTTON>”

ссылка, изображение и текст будут заменены кнопкой с надписью «Щелкни здесь». При этом контейнерный тег “<DIV ID = “my” > сохранится. Если аналогичным образом использовать outerHTML, кнопка также появится, но уже без контейнера “<DIV ID = “my” >.

Свойства innerHTML и outerHTML могут применяться к элементам, заданным неконтейнерными тегами. Тогда innerHTML и outerHTML совпадают.

Для ускорения загрузки графики можно использовать следующие возможности JavaScript. Можно организовать предварительную загрузку изображений в кэш-память браузера, не отображая их на экране. Это особенно эффективно при начальной загрузке страницы. Пока изображения загружаются в память, оставаясь невидимыми, пользователь может рассматривать текстовую информацию.

Для предварительной загрузки изображения требуется создать его объект в памяти браузера. Это можно сделать следующим выражением:

myimg = new Image (ширина, высота)

Параметры должны соответствовать значениям атрибутов WIDTH и HEIGHT тега <IMG>, который используется для отображения предварительно загруженного изображения.

Для созданного в памяти объекта изображения можно создать имя или URL-адрес графического файла:

myimg.scr = “URL-адрес изображения”

что предписывает браузеру загрузить изображения без его отображения.

После загрузки в кэш-память всех изображений и загрузки всего документа можно сделать их видимыми. Для этого свойству src элемента <IMG> нужно присвоить значение этого же свойства объекта изображения в кэш-памяти. Например,

document. images[0]. scr = myimg. scr

Здесь слева указано свойство scr первого в документе элемента, соответствующего тега <IMG>, справа – свойство scr объекта изображения в кэш-памяти.

С помощью JavaScript можно через заданный интервал времени запускать код или функцию. При этом создается эффект одновременного (параллельного) выполнения вычислительных процессов.

Для организации повторения через заданный интервал выполнения некоторого выражения служит метод setInterval( ) объекта window:

setInterval(выражение, период, [, язык])

Первым параметром является строка, например вызов функции. Период указывается в миллисекундах. Третий параметр – необязательный, в котором указывается язык с помощью которого написано заданное выражение. По умолчанию – JavaScript.

Метод setInterval( ) возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем, например для прекращения выполнения процесса методом clearInterval( ).Например,

```
var pr = setInterval( “myfunc(), 100” )
if (confirm ( “Прервать процесс?”) )
clearInterval(pr)
```

Если требуется выполнить действие с некоторой временной задержкой, используется метод setTimeout( ), имеющий синтаксис аналогичный setInterval( ). Для отмены задержки процесса, запущенного setTimeout( ), используют clearTimeout( ).

Задание

Создать HTML-документ, расположив в нем список названий графических объектов, одно исходное отображение, две кнопки.

Щелчок на элементе списка должен приводить к изменению цвета элемента списка и отображению соответствующего графического элемента, и соответствующего ему тестового сопровождения.

При этом изображение кнопки должно быть также изменено.

Щелчок по первой кнопке через 5 секунд должен инициализировать функцию открытия документа в окне, заданного размера, определенного размера текстового поля и название. Окно должно содержать горизонтальные и вертикальные полосы прокрутки, размер окна не должен изменяться по желанию пользователя. Выведенный в окне текст должен быть синим на сером фоне, иметь выделенный заголовок, ссылки на другие объекты. По выбору продемонстрируйте по пять событий и свойств объектов window и document.

Это действие может быть отменено с помощью второй кнопки.

Лабораторная работа №5 **Работа с фреймами**

Фрейм – прямоугольная область окна браузера, в которую можно загрузить HTML-документ. Разбиение окна браузера на отдельные окна производится с помощью тега <FRAMESET>, внутрь которого вставляются теги <FRAME> с атрибутами, указывающими имя фрейма и адрес HTML-документа.

Пример

```
<HTML>
<FRAMESET ROWS= “30%, 70%”>
<FRAMESET SRC= “документ1.htm” NAME = “frame1” >
<FRAMESET SRC= “документ2.htm” NAME = “frame2” >
</FRAMESET>
</HTML>
```

Здесь применяется вертикальное расположение фреймов. Для горизонтального размещения фреймов вместо атрибута ROWS в теге <FRAMESET> использовать COLS.

Используя вложение тега <FRAMESET>, можно разбить уже имеющийся фрейм на два других.

При разбиении окна на фреймы и, в свою очередь, фрейма на другие фреймы возникают отношения родитель-потомок. Каждому из фреймов соответствует свой объект document. Обеспечение доступа к в иерархии объектов представлено на рисунке.

Так при обращении из одного фрейма-потомка к другому, необходимо помнить, что прямой связи между фреймами-потомками не существует. Поэтому сначала нужно обратиться к родительскому окну, а затем к его второму потомку:

parent. frame2. document. write(“ Привет от первого фрейма.“)

Можно изменить элемент одного фрейма из другого. Например, при щелчке на тексте в правом фрейме в левом изменится один из текстовых документов. Тогда документ в левом фрейме с именем LEFT:

```
<HTML>
Делай раз<BR>
Делай два
<H1 ID = "XXX"> Делай три </H1>
</HTML>
```

Документ в правом фрейме:

```
<HTML>
<SCRIPT>
function change( ) {
parent.LEFT.document.all.XXX. innerText = "Делай пять!!!!"
}
</SCRIPT>
<H1 onclick = "change( )"> Щелкни здесь</H1>
</HTML>
```

В теле функции change( ) происходит обращение к левому фрейму с именем LEFT (задается в установочном HTML-файле) через parent. Изменение элемента происходит за счет присвоения значения свойству innerText. Кроме данного свойства можно использовать outerText, innerHTML или outerHTML.

Важно, что изменения в одном фрейме по событию в другом происходят без перезагрузки HTML-документа.

Фреймы удобно использовать при создании навигационных панелей. В одном фрейме располагаются ссылки, а второй предназначен для отображения документов, вызываемых при активизации соответствующих ссылок.

Пример

```
// установочный файл frame.htm
<HTML>
<FRAMESET COLS = “25%, 75%”>
<FRAME SRC = “menu.htm” NAME = “menu” >
<FRAME SRC = “start.htm” NAME = “main” >
</FRAMESET>
</HTML>
```

Здесь start.htm – документ, который первоначально показан во фрейме main.

```
//menu.htm – навигационная панель
<HTML>
<SCRIPT>
function load (url) {
parent. main. location. href = url;
}
</SCRIPT>
<BODY>
```

<A HREF = “javascript:load(‘первый.htm’)”>Первый </A>
<A HREF = “второй.htm” TARGET = “main”> Второй </A>
<A HREF = “третий.htm” TARGET = “top”> Третий </A>
</BODY>
</HTML>

В примере окно браузера разделено на два фрейма. Первый из них играет роль навигационной панели, а второй – окна для отображения документов. Продемонстрированы два способа загрузки новой страницы во фрейм main. В первом случае используется функция load( ), параметр которой указывает, какой файл следует загрузить. При этом место, в которое он загружается, определяется самой функцией load( ). Во второй ссылке используется атрибут TARGET. В третьем ссылке демонстрируется, как можно избавиться от фреймов.

Для удаления фрейма с помощью load( ) достаточно записать:

parent. location. href = url

Атрибут TARGET в теге ссылки <A HREF> обычно применяется в случаях, когда требуется загрузить одну страницу в один фрейм. Язык сценариев используют при необходимости выполнения нескольких действий.

Для ссылок из родительского окна к объектам его дочерних фреймов можно использовать коллекцию frames. Обращение к определенному фрейму из этой коллекции возможно по индексу или по имени фрейма:

window. frames [индекс]

window. имя_фрейма

При обращении к объекту документа, загруженного во фрейм, следует сначала упомянуть объект document:

window. frames(0). document. all. Myinput. Value

window. LEFT. document. all. Myinput. Value

Ссылка из дочернего фрейма на родительский - осуществляется с использованием parent.

При использовании top следует учитывать, что создаваемый сайт может быть загружен в другой. Тогда объект top окажется объектом другого сайта. Поэтому лучше использовать parent для ссылок на вышестоящее окно или фрейм.

Ссылки top или self используют для предотвращения отображения сайта внутри фреймов другого сайта. Сценарий, выполняющий это, следует разместить в начале документа, например:

```
<SCRIPT>
if (top != self )
        top. Location = location
</SCRIPT>
```

т.е., ссылка на свойство top на верхнее окно, должна совпадать со ссылкой self на текущее окно.

Для вставки одного HTML-документа в тело другого средствами браузера служит контейнерный тег <IFRAME>:

<IFRAME SRC = “адрес документа” > </IFRAME>

Данный элемент представляет собой прямоугольную область с прокруткой или без. Такое окно называют плавающим фреймом. Данный документ можно позиционировать с помощью параметров таблицы стилей (тег <STYLE> или атрибут STYLE).

Плавающий фрейм аналогичен обычному фрейму. При создании он помещается в коллекцию frames. Среди его свойств широко используется align – выравнивание плавающего фрейма относительно окружающего содержимого документа. Его возможные значения:

absbotton – выравнивает нижнюю границу фрейма по подстрочной линии символов окружающего текста,

absmiddle – выравнивает середину границу фрейма по центральной линии между top и absbotton окружающего текста,

baseline – выравнивает нижнюю границу фрейма по базовой линии окружающего текста,

botton – совпадает с baseline (только IE)

left – выравнивает фрейм по левому краю элемента-контейнера,

middle – выравнивает воображаемую центральную линию окружающего текста по воображаемой центральной линии фрейма,

right – выравнивает фрейм по правому краю элемента-контейнера,

texttop – выравнивает верхнюю границу фрейма по надстрочной линии символов окружающего текста,

top – выравнивает верхнюю границу фрейма по верхней границе окружающего текста.

Задание

Вариант 1. Окно браузера поделить на два фрейма. В левом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в левом окне по щелчку мыши на миниатюре изображения в правом фрейме. В левом фрейме дополнительно организовать навигационную модель. Создать новый HTML-документ и вставить его в ранее созданный, как плавающий вертикальный фрейм, выравнивая нижнюю границу фрейма по базовой линии окружающего текста.

Вариант 2. Окно браузера поделить на три фрейма. В левом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в среднем окне по щелчку мыши на миниатюре изображения в правом фрейме. В левом фрейме дополнительно организовать навигационную модель. Создать новый HTML-документ и вставить его в ранее созданный, как плавающий вертикальный фрейм, выравнивая воображаемую центральную линию окружающего текста по воображаемой центральной линии фрейма.

Лабораторная работа №6 **Простые визуальные эффекты**

Смена изображений

Для смены одного изображения на другое достаточно с помощью сценария заменить значение атрибута SCR тега <IMG>. Например:

```
<HTML>
<IMG ID = “myimg” SRC= ‘pict1.gif ’
onclick = “document.all.myimg.src = ‘pict2.gif’ ”>
</HTML>
```

Здесь смена изображения из файла pict1.gif на изображение из файла pict2 происходит при первом щелчке на нем. Последующие щелчки не приведут к видимым изменениям, поскольку второе изображение будет заменяться им же. Чтобы при повторном щелчке происходила замена изображения на предыдущее необходимо создать переменную-триггер (флаг), принимающий одно из двух возможных значений, по которому можно определить, какое из двух значений надо отобразить.

```
<HTML>
<IMG ID = “myimg” SRC= ‘pict1.gif ’
onclick = “ imgchange( )”>
<SCRIPT>
var flag=false
function  imgchange( ) {
if (flag) document. all. myimg. src = “pict1.gif ”
        else document. all. myimg. src = “pict2.gif ”
```

```
flag=!flag
}
</SCRIPT>
</HTML>
```

Цветовые эффекты

Задача изменения цвета кнопки при наведении на нее указателя мыши и возвращения в первоначальное состояние при удалении указателя с кнопки может быть решена следующим образом:

```
<HTML>
<STYLE>
mystile {font-weight.bold; background-color: a0a0ao} // серый цвет кнопок
</STYLE>

<FORM onmouseover = “colorchange ( ‘yellow’)” onmouseout
= “colorchange (‘a0a0ao’)”>
<INPUT TYPE = “BUTTON” VALUE = “Кнопка” CLASS = “mystile”
onclick = “alert( ‘Вы нажали кнопку’)” >
</FORM>
<SCRIPT>
function colorchange (color){
if (event. scrElement.type = = “button”)
        event. scrElement. style. backgroundColor = color;
}
</SCRIPT>
</HTML>
```

Функция colorchange( ) проверяет, является ли инициатор события объектом типа button. Если это так, то цвет кнопки меняется. Без этой проверки менялся бы не только цвет кнопок, но и текста.

Аналогичным способом можно изменять цвет фрагментов текста. Но в этом случае текст должен быть заключен в контейнер, например в теги <P>, <B>, <I>, <DIV>.

Можно создать прямоугольную рамку, окаймляющую текст, которая периодически изменяет цвет. Рамка создается тегами одноячеечной таблицы с заданием нужных атрибутов и параметров стиля:

```
<TABLE ID= “tab” BORDER=1 WIDH=200 style= “border:10 solid : red”>
<TR><TD> Доброе утро! </TR></TD>
</TABLE>
```

Функция изменения цвета:

```
<SCRIPT>
function flash( ) {
if ( !document.all) return null;
if (tab.style.borderColor = = ‘red’)   tab.style.borderColor = ‘yellow’
else tab.style.borderColor = ‘red’;
}
setInterval (“flash”, 500);                //мигание рамки с интервалом 500 мс
</SCRIPT>
```

Объемные заголовки

Объемные заголовки часто используются на веб-страницах. Идея создания объемного заголовка состоит в наложении нескольких надписей с одинаковым содержанием с некоторым сдвигом по координатам. Наилучший эффект достигается путем подбора цве-

тов надписей (игрой света и тени) с учетом цвета фона. Для этого используют библиотеку стилей. Функция, создающая заголовок с заданными параметрами:

```
        function d3 (text, x, y, tcolor, fsize, fweight, family, zind) {
/*      text – текс заголовка
        x – горизонтальная координата (left)
        y – вертикальная координата (top)
        tcolor – цвет переднего плана
        fsize – размер шрифта (пт)
        fweight – вес (толщина шрифта)
        family – название семейства шрифтов
        zind z-Index  */
if (!text) return null // если текст не указан ничего не выполняется
//значение параметров по умолчанию
if (!x) x=0
if (!y) y=0
if (!tcolor) tcolor='00aaff '
if (!fsize)  fsize=36
if (!fweight)  fweight =800
if (!family) ffamily='arial'
// внутренние настройки
var sd=5, hd=2
var xzind= " "
if (zind) xzind= "; - Index:"+zind
var xstyle ='font-family:' + family + ';font-size:' + fsize + ';font-weight:' + fweight + ';'
var xstr = '<DIV STYLE = "position: absolute; top:'  + (y +sd ) + '; left :' +
( x + sd ) +  xzind + ' ">'
xstr+='<P styl e  = " ' + xstyle + 'color: darked">' + text + '</P></DIV>'
xstr+= '<DIV STYLE = " position: absolute; top:' + y  + '; left :' +
x  +  xzind + ' ">'
xstr+='<P styl e  = " ' + xstyle + 'color: silver">' + text + '</P></DIV>'
xstr+= '<DIV STYLE = " position: absolute; top:' + ( y + hd )  + '; left :' +
(x +hd)  +  xzind + ' ">'
xstr+='<P styl e  = " ' + xstyle + 'color:' + tcolor +   '">' + text + '</P></DIV>'
document.write(xstr)            //запись в документ
}
```

Параметр z-Index позволяет установить слой, в котором находится заголовок, и тем самым указать, будет ли заголовок располагаться над или под другим видимым элементом документа. Элементы с более высоким значением z-Index находятся над элементами, у которых z-Index меньше. Перекрытие элементов с одинаковыми значениями z-Index определяется порядком их следования в HTML-документе.

Вызов приведенной выше функции может выглядеть так:

```
d3 ("это не графика, это просто стиль текста", 50, 50, 'blue', 72, 800, 'times')
```

Задание

Выполнить следующие действия на веб-странице:

1. Создать программу для работы с галереей миниатюр. При щелчке кнопкой мыши по миниатюре изображение должно увеличиваться, а затем при щелчке на увеличенном изображении оно должно уменьшаться. Доработайте приведенную в тексте функцию функцию imgchange( ). Для решения этой задачи потребуется массив флагов и функция обработчик, определяющая на каком именно изображении произошел щелчок:

```
var p1=new Array ("pict1.gif" , … )  //массив имен исходных файлов
var p2=new Array ("pict2.gif" , … )  //массив имен замещающих файлов
```

```
//формирование тегов, описывающих изображения
var xstr =  “ “
for (i=0; i<p1.length; i++)
xstr+= ‘<IMG ID = “i’ + i +’ ” SRC = “ ’+ p1[i]+’ ” onclick = “imgchange( )” >’
}
document.write(xstr)            // запись в документ
```

2.Выполнить замену фрагмента текста с черного на красный при наведении на него указателя мыши.

3. Выделите фрагмент текста мигающей трехцветной рамкой.

4.Создать эффект динамического изменения цвета ссылок. Различать цвета мерцания использованных и неиспользованных ссылок. (Множество цветов задать массивом. Использовать свойства linkColor и linkColor объекта document). Для изменения цвета случайным образом можно использовать метод random( ) (счетчик случайных чисел) встроенного объекта Math. Если требуется получить случайное число x, лежащее в интервале от A до B, то x=A+(B – A)*Math. random( )

5. Создать три объемных заголовка, поэкспериментировав со значениями внутренних параметров sd, hd, а также с заданием параметров по умолчанию.

Лабораторная работа №7 **Применение фильтров**

С помощью фильтров каскадных таблиц стилей можно получить разнообразные эффекты: постепенное появление или исчезновение рисунка, плавное преобразование одного изображения в другое, задание степени прозрачности и др.

Фильтр следует понимать как некий инструмент преобразования изображения, взятого из графического файла и вставленного в HTML – документ с помощью тега <IMG>. Следует иметь в виду, что фильтры работают только в IE4+.

Фильтры можно применять не только к графическим объектам, но и к текстам, текстовым областям, кнопкам.

Прозрачность

С помощью фильтра alpha можно установить прозрачность графического объекта. Сквозь прозрачные графические объекты видны нижележащие изображения. Прозрачность имеет несколько вариантов градиентной формы. Например, она может увеличиваться от центра к краям изображения

Фильтр alpha задается с помощью каскадной таблицы стилей и имеет ряд параметров. В примере для графического изображения стиль определяется с помощью атрибута STYLE:

```
<IMG ID = “myimg” SCR = “ pict. gif ”
STYLE = “position: absolute; top:10; left: 50;
filter: alpha (opacity = 70, style = 3)”>
```

Здесь целочисленный параметр opacity определяет степень непрозрачности. Значение 0 соответствует полной прозрачности изображения, а 100 – полной непрозрачности. Параметр style задает градиентную форму распределения прозрачности по изображению как целое число от 0 до 3. По умолчанию значение параметра равно 0, и градиент не применяется. Фильтр имеет и другие параметры, определяющие прямоугольную область изображения, к которому применяется фильтр. По умолчанию фильтр применяется ко всему изображению.

Фильтр можно определить в каскадной таблице силей внутри контейнерного тега <STYLE>:

```
<HTML>
<STYLE>
#myimg{position: absolute; top:10; left: 50; filter: alpha (opacity = 70, style = 3)}
```

```
</STYLE>
< IMG ID = “myimg” SCR = “ pict. gif ”
</HTML>
```

Доступ к свойствам фильтра в сценарии:

```
document. all. id_изображения. filters [“имя_фильтра”]. параметр = значение
```

Для рассмотренного примера это выражение имеет вид:

```
document. all. myimg. filters [“alpha”]. opacity = 30
```

Для остальных параметров alpha аналогично.

Для IE5.5+ можно использовать другой синтаксис, в котором в каскадной таблице стилей задается ссылка на специальный компонент и имя фильтра:

```
#myimg { filter: progid: DXImageTransform . Microsoft . alpha
(opacity = 70, style = 3)}
```

Тогда доступ к свойствам фильтра:

```
document.all.myimg.filters [“DXImageTransform. Microsoft alpha”]. opacity = 30
```

Трансформация

Фильтр alpha статический. Существуют и динамические фильтры: apply( ) – фиксирует изображение, play( ) – трансформирует, revealtrans( ) –преобразовывает изображение, stop( ) останавливает процесс преобразования при необходимости.

Фильтр revealtrans( ) имеет параметры: duration – длительность преобразования в секундах (число с плавающей точкой) и transition – тип преобразования (целое от 0 до 23).

Для эффекта появления изображения можно воспользоваться фрагментом, который происходит после загрузки документа, т.е. по событию onload:

```
<HTML>
<BODY onload = “transform()” >
<IMG ID = “myimg” SRC = “ pict. gif ” STYLE = “position: absolute; top:10;
left: 50; visibility = “hidden” filter: revealtrans (duration = 3, transition =12)” >
//transition =12 соответствует плавной трансформации
</BODY>
<SCRIPT>
function transform ( ) {                                    //появление изображения
document.all.myimg.style. visibility = “hidden”     // изображение невидимо
myimg. filters ( “revealtrans”). apply( )
myimg.style. visibility = “visible”
myimg. Filters ( “revealtrans”). play( )                // выполняем преобразования
}
</SCRIPT>
</HTML>
```

Для замены одного изображения на другое необходимо установить начальное и конечное изображение путем присвоения нужных значений свойству src объекта, соответствующего изображению, например фрагментом:

```
document.all.myimg.src = “pict2. gif ”
```

Рассмотренный синтаксис воспринимается браузерами IE4+. Для IE5.5+ в каскадной таблице стилей задается ссылка на специальный компонент и имя фильтра. Так для трансформации изображения по щелчку мыши на графическом объекте в другое, и обратно, можно воспользоваться программой:

```
<HTML>
<STYLE>
#myimg{position: absolute; top:10; left: 50; filter: progid: DXImageTransform . Microsoft revealtrans (duration = 3, transition = 12)}
</STYLE>
< IMG ID = “myimg” onclick = “transform( )” SCR = “ ear. gif ”>
```

```
<SCRIPT>
 function transform( ) {
//фиксация исходного изображения
myimg. filters (“DXImageTransform . Microsoft revealtrans”). apply ( )
//определение конечного изображения
if (document. all. mying. src. index0f (“ear”)!= -1)
document. all. mying. src = “s.gif”
        else document. all. mying. src = “ear.gif”
//выполняем преобразование
myimg. filters (“DXImageTransform . Microsoft revealtrans”). play ( )
}
</SCRIPT>
</HTML>
```

В браузере IE5.5+ возможно прменение фильтра basicimage, с помощью которого изображение можно повернуть на угол, кратный 90 градусам, задать прозрачность, зеркально отразить и др.

Задание

В HTML-документе из предыдущей лабораторной работы применить к рисункам методы трансформации и изменения прозрачности изображения и фона.

Лабораторная работа №8 **Динамическое изменение элементов документа**

Элементы HTML-документа задаются тегами, большинство из которых имеют параметры (атрибуты). В объектной модели документа тегам соответствуют объекты, а атрибутам – свойства этих объектов. Названия свойств объектов, как правило, совпадают с названиями атрибутов, но записываются в нижнем регистре.

Наиболее удобный способ динамического изменения HTML-документа основан на использовании свойств innerText, outerText, innerHTML и outerHTML. С их помощью можно получить доступ к содержимому элемента. Изменяя значения перечисленных свойств можно частично или полностью изменить сам элемент. Например, можно изменить только надпись на кнопке, а можно превратить кнопку в изображение или Flash-анимацию.

Значением свойства innerText является все текстовое содержимое между открывающим и закрывающим тегами элемента. Внутренние теги игнорируются. Данные открывающего и закрывающего тегов соответствующего элемента также не входят.

В отличие от предыдущего свойство outerText включает в себя данные открывающего и закрывающего тегов. Таким образом, outerTextесть весь текст, содержащийся в контейнере, включая его внешние теги. Например, задан HTML-код:

```
<DIV ID = “my” >
<A HREF = ‘raznoe.htm’>
<IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B>
</A>
<\DIV>
```

Здесь свойства innerText и outerText для элемента, заданного контейнерным тегом <DIV>, совпадают:

```
document.all.my.innerText    //значение равно – «Ссылка на раздел Разное»
```

При присвоении свойствам innerText и outerText новых значений нужно помнить, что если значения содержат теги, то они не интерпретируются, а воспринимаются как обычный текст.

Свойство innerHTML содержит внутренний HTML-код контейнера элемента. Присвоение этому свойству нового значения, содержащего HTML-код, приводит к интерпре-

тации кода. Свойство outerText дополнительно включает внешние открывающие и закрывающие теги элемента.

Для приведенного HTML-кода значение document. all. my. innerHTML равно "<A HREF = 'raznoe.htm'> <IMG SRC = 'picture.jpg> Ссылка на раздел <B> Разное <\B> </A>"/

Значение document. all. my. outerHTML – "<DIV ID = "my" > <A HREF = 'raznoe.htm'> <IMG SRC = 'picture.jpg> Ссылка на раздел <B> Разное <\B>
</A> <\DIV>".

Если в сценарии выполнить выражение

document.all.my.innerHTML = "<BUTTON>Щелкни здесь</BUTTON>"

ссылка, изображение и текст будут заменены кнопкой с надписью Щелкни здесь. При этом контейнерный тег "<DIV ID = "my" > сохранится. Если аналогичным образом использовать outerHTML, кнопка также появится, но уже без контейнера "<DIV ID = "my" >.

Свойства innerHTML и outerHTML могут применяться к элементам, заданным неконтейнерными тегами. Тогда innerHTML и outerHTML совпадают.

Для ускорения загрузки графики можно использовать следующие возможности JavaScript. Можно организовать предварительную загрузку изображений в кэш-память браузера, не отображая их на экране. Это особенно эффективно при начальной загрузке страницы. Пока изображения загружаются в память, оставаясь невидимыми, пользователь может рассматривать текстовую информацию.

Для предварительной загрузки изображения требуется создать его объект в памяти браузера. Это можно сделать следующим выражением:

myimg = new Image (ширина, высота)

Параметры должны соответствовать значениям атрибутов WIDTH и HEIGHT тега <IMG>, который используется для отображения предварительно загруженного изображения.

Для созданного в памяти объекта изображения можно создать имя или URL-адрес графического файла:

myimg.scr = "URL-адрес изображения"

что предписывает браузеру загрузить изображения без его отображения.

После загрузки в кэш-память всех изображений и загрузки всего документа можно сделать их видимыми. Для этого свойству src элемента <IMG> нужно присвоить значение этого же свойства объекта изображения в кэш-памяти. Например,

document. images[0]. scr = myimg. scr

Здесь слева указано свойство scr первого в документе элемента, соответствующего тега <IMG>, справа – свойство scr объекта изображения в кэш-памяти.

С помощью JavaScript можно через заданный интервал времени запускать код или функцию. При этом создается эффект одновременного (параллельного) выполнения вычислительных процессов.

Для организации повторения через заданный интервал выполнения некоторого выражения служит метод setInterval( ) объекта window:

setInterval(выражение, период, [, язык])

Первым параметром является строка, например вызов функции. Период указывается в миллисекундах. Третий параметр – необязательный, в котором указывается язык с помощью которого написано заданное выражение. По умолчанию – JavaScript.

Метод setInterval( ) возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем, например для прекращения выполнения процесса методом clearInterval( ).Например,

```
var pr = setInterval( "myfunc(), 100" )
if (confirm ( "Прервать процесс?") )
clearInterval(pr)
```

Если требуется выполнить действие с некоторой временной задержкой, используется метод setTimeout( ), имеющий синтаксис аналогичный setInterval( ). Для отмены задержки процесса, запущенного setTimeout( ), используют clearTimeout( ).

Задание

1. Создать HTML-документ, в котором отображается список названий графических объектов и одно исходное отображение. Щелчок на элементе списка должен приводить к отображению соответствующего элемента.
2. Создать в HTML-документе две кнопки. Щелчок по кнопке ПУСК открывает через 5 секунд новое окно и загружает в него некоторый документ. Это действие может быть отменено с помощью кнопки ОТМЕНА.

Лабораторная работа №9
Знакомство со средой программирования Borland C++ BUILDER

Все пользовательские программы в среде Borland C++ BUILDER оформляются в виде проектов. Действия по управлению объектами осуществляет программный специальный программный комплекс – Менеджер проектов.

Интерфейс Borland C++BUILDER представлен следующими окнами:

1. главное окно (Project1);
2. окно формы (Form1);
3. окно Инспектора Объектов (object Inspector);
4. окно Кода программы (Unit1.cpp).

Команда File→New Application открывает новую, так называемую, форму – заготовку для помещения обработчиков событий компонентов, которые будут размещаться в форме из набора (палитры) и реакции на события. Реакции в программах задаются обработчиками событий.

C++ BUILDER связывает с каждым приложением, оформленным как проект, следующие исходные файлы:

Unit1.cpp – текст программы;

Unit1.h – интерфейсный модуль с объявлениями компонентов, расположенных в данной форме, глобальных переменных, функций и т.д.;

Unit1.dfm – хранит описания формы и всех расположенных в ней компонентов. Это файл поддерживается и обновляется средой автоматически;

Project1.cpp – главная управляющая программа проекта – проектный файл. Любая новая форма автоматически включается в этот файл. С его помощью вызывается, выполняется и завершается код проекта;

Project1.res – файл ресурсов проекта. В нем хранятся значки, используемые в проекте, заданные как разработчиком, так и системой.

Форма – окно будущей программы, имеющее заголовок и системные кнопки. На форме размещаются компоненты, формирующие программу.

Помещение компонента в форму осуществляется следующими способами:

1. найти нужный компонент на вкладках палитры, щелкнуть по нему мышью, перевести указатель мыши на нужное место окна дизайнера форм, снова щелкнуть мышью.
2. дважды щелкнуть на нужном компоненте на вкладке;
3. если требуется несколько экземпляров одного компонента, нажать Shift и щелкнуть на нужном компоненте палитры, затем указывая необходимые места формы. Для прекращения – дважды щелкнуть на компоненте вкладки.

Одновременно с открытием новой формы создается новый элемент, окно которого появляется на экране вместе с формой – Инспектор объекта. Он также создается для каждого помещенного в форму компонента. Инспектор объекта позволяет увидеть основные свойства (Properties) и события (Events) объекта. Полный перечень свойств, событий и ме-

тодов можно увидеть по справочной службе Help (F1) при выделенном компоненте. Наиболее часто используемым событием является OnClick – щелчок мыши.

Инспектор объектов имеет свое контекстное меню, которое открывается правой кнопкой мыши.

Основными свойствами компонентов являются:

Name – имя объекта, в котором можно использовать латинские буквы и арабские цифры. Это свойство является обязательным;

Font – параметры шрифта;

Caption – заголовок;

Visible – видимость (возможные значения true и false);

Color – цвет объекта;

Width – ширина объекта;

Height – высота объекта.

Окно кода первоначально перекрыто окном формы. Для редактирования текста модуля требуется:

- открыть вкладку с именем требуемого модуля в окне Редактора кода (для переключения между окном Редактора и формой используют клавишу F12) или открыть диалоговое окно командой View|Units и выбрать в нем требуемый модуль.

- поместить курсор Редактора в нужное место текста и редактировать по правилам стандартных текстовых редакторов.

У Редактора кодов существует свое контекстное меню, открывающееся правой кнопкой мыши. Вкладки этого окна позволяют изменять параметры настройки редактора: устанавливать общие настройки(автоматический отступ, вставка и пр.), изменять режима ввода на экран (форма курсора, ширина полей, шрифт), управлять цветом и шрифтом для выделения синтаксических конструкций (ключевых слов, комментариев и пр.), организовывать различные подсказки с помощью так называемого «суфлера кодов». Суфлер кода помогает быстрее и правильнее набирать текст программы, выдавая оперативную информацию.

Для размещения компонента на форме нужно среди палитры компонентов щелкнуть мышью на выбранном, после чего щелкнуть в том месте формы, где предполагается размещение объекта. Удалить компонент можно с помощью клавиши Delete.

К основным компонентам относятся: **TLabel –** метка, служащая для отображения текста на экране; **TPanel** – контейнер общего назначения; **TButton** – командная кнопка, позволяющая выполнять некоторые действия при ее нажатии при выполнении программы.

Компонент TLabel выводит в форму текст, который пользователь в режиме исполнения приложения не может редактировать. Если цвет метки совпадает с цветом фона, то при сокращении надписи до нулевой длины, она может исчезнуть. Тогда для ее розыска нужно развернуть список компонент в Инспекторе Объектов.

Свойства TLabel

Aligment – задает способ выравнивания текста;

FocusControl – имеет раскрывающийся список, в который попадают компоненты, которые могут быть связаны с меткой и получать от нее фокус ввода. Для этого после выбора компонента, в ее тексте (свойство Caption) указывается символ & перед символом, который станет горячей клавишей при исполнении приложения.

Layout – размещение текста, заданного в свойстве Caption, в поле метки.

Transparent – если некоторый компонент расположен под меткой, он может быть невидимым. Чтобы этого не произошло, надо сделать метку прозрачной (трансарентной).

WordWrap – если это свойство равно true, а свойство AutoSize – false, все слова текста в свойстве Caption станут располагаться в разных строках.

Компонент TPanel используется в качестве базового класса для определения объектов, которые включают в себя другие объекты. При перемещении панели компоненты перемещаются вместе с ней. Перечислим лишь основные ее свойства, события и методы.

Свойства TPanel

BevelInner – определяет стиль внутренней кромки;

BevelOuter – стиль внешней кромки;

BevelWidth – ширина кромок в пикселах;

BorderWidth – расстояние в пикселах между внутренней и внешней кромками;

Align – размещает панель в форме в соответствии со значениями этого свойства;

Aligment – определяет выравнивание названия панели относительно самой панели;

Ct3D – появление панели в трехмерном или двумерном виде.

Методы TPanel

CanFocus( ) – показывает, может ли компонент получить фокус, т.е. стать активным; возвращает true, если свойства Visible и Enabled компонента и его родителя имеют значение true.

Focused( ) – определяет, имеет ли компонент входной фокус;

Hide( ) – делает компонент невидимым, устанавливая его свойство Visible в false.

SetFocus( ) – делает компонент активным;

Show( ) – показывает компонент, если он до этого был спрятан.

Редактор кода, cpp-модуль h-файл

Когда открывается новая форма, к ней создается два файла: один для программ – обработчиков событий (его расширение cpp), а другой с расширением h. Обоим файлам система присваивает имена по умолчанию, так для первой формы Unit1.cpp и Unit1.h. Файл Unit1.cpp можно редактировать, внося в него требуемые команды. (его можно увидеть в Инспекторе Объекта по нажатию клавиши F12).Cpp-модуль содержит указатель на класс TForm, что позволяет обращаться к членам класса. Формы автоматически нумеруются в порядке их создания. Любая форма, вставляемая в проект, получает имя Formi, i=1, 2, 3, … То же происходит с компонентами формы: кнопками, метками и пр.

Обращения к элементам класса происходит в виде:

имя объекта->имя элемента класса

Например,

```
Form1->Button1
Form1->Button1->Caption
```

Имена обработчиков событий также даются системой по определенному правилу: к имени компонента добавляется имя события.

Чтобы нарисовать что-либо в форме, можно воспользоваться событием OnPaint, которое возникает всякий раз, когда рисунок в форме изменяется. Общий вид обработчика этого события формы следующий:

```
{
int x,y;
x=y=0; //начальные координаты графика
Form1->Canvas->MoveTo(0,Height); // начальная точка графика
Form1->Canvas->Pen->Color=clRed; // цвет графика – красный
// в цикле рисуем точки графика
while (x< Form1->Width && y< Form1->Height )
{
x++;              // увеличиваем x на единицу
y=f(x); // задаем значение функции – его нужно конкретизировать, напр. y=x*x;
Form1->Canvas->LineTo(x, Height –y); // рисуем точку линии
}
}
```

Задание

1. Задайте свойства для формы. Разместите на форме TLabel , укажите произволь-

ный текст, задав шрифт, его размер, цвет. Ниже расположите компонет TPanel, задав внешнюю и внутреннюю ширину рамки так, чтобы панель была вдавленной. Самостоятельно изучите методы и свойства TButton. На панели разместите три кнопки с надписями произвольных цветов. Создайте поочередно события на реакцию нажатия каждой кнопки. Должен меняться в соответствии с надписью цвет формы, а также текстовая надпись, ее цвет, размер. Дополнительно создайте окна сообщения с условием задачи и фамилией разработчика.

Лабораторная работа №10
Компоненты TEdit, TMainMenu, TMemo, TPopupMenu

Компонент TEdit задает в форме однострочное редактируемое окно для ввода и вывода данных.

Свойства TEdit

AutoSelect – значение true, когда компонент получает фокус ввода на весь текст, false – курсор ввода остановится в начале текста.

BorderStyle – наличие или отсутствие окантовки;

CharCase – задает регистр ввода текста: EcLowerCase – текст преобразуется в символы нижнего регистра, EcUpperCase – верхний регистр; EcNormal – оба регистра.

HideSelection – значение true, если при перемещении фокуса ввода выделенный текст не меняется подсветки, false – подсветка исчезает при выделении другого компонента.

PasswordChar – при записи вида Edit->PasswordChar='*'; вместо вводимых символов в поле Edit высветятся звездочки.

ReadOnly – при значение true пользователь не может менять текст в поле компонента.

Text – поле ввода-вывода текста.

Методы TEdit

Clear( ) – очищает поле компонента, удаляя весь текст.

ClearSelection( ) – удаляет выделенный текст.

ClearUndo( ) – очищает буфер Undo, после чего невозможна отмена предыдущих изменений текста.

CopyToClipBoard( ) – удаляет выделенный текст, копируя его в буфер памяти.

CutToClipBoard( ) – вставляет текст из буфера памяти, заменяя выделенный текст.

PasteFormClipBoard( ) – вставляет текст из буфера памяти, заменяя выделенный текст.

SelectAll( ) – выделяет весь текст в поле компонента.

Undo( ) – отменяет все изменения после последнего вызова ClearUndo( ). Если этот метод не вызывался – отменяет все изменения.

CanFocus( ) – определяет, может ли компонент получить фокус ввода.

SetFocus( ) – делает компонент активным.

GetTextLen( ) – возвращает длину текста в компоненте.

Hide( ) – делает компонент невидимым.

Show( ) – делает компонент видимым.

Компонент TMemo – многострочное редактируемое поле. TMemo – массив пронумерованных текстовых строк типа Tstrings. Для обращения к его i-ой строке следует писать: Memo->Lines->Strings[i];

Количество строк: Memo->Lines->Count;

Длина i-ой строки: Memo->Lines->Strings[i].Lengs;

Очищение текстового поля: Memo->Lines->Clear( );

Свойства TMemo

Lines – открывает Редактор текстовых строк, в котором можно набрать текст как в

обычном текстовом редакторе.

MaxLength – задает максимальную длину строки текста.

ScrollBars – задает полосы прокрутки окна Memo-поля.

WantReturns – при значении true можно ли вставлять символы возврата в текст.
События и методы аналогичны соответствующим событиям и методам компонента TEdit.

Компонент TMainMenu создает главное меню приложения и с помощью него управляет работой всех его частей, т.е. запускает разные части приложения на выполнение отдельными командами, обеспечивает выход из приложения.

Меню добавляется в форму перенесением его значка из палитры компонентов. С формой меню связывается через свойство формы Menu. После формирования меню, запуск приложения будет сопровождаться появлением в верхней левой части формы строки, содержащей главные опции меню. Главные опции могут распадаться на детальные команды, расположенные сверху вниз.

Для формирования опций меню используют Дизайнер меню, который открывается через команду MenuDesigner в контекстном меню компонента, либо двойным щелчком по компоненту.

В левом верхнем углу окна Дизайнера меню располагается синие поле, предназначенное для названия первой опции. После задания его названия (свойство Caption), справа появится новое серое поле – заготовка для следующей опции. Каждая опция – новый объект и имеет собственный Инспектор объекта.

Если данная опция – последняя в иерархии, т.е. является исполнительной, нужно с помощью ее Инспектора объекта перейти на вкладку Events и дважды щелкнуть в поле события OnClick. В открывшийся обработчик события нужно вписать требуемые действия. После этого можно продолжить формирование следующей опции горизонтальной строки.

Свойства TMainMenu

Images – обеспечивает появление изображений слева от названия команд и подменю.

Items – массив, хранящий все опции меню. В поле этого свойства содержится кнопка с многоточием, открывающая дизайнер меню. Это свойство само является указателем на класс TMenuItem и может использоваться для обращения к свойствам и методам этого класса. Кроме того, оно является массивом, хранящим все опции меню. Например, количество строк меню с именем MainMenu1 можно посчитать так: MainMenu1-> Items->Count. К опции можно обратиться по номеру: MainMenu1->Items[i]

OwnerDraw – задает возможность рисования некоторого объекта при выборе опции меню, по умолчанию имеет значение false. При установление true в обработчике события OnDrwItem будет появляться рисунок

Основные свойства опций TMainMenu

Bitmap – позволяет выбрать значок в открывающемся диалоговом окне. Значок появится слева от названия опции.

Checked – осуществляет контроль выбора данной команды меню: в обработчике события OnClick этой опции надо присвоить свойству Checked значение true. Если свойство RadioItem некоторой опции имеет значение false, то при щелчке мышью на этой опции слева от ее названия появится галочка. Если же наоборот – опция не будет отмечена. При Checked= RadioItem= true будет всегда помечена только одна опции из всех, у которых свойство GroupIndex имеет одинаковое значение. Пометка будет сделана в виде жирной точки слева от названия опции.

Shortcut – в раскрывающемся списке надо выбрать комбинацию клавиш, которая в режиме исполнения приложения заменит нажатие мышью на опцию. Действие этих клавиш возможно только при открытом меню и распространяется только на команды подменю.

Компонент TPopupMenu может быть связан с любым другим компонентом (кнопкой, формой и пр.) у которого имеется свойство PopupMenu (всплывающее меню). При помещении TPopupMenu в форму, его имя будет видно в любом из компонентов формы со свойством PopupMenu. Если меню связано с формой, оно появляется при нажатии правой кнопки мыши в активной форме.

Свойства TPopupMenu

Многие свойства совпадают со свойствами TMainMenu. Свойство Item также является массивом строк-опций меню и указателем на класс TMenuItem. Обращаться к опции можно, указав ее порядковый номер в меню: PopupMenu->Items[i] Нумерация начинается с нуля, а общее количество считается как PopupMenu->Items->Count. Для обращения к вложенным опциям надо указать имя объекта – основной опции.

Aligment – задает расположение меню, при его появлении при нажатии правой кнопки мыши.

AutoPopup – при значении false появление выпадающего меню надо устанавливать программно с помощью метода Popup, иначе оно автоматически привязывается к компоненту, с которым связано.

MenuAnimation – задает анимационный эффект появления меню на экране.

Trackbutton – задает кнопку мыши, при нажатии которой меню появляется.

Задание

Создать приложение согласно заданиям индивидуального варианта. В каждом приложении должна обеспечиваться регистрация пользователя в приложении.

Регистрация пользователя должна реализовываться с помощью дополнительной формы, в которой расположены:

- компоненты TEdit (для ввода имени и пароля пользователя);
- кнопка «Завершение регистрации», которая прекращает регистрацию вместе с приложением;
- экземпляр компонента TMemo, в котором заданы истинные имя пользователя и пароль, с которыми надо сверять вводимые значения. На этапе разработки данный компонент невидимый.

Форма регистрации должна иметь статус главной, т.е. при запуске проекта она вызывается первой. При условии верной регистрации должен осуществляться вызов формы самого приложения. В четных вариантах при регистрации в имени и пароле могут присутствовать только символы нижнего регистра, в нечетных - верхние.

Расчеты в вариантах производить с помощью циклических алгоритмов.

Лабораторная работа № 11
Компоненты TListBox, TComboBox, TMaskEdit

Компонент TListBox создает прямоугольную область, в которой отображается список текстовых строк, которые можно добавлять, удалять, выбирать. Список строк может быть заранее приготовлен в файле, который загружается в ListBox. На этапе разработки приложения часто формируют, так называемый отладочный список, который формируют с помощью редактора текста (аналогично компоненту TMemo). Редактор текста открывается в свойстве Items.

Свойства TListBox

Items является указателем на класс TStrings, который имеет свойство Strings, являющееся массивом строк типа AnsiString. Свойство Count содержит количество строк. Запись вида ListBox1->Items->Count возвращает количество строк.

Класс TStrings содержиттметоды Add( ) – добавить строку в конец списка, Delete( ) – удалить строку и др. Например,

ListBox1->Items-> Add(“новая строка”);

ListBox1->Items->LoadFromFile(“spisok.txt”);

ItemIndex хранит выбранную в списке строку:
ListBox1->Items->Strings[ListBox1-> ItemIndex]

MultiSelect – определяет возможность выбора более одной строки за один раз. Когда это свойство имеет значение true, и выбрано множество строк, значение свойства ItemIndex, указывает на строку, имеющую фокус, т.е. на последнюю отмеченную.

Sorted – устанавливает сортировку по алфавиту.

Style – определяет вид вывода на экран названия элементов.

Методы TListBox

Clear( ) – удаляет список,

SetFocus( ) – делает список активным,

Hide( ) – скрывает список,

Show( ) – делает список видимым,

Sort( ) – сортирует элементы в порядке возрастания.

Если количество строк не помещается в отведенный размер, автоматически включается вертикальная полоса прокрутки, при условии, что длина строки меньше ширины прямоугольника. Для выполнения этого условия, можно ограничить длину строки:
int MaxWidth=500;
ListBox1->Perform(LB_SETHORIZONTALEXTENT, MaxWidth, 0)

Компонент TComboBox является комбинацией редактируемого поля и списка TListBox – редактируемое поле с треугольной кнопкой справа. Выбор данных из списка возможен двумя способами: щелчком мыши по нужной строке или вводом нужной строки в редактируемое поле.

Компонент TMaskEdit создает редактируемое текстовое поле (маску) для ввода данных специфического формата: даты, времени, номера телефона и т.п. При задании формата ввода данных проверяется их соответствии формату. Маска налагает ограничения на вводимые символы. Маска состоит из трех полей, разделенных точкой с запятой.

Первое поле маски может содержать перечень специальных символов:
! – его присутствие интерпретирует необязательные символы как лидирующие пробелы;
> – все последующие символы должны быть набраны в верхнем регистре до коца маски, либо до появления <;
< – аналогично предыдущему для нижнего регистра;
<> – возможно использование обоих регистров;
\ – символ ввода, следующий за этим – символьная константа;
L – ввод символов алфавита языка, установленного по умолчанию;
l – в позицию, которую он занимает, можно вводить только символы алфавита;
A – ввод алфавитно-цифровых символов;
a – совпадает с предыдущим, но не требует обязательного ввода;
C – требует обязательного ввода любого символа в позицию, которую он занимает;
0 – требует обязательного ввода цифры;
9 – требует необязательного ввода цифры;
– необязательный ввод цифры или знаков + и –;
: – используется при вводе времени
/ – используется при вводе даты;
; – деление частей маски;
_ – вставляет пробелы в текст.

Остальные символы могут использоваться в качестве литералов, вставляемых автоматически (курсор ввода перескакивает через них). Кроме того, после использования в маске символа \ все последующие символы воспринимаются как литералы.

Второе поле маски – один символ, указывающий, могут ли символы-литералы из маски включаться в данные. Например, маски телефонного номера: (000)_000-0000;0;*. Нуль во втором поле означает, что свойство Text включает 10 цифр, а остальные литералы

не включаются в результат ввода.

Третье поле маски содержит символ, который пользователь увидит до того, как будут введены данные. По умолчанию этот символ _.

Маска задается с помощью редактора маски, который открывается нажатием кнопки многоточие, расположенной в свойстве EditMask.

Задание

Создать приложение, содержащее возможность выбора в списке фамилий нескольких строк и копирование их (по нажатию кнопки) в другой список. Исходный список должен храниться в файле. Отсортировать новый список в алфавитном порядке. Для каждой фамилии из списка предусмотреть ввод персональных данных, согласно заданию варианта. Для ввода персональных данных использовать маски.

Лабораторная работа №12
Компоненты TIMage, TShape, TBevel

Через компонент TIMage в форму выводится графическое изображение, которое указывается в свойстве Picture. Компонент TIMage содержит свойства, определяющие формат вывода изображения внутри границ объекта, заданного в виде пустого квадрата. Некоторые из них:

Picture – указатель на класс TPicture, объектами которого являются значки, метафайлы, растровые изображения. Для загрузки изображения в компонент в поле этого свойства находится кнопка с многоточием, с помощью которой можно открыть диалоговое окно загрузки изображения. Методы TPicture: LoadFromFile( ) позволяет загрузку изображения: Image->Pictuche->SaveToFile(“имя файла”); SaveToFile( ) – сохраняет изображение в файле;

Stretch – имеет значение true, если изображение должно принимать размеры и форму компонента и менять их в соответствии с изменением параметров формы. Изменение размеров изображения происходит непропорционально, что приводит к искажению изображения. Во избежание этого требуется пользоваться свойством AutoSize. Если это свойство установлено в true, рамка компонента автоматически настраивается на размер изображения и ее нельзя изменить. Свойство Stretch при этом отключается.

Canvas – класс со свойствами и методами, позволяющими рисовать изображения. Оно доступно только для изображения формата bmp. Орудиями рисования являются Pen, Brush и методы.

Assign – используется при копировании изображения в буфер по правилу : куда->Assign(откуда); Для запоминания изображения в буфере в программный модуль должен быть включен файл Clipbrd.hpp.

Clipboard(0->Assign(Image1->Picture) //копирование изображения в буфер

Компонент TShape предназначен для рисования простых геометрических фигур, вид которых задается свойством Shape. Цвет и способ штриховки фигуры задаются в свойстве Bruch с помощью вложенных свойств Color и Style. Характеристики контура фигуры задаются во вложенных свойствах свойства Pen, а заливка свойством Bruch.

Компонент TBevel задает обрамление: рамку, кадр или линии (свойство Shape), которые могут быть вогнутыми или выпуклыми (свойство Style).

Задание

Создать программу, демонстрирующую свойства перечисленных компонент.

Лабораторная работа №13
Компоненты TPageControl, TScrollBar, TScrollBox

Компонент TPageControl позволяет построить набор страниц в рамках одной фор-

мы, которые перекрывают друг друга и которые можно перелистывать с помощью щелчка. Для создания страниц необходимо поместить компонент TPageControl в форму и открыть контекстное меню. Управлять формированием страниц возможно командами: New Page, Previos Page, Delete Page. Если страницы не умещаются в поле компонента, автоматически формируются полосы прокрутки. Изменение формы полей с названиями страниц происходит по свойству Style.

Каждая страница компонента TPageControl представляет собой объект – экземпляр класса TTabSheet с собственными свойствами и событиями, например: PageIndex указывает номер страницы в списке, нумерация страниц начинается с нуля. Значение поля изменяется, если страница удаляется или перемещается. Например, значение TabSheet1->PageIndex выводит номер страницы во множестве страниц компонента.

TabIndex показывает порядковый номер страницы среди видимых страниц.

Свойства компоненты TPageControl:

ActivePage указывает на активную (открытую) в данный момент страницу. Можно с помощью него установить активную страницу: PageControl1->ActivePage=TabSheet5;

RaggedRight, равное true, сжимает поля с названиями страниц и делает их компактными.

TabPosition определяет, в какой части компонента будут появляться вкладки: в верхней(Top), нижней(Botton), слева(Left) или справа(Right).

ScrollOpposite задает, как при многострочном режиме будут прокручиваться вкладки. При значение false все вкладки после их переключения остаются на той же стороне компонента, на которой они были. Если значение true, и при этом значение TabPosition – Left или Right, все вкладки которые находятся справа (слева) от переключаемой вкладки перемещаются на противоположную сторону компонента.

ActivePageIndex представляет индекс конкретной страницы. Множество страниц компонента определено в массиве Page[].

Pages – массив страниц. С его помощью можно добраться до любой страницы, задав ее номер, например для пятой страницы, PageControl1->Page[4].

PageCount содержит число страниц компонента TPageControl.

Из событий компонента компонента TPageControl выделяется OnChange – оно возникает, когда страница выбрана.

Компонент TScrollBar используется для прокручивания содержимого окон, форм, компонентов, у которых нет свойства прокрутки содежимого.

Свойства TScrollBar:

Kind указывает вертикальность (SbVertical) или горизонтальность (SbHorizontal) полосы прокрктки.

LargeChange задает, какое свойство примет Position, если пользователь щелкнкт мышью по полосе прокрутки вне движка или нажмет клавиши «PageUp» «PageDown»

Max и Min определяют диапазон, в котором изменяется свойство Position.

PageSize определяет размер движка полосы прокрутки.

Position показывает текущую позицию движка в полосе прокрутки. Если его задать программно, можно заставить перемещаться движок полосы прокрутки. Например, Position=Min заставит движок появиться с левого края горизонтальной прокрутки или – с верхнего – в вертикальной.

SmallChange определяет, насколько изменяется свойство Position, когда пользователь нажимает кнопки со стрелками на полосе прокрутки или клавиатуре.

Лабораторная работа №14
Компонент TServerSocket

Данный компонент делает приложение сервером, работающим по протоколу TCP/IP.

Свойства *TServerSocket*

Свойство Active - это свойство указывает, открыто ли так называемое *стыковочное соединение* и доступно ли оно для связи с другими машинами. Это свойство унаследовано от абстрактного класса TAbstractSocket, который содержит свойства и методы, позволяющие работать приложению со стыковочными компонентами (sockets) Windows, которые, в свою очередь, включают в себя набор коммуникационных протоколов, позволяющих приложению соединяться с другими машинами для чтения и записи информации. Стыковочные компоненты позволяют приложению формировать соединения с другими машинами, не заботясь о деталях существующего сетевого программного обеспечения. Свойства Компонента TAbstractSocket описывают IP-адрес стыковочного соединения и обслуживают его. IP-адрес - это строка из четырех чисел, разделенных между собой точками (например, 192.200.1.15).

Чтобы создать стыковочное соединение с другой машиной используется класс TClientSocket, а для создания стыковочного соединения, которое отвечает на запросы о соединении, поступающие от других машин используется класс TServerSocket. Перед тем как использовать или изменить стыковочное соединение, следует посмотреть его свойство Active, чтобы определить, открыто ли соединение и готово ли оно к работе. Для клиентских стыковочных соединений (компонент TClientSocket) установка свойства Active открывает или закрывает стыковочное соединение с другой машиной. Для серверных стыковочных соединений установка свойства Active открывает или закрывает прослушивание запросов от клиентов (прослушивание — это процесс, когда сервер ждет появления запросов от других машин и устанавливает соединение с той из них, от которой он принял запрос). В режиме проектирования приложения установка свойства Active в true заставляет стыковочное соединение сервера открыть соединение (т. е. будет установлено прослушивание запросов от других машин), когда приложение запустится.

Свойство Port - задание числового идентификатора, идентифицирующего серверное стыковочное соединение. Номера портов позволяют отдельной системе, заданной на стороне клиента свойствами Host или Address принимать одновременно множество соединений. (В свойстве Host задается URL, синоним IP-адреса, например, www. Mail.ru, в свойстве Address задается IP-адрес.) У свойства Port может быть много значений: это связано с обслуживанием данных, передаваемых по разным протоколам. Серверные стыковочные соединения устанавливают свойство Port в определенное значение, которое могут использовать клиенты для инициирования соединений. Для Клиентских стыковочных соединений Port — это идентификатор порта сервера, с которым клиент желает осуществить соединение. Значение свойства Port обычно связано с видом обслуживания, которое требуется клиенту от серверного приложения. Изменение значения свойства должно происходить при неактивном соединении, иначе возникает исключительная ситуация типа EsocketError.

Свойство ServerType указывает, каким будет каждое принятое сервером соединение: выделяется ли оно автоматически в отдельный исполняемый поток, или нет. Первое происходит, если ServerType установлено в stThreadBlocking. В этом случае исполнение потока не происходит, пока не будет прочитана или записана вся информация, проходящая через соединение. Для каждого соединения такой поток генерирует события OnClientRead или OnClientWrite. Если свойство serverType имеет значение stNonBiocking, то обработка всех процессов чтения и записи информации, проходящих через стыковочные соединения, происходит асинхронно. При этом режиме все клиентские соединения обрабатываются по умолчанию в одном исполнительном потоке.

Свойство Service задает имя службы, для которой используется стыковочное соединение. Служба – это подсистема сетевого программного обеспечения, выполняющая конкретную задачу. Для серверных стыковочных соединений использование свойства Service вместо свойства Port означает, что сервер будет прослушивать TCP/IP-запросы соответствующего ему порта.

Свойство Socket указывает на TServerwinSocket-объект, который описывает концевую точку прослушиваемого соединения. С помощью этого свойства можно получить (пользуясь свойствами и методами класса TServerWinSocket) следующие данные:

информацию о потоках соединения, которые сформированы для обработки серверным стыковочным соединением;

информацию о соединениях в виде Socket->connections[int Index] — массива открытых соединений с клиентскими стыковочны ми узлами для данного прослушивающего стыковочного узла;

другие данные.

Свойство ThreadCacheSize – задает максимальное число потоков, используемых для новых клиентских соединений.

Компонент *TClientSocket*

Этот компонент превращает приложение в клиента сетевого соединения по протоколу TCP/IP.

Свойства *TClientSocket*

Свойство Active аналогично одноименному свойству компонента TServerSocket.

Свойство Address задается IF-адрес серверноного проиложения. Для каждого стыковочного клиентского соединения надо придать свойству Address значение IP-адреса того сервера, с которым это стыковочное соединение должно быть установлено. Когда соединение открыто, значение Address связывается с соединением. Если стыковочное соединение содержит значение свойства Host вместо значения свойства Address, то IP-адрес извлекается из специальной таблицы. Одна серверная система может поддерживать более одного IP-адреса.

Свойство ClientType определяет, как осуществляет клиентский стыковочный узел чтение и запись информации: асинхронно или нет. Если ClientType = ctNonBlocking, клиентский стыковочный узел отвечает на асинхронные события записи и чтения.

Свойство Host – это URL, синоним IP-адреса серверного приложения, например. **www.rambler.com.** Если задан IP-адрес, это свойство указывать не надо.

Свойство Port – номер порта стыковочного узла сервера, с которым клиент собирается осуществить соединение.

Свойство Service аналогично одноименному свойству компонента TServerSocket.

Свойство Socket описывает клиентский стыковочный узел с помощью свойств и методов класса TClientWinSocket, указателем на который оно является. Например, свойство позволяет:

определить адрес и порт стыковочных соединений и клиента, и сервера, связанных с соединением;

читать и писать информацию через стыковочное соединение:

```
AnsiString   S=Socket->ReceiveText (),   Socket->SendText(S);
```

События *TClientSocket*

Событие onConnect происходит сразу после того, как установлено соединение со стыковочным узлом сервера. В зависимости от службы, заданной в соответствующем свойстве, в этот момент могли бы начаться чтение или запись через стыковочный узел клиента. Когда стыковочный узел открывает соединение, возникают следующие события:

onLookup возникает раньше попытки найти стыковочный узел сер вера;

OnConnecting возникает после того, как найден стыковочный узел сервера, требование соединения принято сервером и подготовлено клиентским стыковочным узлом;

OnConnect возникает после того, как соединение установлено.

Событие OnDisconnect происходит прямо перед тем, как клиентский стыковочный узел закрывает соединение с сервером. OnDisconnect происходит после проверки того, что свойство Active установлено в false, но до закрытия существующего соединения.

Событие OnError – это реакция на ошибки, возникающие при работе стыковочного соединения. При обработке ошибки, надо установить параметр

ErrorCode обработчика в 0, иначе после выхода из обработчика возникнет исключительная ситуация типа ESocketError.

Событие OnLookup возникает перед попыткой найти серверное стыковочное соединение, с которым собирается соединиться данный клиент.

Событие OnRead возникает при чтении информации из стыковочного соединения. Если стыковочное соединение – блокированное, то для чтения из него надо использовать объект TwinSocketstream. В противном случае надо использовать методы параметра Socket. (В стыковочных соединениях Nonblocking для последнего бита данных, передаваемых через это соединение, не всегда возникает событие OnRead. Поэтому в этих случаях надо проверять непрочитанные данные в обработчике события OnDisconnect, чтобы быть уверенным, что все обработано верно.)

Событие OnWrite возникает тогда, когда клиентское стыковочное соединение пишет информацию в стыковочное соединение сервера. Если стыковочное соединение блокировано, то для записи следует использовать методы объекта TwinSocketstream. В противном случае используются методы параметра Socket.

Задание

Используйте компоненты TServerSocket и TClientSocket в приложении, обеспечивающем обмен сообщениями между клиентом и сервером. Приложения после запуска должно выполнять прослушивание сервером клиентов. Режим запускается либо командой File|Listen меню приложения, либо дублирующей эту команду кнопкой со знаком "?". сказки).

В нижней части формы приложения расположить компонент TStatusBar, в поле которого отображаются производимые действия. После запуска прослушивания можно запускать соединение с сервером со стороны клиента (командой File|Connect меню или кнопкой с изображением конверта). В этом случае запрашивается URL сервера с помощью функции InputQuery( ). Для проверки работы компонентов на локальной машине, в ответ па запрос следует задать собственного компьютера, которое можно найти либо в реестре Windows, либо с помощью команды Сведения о системе.

Лабораторная работа №14
МЕТОДИКА РАСЧЕТА КОНФИГУРАЦИИ СЕТИ ETHERNET

Гарантию корректной работы сети дает соблюдение многочисленных ограничений, установленных для различных стандартов физического уровня сетей Ethernet.

Наиболее часто приходится проверять ограничения, связанные с длиной отдельного сегмента кабеля, а также количеством повторителей и общей длиной сети. Правила «5-4-3» для коаксиальных сетей и «4-х хабов» для сетей на основе витой пары и оптоволокна не только дают гарантии работоспособности сети, но и оставляют большой «запас прочности» сети. Например, если посчитать время двойного оборота в сети, состоящей из 4-х повторителей 10Base-5 и 5-ти сегментов максимальный длины 500 м, то окажется, что оно составляет 537 битовых интервала. А так как время передачи кадра минимальной длины, состоящего вместе с преамбулой 72 байт, равно 575 битовым интервалам, то видно, что разработчики стандарта Ethernet оставили 38 битовых интервала в качестве запаса для надежности. Тем не менее, комитет 802.3 говорит, что и 4 дополнительных битовых интервала создают достаточный запас надежности.

Комитет IEEE 802.3 приводит исходные данные о задержках, вносимых повторителями и различными средами передачи данных, для тех специалистов, которые хотят самостоятельно рассчитывать максимальное количество повторителей и максимальную общую длину сети, не довольствуясь теми значениями, которые приведены в правилах «5-4-3» и «4-х хабов». Особенно такие расчеты полезны для сетей, состоящих из смешанных кабельных систем, например коаксиального и волоконно-оптического, на которые прави-

ла о количестве повторителей не рассчитаны. При этом максимальная длина каждого отдельного физического сегмента должна строго соответствовать стандарту.

Чтобы сеть Ethernet, состоящая из сегментов различной физической природы работала корректно, необходимо выполнение четырех основных условий:

- количество станций в сети не более 1024;
- максимальная длина каждого физического сегмента не более величины, определенной в соответствующем стандарте физического уровня;
- время двойного оборота сигнала (Path Delay Value, PDV) между двумя самыми удаленными друг от друга станциями сети не более 575 битовых интервала;
- сокращение межкадрового интервала IPG (Path Variability Value, PVV) при прохождении последовательности кадров через все повторители должно быть не больше, чем 49 битовых интервала. Так как при отправке кадров конечные узлы обеспечивают начальное межкадровое расстояние в 96 битовых интервала, то после прохождения повторителя оно должно быть не меньше, чем 96 - 49 = 47 битовых интервала.

Соблюдение этих требований обеспечивает корректность работы сети даже в случаях, когда нарушаются простые правила конфигурирования, определяющие максимальное количество повторителей и общую длину сети в 2500 м.

Расчет PDV

Для упрощения расчетов обычно используются справочные данные IEEE, содержащие значения задержек распространения сигналов в повторителях, приемопередатчиках и различных физических средах. В таблице 6 приведены данные, необходимые для расчета значения PDV для всех физических стандартов сетей Ethernet. Битовый интервал обозначен как bt.

Тип сегмента	**База левого сегмента, bt**	**База промежуточного сегмента, bt**	**База правого сегмента, bt**	**Задержка среды на 1 м, bt**	**Максимальная длина сегмента, м**
10Base-5	11,8	46,5	169,5	0,0866	500
10Base-2	11,8	46,5	169,5	0,1026	185
10Base-T	15,3	42,0	165,0	0,113	100
10Base-FB	—	24,0	-	0,1	2000
lOBase-FL	12,3	33,5	156,5	0,1	2000
AUI (> 2 м)	0	0	0	0,1026	2+48

Таблица 1. Данные для расчета значения PDV

Комитет 802.3 максимально упростил выполнение расчетов. Данные, приведенные в таблице, включают сразу несколько этапов прохождения сигнала, такие как задержки, вносимые повторителем, состоят из задержки входного трансивера, задержки блока повторения и задержки выходного трансивера. В таблице все эти задержки представлены одной величиной, названной базой сегмента. Складывать задержки дважды, вносимые кабелем, не нужно – в таблице даются удвоенные величины задержек для каждого типа кабеля.

Использование табличных понятий левый сегмент, правый сегмент и промежуточный сегмент рассмотрено на примере сети, приведенной на рисунке 6.

Левым сегментом называется сегмент, в котором начинается путь сигнала от выхода передатчика конечного узла. На рисунке это сегмент 1. Затем сигнал проходит через промежуточные сегменты 2 – 5 и приходит до наиболее удаленного узла сегмента 6, который называется правым. Именно здесь в худшем случае происходит столкновение кадров и возникает коллизия, что и подразумевается в таблице.

С каждым сегментом связана постоянная задержка, названная базой, которая зависит от типа сегмента и от положения сегмента на пути сигнала (левый, промежуточный или правый). База правого сегмента, в котором возникает коллизия, намного превышает

базу левого и промежуточных сегментов.

С каждым сегментом также связана задержка распространения сигнала вдоль кабеля сегмента, зависящая от длины сегмента и вычисляемая путем умножения времени распространения сигнала по одному метру кабеля (в битовых интервалах) на длину кабеля в метрах. Расчет заключается в вычислении задержек, вносимых каждым отрезком кабеля (приведенная в таблице задержка сигнала на 1 м кабеля умножается на длину сегмента), а затем суммировании этих задержек с базами левого, промежуточных и правого сегментов. Общее значение PDV не должно превышать 575.

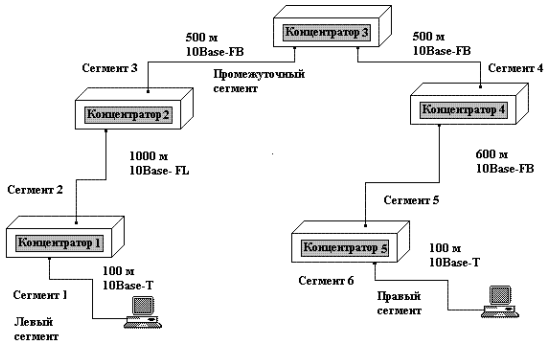


Рисунок 1**.** Пример сети Ethernet

Так как левый и правый сегменты имеют различные величины базовой задержки, то в случае различных типов сегментов на удаленных краях сети необходимо выполнить расчеты дважды: один раз принять в качестве левого сегмента одного типа, а во второй — сегмент другого типа. Результатом можно считать максимальное значение PDV. В примере крайние сегменты сети принадлежат к одному типу — стандарту 10Base-T, следовательно, двойной расчет не требуется. Если же сегменты были разного типа, то в первом случае нужно было бы принять в качестве левого сегмент между станцией и концентратором 1, а во втором считать левым сегмент между станцией и концентратором 5.

Приведенная на рисунке сеть в соответствии с правилом 4-х хабов не является корректной — в сети между узлами сегментов 1 и 6 имеется 5 хабов, хотя не все сегменты являются сегментами 10Base-FB. Кроме того, общая длина сети равна 2800 м, что нарушает правило 2500 м.

Расчет значения PDV для рассматриваемого примера:

Левый сегмент 1: 15,3 (база) + 100 x 0,113 = 26,6.
Промежуточный сегмент 2: 33,5 + 1000 x 0,1 = 133,5.
Промежуточный сегмент 3: 24 + 500 x 0,1 - 74,0.
Промежуточный сегмент 4: 24 + 500 x 0,1 - 74,0.
Промежуточный сегмент 5: 24 + 600 x 0,1 - 84,0.
Правый сегмент 6: 165 + 100 x 0,113 = 176,3.
Сумма всех составляющих дает значение PDV, равное 568,4.

Полученное значение PDV меньше максимально допустимой величины 575, и, следовательно, сеть проходит по критерию времени двойного оборота сигнала, несмотря на то, что ее общая длина составляет больше 2500 м, а количество повторителей — больше 4-х.

Расчет PVV

Чтобы признать конфигурацию сети корректной, нужно рассчитать также уменьшение межкадрового интервала повторителями, то есть величину PVV.

Для расчета PVV также можно воспользоваться значениями максимальных величин уменьшения межкадрового интервала при прохождении повторителей различных физических сред, рекомендованными IEEE и приведенными в таблице 2.

Тип сегмента	Передающий сегмент, bt	Промежуточный сегмент, bt
10Base-5 или 10Base-2	16	11
10Base-FB	—	2
10Base-FL	10,5	8
10Base-T	10,5	8

Таблица 2. Сокращение межкадрового интервала повторителями

В соответствии с этими данными рассчитаем значение PVV для примера.
Левый сегмент 110Base-T: сокращение в 10,5 bt.
Промежуточный сегмент 2 10Base-FL: 8.
Промежуточный сегмент 3 10Base-FB: 2.
Промежуточный сегмент 4 10Base-FB: 2.
Промежуточный сегмент 5 10Base-FB: 2.

Сумма этих величин дает значение PVV, равное 24,5, что меньше предельного значения в 49 битовых интервала. В результате приведенная в примере сеть соответствует стандартам Ethernet по всем параметрам, связанным и с длинами сегментов, и с количеством повторителей.

Если расчеты показывают, что сеть неработоспособна, то для преодоления этих ограничений предлагаются следующие методы:

1. Уменьшение длины кабелей с целью снижения задержки прохождения сигнала по сети (если возможно).
2. Уменьшение количества концентраторов для снижения задержек (если возможно).
3. Выбор кабеля с наименьшей задержкой. Кабели различных марок имеют разные задержки, то есть разные скорости распространения сигнала. Различия могут достигать 10%.
4. Разбиение сети на две части или более с помощью коммутатора – более радикальный метод. Коммутатор снижает требования к сети во столько раз, на сколько сегментов (*зон конфликта*) он разбивает сеть. Для каждой новой части сети требуется произвести расчет работоспособности еще раз. Сегмент, который присоединяет коммутатор, также входит в *зону конфликта*, и его надо учитывать при расчетах.

Выбор конфигурации Fast Ethernet

Для определения работоспособности сети Fast Ethernet стандарт IEEE 802.3 предлагает две модели, называемые Transmission System Model 1 и Transmission System Model 2. Первая модель основана на нескольких несложных правилах. Она исходит из того, что все компоненты сети (в частности, кабели) имеют наихудшие из возможных временные характеристики, поэтому всегда дает результат со значительным запасом. Вторая модель использует систему точных расчетов с реальными временными характеристиками кабелей. В связи с этим ее применение позволяет иногда преодолеть жесткие ограничения первой модели.

Правила модели 1

В соответствии с первой моделью, при выборе конфигурации надо руководствоваться следующими принципами:

Сегменты, выполненные на электрических кабелях (витых парах) не должны быть длиннее 100 метров. Это относится к кабелям всех категорий – 3, 4 и 5, к сегментам 100BASE-T4 и 100BASE-TX.

Сегменты, выполненные на оптоволоконных кабелях, не должны быть длиннее 412 метров.

Если используются адаптеры с внешними (выносными) трансиверами, то трансиверные кабели (MII) не должны быть длиннее 50 сантиметров.

Модель 1 выделяет три возможные конфигурации сети Fast Ethernet:

1. Соединение двух абонентов (узлов) сети напрямую, без репитера или концентратора (рисунок 2). Абонентами при этом могут выступать не только компьютеры, но и сетевой принтер, порт коммутатора, моста или маршрутизатора. Такое сопряжение называется соединением DTE - DTE или двухточечным.

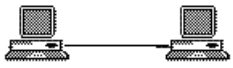


Рисунок 2. Двухточечное соединение компьютеров без концентратора

2. Соединение двух абонентов сети с помощью одного концентратора (репитера) класса I или класса II (рисунок 3).

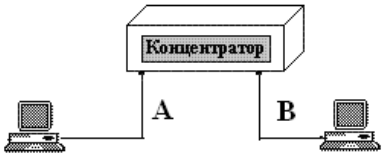


Рисунок 3. Соединение с одним концентратором

3. Соединение двух абонентов сети с помощью двух концентраторов класса II (рисунок 4). При этом предполагается, что для связи концентраторов всегда используется электрический кабель длиной не более 5 метров.

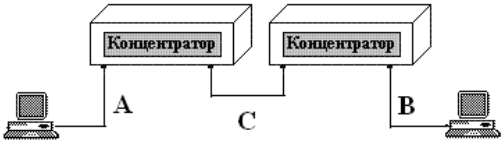


Рисунок 4. Соединение с двумя концентраторами

Концентраторы класса II имеют меньшую задержку, поэтому их может быть два. Использование трех концентраторов в соответствии с моделью 1 не допускается.

В случае выбора первой конфигурации (двухточечной) правила модели 1 предельно просты: электрический кабель не должен быть длиннее 100 метров, полудуплексный оптоволоконный – не более 412 метров, полнодуплексный оптоволоконный – 2000 метров (при этом задержка сигнала в кабеле не имеет значения, так как метод CSMA/CD не работает). В случае применения второй конфигурации (с одним концентратором) надо ограничивать длину кабелей A и B сети в соответствии с таблицей 3.

Вид кабеля A	**Вид кабеля B**	**Класс концентратора**	**Макс. длина кабеля A, м**	**Макс. длина кабеля B, м**	**Макс. размер сети, м**
TX, T4	TX, T4	I или II	100	100	200
TX	FX	I	100	160,8	260,8
T4	FX	I	100	131	231
FX	FX	I	136	136	272
TX	FX	II	100	208,8	308,8
T4	FX	II	100	204	304
FX	FX	II	160	160	320

Таблица 3. Максимальная длина кабелей в конфигурации с одним концентратором

В случае выбора третьей конфигурации сети (с двумя концентраторами) надо ограничивать длину кабелей A и B в соответствии с таблицей 4. При этом по умолчанию предполагается, что кабель C имеет длину 5 метров.

Вид кабеля A	**Вид кабеля B**	**Макс. длина кабеля A, м**	**Макс. длина кабеля B, м**	**Макс. размер сети, м**
TX, T4	TX, T4	100	100	205
TX	FX	100	116,2	221,2
T4	FX	136,3	136,3	241,3
FX	FX	114	114	233

Таблица 4. Максимальная длина кабелей в конфигурации с двумя концентраторами

В обеих конфигурациях с концентраторами при использовании одновременно электрического и оптоволоконного кабелей можно за счет уменьшения длины электрического кабеля увеличить длину оптоволоконного. Причем уменьшению длины электрического кабеля на 1 метр соответствует увеличение длины оптоволоконного кабеля на 1,19 метра. Например, уменьшив кабель TX на 10 метров, можно увеличить кабель FX на 11,9 метра, и его предельная длина составит при двух концентраторах 128,1 метра. Немного увеличится и предельный размер сети (в нашем примере на 1,9 метра).

В случае использования двух оптоволоконных кабелей можно уменьшать один из кабелей за счет увеличения другого. При уменьшении одного кабеля на 10 метров можно увеличить другой тоже на 10 метров. Если же используется два электрических кабеля, то увеличивать один из них за счет уменьшения другого нельзя, так как их длина в принципе не может превышать 100 метров из-за затухания сигнала в кабеле.

Концентратор класса II в принципе не может одновременно поддерживать сегменты с разными методами кодирования TX/FX и T4. Поэтому варианты, соответствующие вторым снизу строкам обеих таблиц 3 и 4 никогда не реализуются на практике, но стандарт все же дает цифры и для них.

Во всех перечисленных случаях под размером сети понимается размер *зоны конфликта* (области коллизии). При этом надо учитывать, что включение в сеть одного коммутатора позволяет увеличить полный размер сети вдвое.

Пример сети максимальной конфигурации в соответствии с первой моделью для витой пары показан на рисунке 5. Здесь максимальный размер *зоны конфликта* складывается из сегментов A, B и C, то есть составляет: 100 + 5 + 100 = 205 метров, что удовлетворяет условию работоспособности сети (таблица 4, верхняя строка).

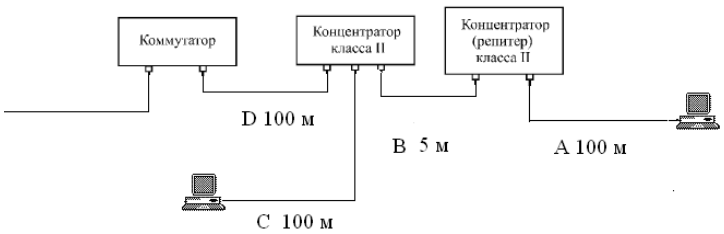


Рисунок 5. Пример максимальной конфигурации сети Fast Ethernet

Сегмент D также входит в *зону конфликта*, так как коммутатор является полноправным передатчиком пакетов сети. Длина сегмента D не может превышать 100 метров, чтобы суммарная длина сегментов A, B и D не была больше все тех же 205 метров. Сегменты, отделенные от рассматриваемой *зоны конфликта* коммутатором, никак не влияют на ее работоспособность.

Расчет по модели 2

Вторая модель для сети Fast Ethernet, как и в случае Ethernet, основана на вычислении суммарного двойного времени прохождения сигнала по сети. В отличие от второй модели, используемой для оценки конфигурации Ethernet, здесь не проводится расчетов величины сокращения межпакетного интервала. Это связано с тем, что даже максимальное количество репитеров и концентраторов, допустимых в Fast Ethernet (два), в принципе не может вызвать недопустимого сокращения межпакетного интервала.

Для расчетов в соответствии со второй моделью сначала надо выделить в сети путь с максимальным двойным временем прохождения и максимальным числом репитеров (концентраторов) между компьютерами, то есть путь максимальной длины. Если таких путей несколько, то расчет должен производиться для каждого из них.

Расчет в данном случае ведется на основании таблицы 5.

Тип сегмента	Задержка на метр	Макс. задержка
Два абонента TX/FX	-	100

Два абонента T4	-	138
Один абонент T4 и один TX/FX	-	127
Сегмент на кабеле категории 3	1,14	114 (100 м)
Сегмент на кабеле категории 4	1,14	114 (100 м)
Сегмент на кабеле категории 5	1,112	111,2 (100 м)
Экранированная витая пара	1,112	111,2 (100 м)
Оптоволоконный кабель	1,0	412 (412 м)
Репитер (концентратор) класса I	-	140
Репитер (концентратор) класса II с портами TX/FX	-	92
Репитер (концентратор) класса II с портами T4	-	67

Таблица 5. Двойные задержки компонентов сети Fast Ethernet

Для вычисления полного двойного (кругового) времени прохождения для сегмента сети необходимо умножить длину сегмента на величину задержки на метр, взятую из второго столбца таблицы. Если сегмент имеет максимальную длину, то можно сразу взять величину максимальной задержки для данного сегмента из третьего столбца таблицы.

Затем задержки сегментов, входящих в путь максимальной длины, надо просуммировать и прибавить к этой сумме величину задержки для приемопередающих узлов двух абонентов (три верхние строчки таблицы) и величины задержек для всех репитеров (концентраторов), входящих в данный путь (три нижние строки таблицы).

Суммарная задержка должна быть меньше, чем 512 битовых интервалов. При этом надо помнить, что стандарт IEEE 802.3u рекомендует оставлять запас в пределах 1 – 4 битовых интервалов для учета кабелей внутри соединительных шкафов и погрешностей измерения. Лучше сравнивать суммарную задержку с величиной 508 битовых интервалов, а не 512 битовых интервалов.

Все задержки, приведенные в таблице, даны для наихудшего случая. Если известны временные характеристики конкретных кабелей, концентраторов и адаптеров, то практически всегда предпочтительнее использовать именно их. В ряде случаев это может дать заметную прибавку к допустимому размеру сети.

Пример расчета по второй модели для сети, показанной на рисунке 5. Здесь существуют два максимальных пути: между компьютерами (сегменты A, B и C) и между верхним (по рисунку) компьютером и коммутатором (сегменты A, B и D). Оба эти пути включают в себя два 100-метровых сегмента и один 5-метровый. Предположим, что все сегменты представляют собой 100BASE-TX и выполнены на кабеле категории 5. Для двух 100-метровых сегментов (максимальной длины) из таблицы следует взять величину задержки 111,2 битовых интервалов.

Для 5-метрового сегмента при расчете задержки, умножается 1,112 (задержка на метр) на длину кабеля (5 метров): 1,112 * 5 = 5,56 битовых интервалов.

Величина задержки для двух абонентов TX из таблицы – 100 битовых интервалов.

Из таблицы величины задержек для двух репитеров класса II – по 92 битовых интервала.

Суммируются все перечисленные задержки: 111,2 + 111,2 + 5,56 + 100 + 92 + 92 = 511,96

это меньше 512, следовательно, данная сеть будет работоспособна, хотя и на пределе, что не рекомендуется.

Для гарантии лучше несколько уменьшить длину кабелей или взять кабели, имеющие меньшую задержку. Например, при использовании кабеля AT&T 1061 (NVP = 0,7, tз = 0,477) получаются следующие величины задержек для 100-метровых сегментов: (0,477 * 2) * 100 = 95,4 битовых интервалов (умножение на два необходимо, чтобы получить двойное время прохождения), а для 5-метрового сегмента – 4,77 битовых интервалов. Суммарная задержка при этом составит: 95,4 + 95,4 + 4,77 + 100 + 92 + 92 = 483,57,

то есть гораздо меньше 512 и даже 508, что означает полностью работоспособную сеть.

Пользуясь моделью 2, можно обойти некоторые ограничения модели 1, так как модель 1 строится из расчета на наихудший случай. Например, в сети может присутствовать больше двух концентраторов класса II или больше одного концентратора класса I, а кабель, соединяющий концентраторы, может быть длиннее 5 метров.

На рисунке 6 оказана сеть, содержащая три концентратора класса II, соединенных между собой отрезками кабеля длиной по 10 метров. Компьютеры соединены с концентраторами сегментами 100BASE-TX длиной по 50 метров.

Рисунок 6. Пример работоспособной конфигурации сети, нарушающей правила модели 1

Расчет двойного времени прохождения для этого случая.

1. Каждый из трех концентраторов класса II с портами TX даст задержку 92 битовых интервала. Суммарная задержка концентраторов составит 276 битовым интервалам.
2. Для двух соединительных кабелей между концентраторами задержка равна 2 * 1,112 * 10 = 22,24 битовых интервала.
3. Для двух сегментов TX по 50 метров задержка составит 2 * 1,112 * 50 = 111,2 битовых интервала.
4. Для двух абонентов TX задержка будет равна 100 битовым интервалам.
5. Итого суммарная задержка: 276 + 22,24 + 111,2 + 100 = 509,44 битовых интервала.

Данная сеть работоспособна. Но при этом надо учитывать, что каждый дополнительный концентратор класса II уменьшает общую допустимую длину кабеля на 92/1,112 = 82,7 метра. Сеть с четырьмя концентраторами не будет иметь смысла, так как на задержку в кабеле уже не остается почти никакого запаса (четыре концентратора дадут суммарную задержку в 92 * 4 = 368 битовых интервалов).

Какова будет максимальная величина сети Fast Ethernet? Для этого надо взять сеть с одним концентратором класса II и два сегмента 100BASE-FX. Элементарный расчет показывает, что при одинаковых сегментах длина каждого из них может достигать 160 метров (рисунок 7), а общая длина сети составит 320 метров.

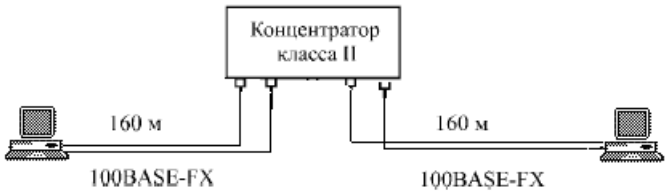


Рисунок 7. Сеть Fast Ethernet максимальной длины

Расчет двойного времени прохождения для этого случая будет выглядеть так:

$$92 + 100 + 2 * 1,0 * 160 = 512$$

Получается, что сеть работоспособна, хотя и на пределе. В данном случае важна только суммарная длина обоих кабелей. При уменьшении длины какого-нибудь из сегментов можно без потери работоспособности увеличить на точно такую же величину длину другого сегмента.

Если в приведенной конфигурации используется концентратор класса I, а не концентратор класса II, то допустимая суммарная длина сегментов сокращается с 320 метров

до 272 метров (расчет для этого случая очевиден). А согласно стандарту запаса лучше уменьшить суммарную длину кабеля на 1 – 4 метра, что даст снижение круговой задержки на 1 – 4 битовых интервала.

В заключение следует отметить, что модель 2 целесообразно применять в основном при наличии в сети оптоволоконных сегментов. На электрическом кабеле даже при большом желании довольно трудно создать сеть значительного размера.

Задание

На рисунке в вариантах задания представлен план расположения компьютеров в зданиях. Требуется предложить проект ЛВС по следующему плану:

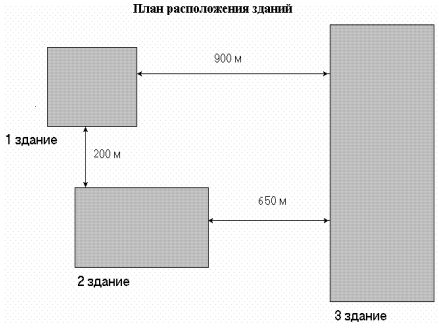
1. Выбрать топологию ЛВС. Обосновать выбор.
2. Выбрать оптимальную конфигурацию сети Ethernet.
3. Нарисовать функциональную схему ЛВС и составить перечень аппаратных средств.
4. Произвести ориентировочную трассировку кабельной системы и выполнить расчет длины кабельного соединения для выбранной топологии с учетом переходов между этажами.
5. С целью выполнения ограничений на длину кабеля и количества станций в сегменте выбранной конфигурации установить необходимые коммуникационные устройства.
6. Оценить проектируемую сеть по критерию времени двойного оборота (рассчитать PDV) и характеристики уменьшения межкадрового интервала повторителями (PVV).

Вариант 1

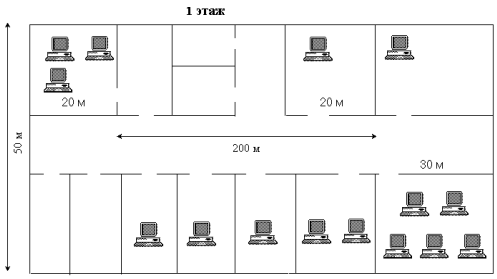
Первое из зданий трехэтажное. Компьютеры расположены в отделах на первом и третьем этажах. Общее число компьютеров – 12 (10 на первом этаже и 2 на втором).

Второе и третье здания одноэтажные. В каждом из них – 15 компьютеров.

План расположения зданий

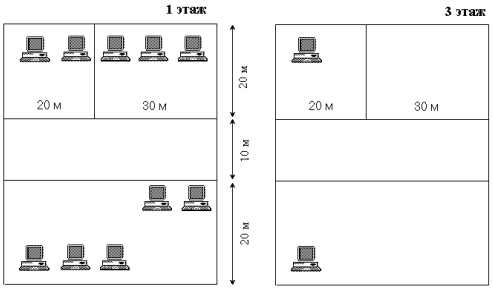


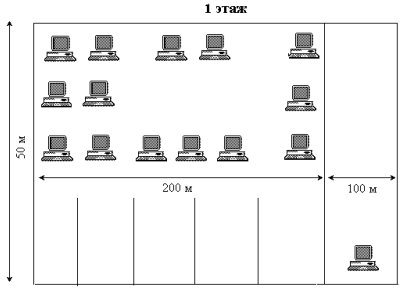
План третьего здания



План первого здания

План второго здания





МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО САМОСТОЯТЕЛЬНОЙ РАБОТЕ СТУДЕНТОВ

Самостоятельная работа – планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа является важным видом учебной и научной деятельности студента и играет значительную роль в рейтинговой технологии обучения. Государственным стандартом предусматривается, как правило, 50% часов из общей трудоемкости дисциплины на самостоятельную работу студентов. Количество часов самостоятельной работы и ее распределение между дидактическими единицами приведено в рабочей программе.

Задачами самостоятельной работы являются:

систематизация и закрепление полученных теоретических знаний и практических умений студентов;

углубление и расширение теоретических знаний;

формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;

развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;

формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;

развитие исследовательских умений;

использование собранного и полученного в ходе самостоятельных занятий для эффективной подготовки к итоговым зачетам и экзаменам.

В процессе самостоятельной работы студент приобретает навыки самоорганизации, самоконтроля, самоуправления, становится активным самостоятельным субъектом учебной деятельности.

Выполняя самостоятельную работу под контролем преподавателя, студент должен:

– освоить минимум содержания, выносимый на самостоятельную работу студентов и предложенный преподавателем в соответствии с Государственными образовательными стандартами высшего профессионального образования по данной дисциплине.

– планировать самостоятельную работу в соответствии с графиком самостоятельной работы, предложенным преподавателем.

– самостоятельную работу студент должен осуществлять в организационных формах, предусмотренных учебным планом и рабочей программой преподавателя.

– выполнять самостоятельную работу и отчитываться по ее результатам в соответствии с графиком представления результатов, видами и сроками отчетности по самостоятельной работе студентов.

Основной формой самостоятельной работы студента является изучение конспекта лекций, их дополнение, рекомендованной литературы, подготовка отчетов к лабораторным работам. Но для успешной учебной деятельности, ее интенсификации, необходимо учитывать следующие субъективные факторы:

Начинать самостоятельные внеаудиторные занятия следует с первых же дней семестра, пропущенные дни будут потеряны безвозвратно, компенсировать их позднее усиленными занятиями без снижения качества работы и ее производительности невозможно. Первые дни семестра очень важны для того, чтобы включиться в работу, установить определенный порядок, равномерный ритм на весь семестр.

КОНТРОЛЬ ЗНАНИЙ

Текущий контроль знаний предполагает защиту лабораторных работ и выполнение тестовых заданий. В тестовые задания входят вопросы по лекционному материалу и вопросы самостоятельной работы, приведенные в рабочей программе.

Вариант теста текущего контроля
Вариант 1

1. Какие функции выполняются на транспортном уровне эталонной модели взаимодействия открытых систем (OSI)?

а) определение начала и окончания сеанса связи; определение времени, длительности и режима сеанса связи; определение точек синхронизации для промежуточного контроля и восстановления при передаче данных;

б) контроль последовательности передачи данных; установка соответствия между транспортными (логическими) и сетевыми адресами абонентов; обнаружение и обработка ошибок передачи данных;

в) определение маршрутизации в сети и связь между сетями; обеспечение независимости высших уровней от используемой для передачи информации физической среды; обнаружение и обработка ошибок передачи данных;

г) определение метода доступа к среде передачи данных; определение логической топологии сети передачи данных; определение физической адресации.

2. ***Какое устройство называется повторителем?*** **а)** организовать обмен данными между сетевыми объектами, использующими различные протоколы обмена данными;

б) расширить сеть подключением дополнительных сегментов кабеля;

в) объединить несколько сегментов, так что передача данных между станциями внутри одного сегмента не будет влиять на передачу данных в других сегментах;

г) соединять сети разного типа, использующие одну сетевую операционную систему или протокол обмена данными.

3. Какую из ситуаций называют коллизией?

а) когда станция, желающая передать пакет, обнаруживает, что в данный момент другая станция уже заняла передающую среду;

б) когда в данных, переданных станцией, обнаружена ошибка;

в) когда две рабочие станции одновременно передают данные в разделяемую передающую среду;

г) когда передача данных завершена станцией успешно.

4. Какое название носит сети метод доступа станций к передающей среде в сети Ethernet?

а) маркерное кольцо

б) множественный доступ с контролем несущей и обнаружением коллизий

в) маркерная шина

г) раннее освобождение маркера

5. При каком способе обмена информацией между узлами сети данные передаются в одном направлении?

а) при симплексной передаче; **б)** при дуплексной передаче;

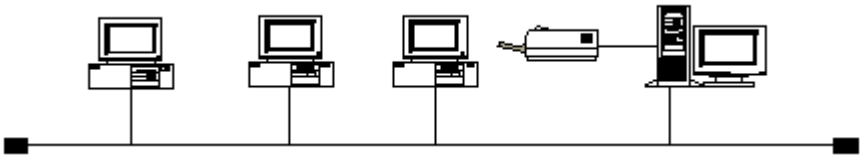
в) при полудуплексной передаче; **г)** при полусимплексной передаче

6. Как называется характеристика сети, предполагающая скрытие особенностей работы сети от конечного пользователя?

а) интегрируемость; **б)** прозрачность;

в) надежность; **с)** масштабируемость.

7. ***По какой топологии изображено подключение устройств на рисунке?***



а) звезда; **б)** шина; **в)** ячеистая топология; **г)** кольцо.

***8. Какой из перечисленных вариантов характеристик соответствует технологии* 10BASE-F?**

	а	б	в	г
Длина сегмента кабеля, м	500	185	100	2000
Тип кабеля	50 Омный коаксиальный, «толстый»	50 Омный коаксиальный, «тонкий»	Витая пара (UTP)	Оптоволокно
Топология подключения устройства	Шина	Шина	Звезда	Звезда

9. Какое из утверждений о логическим адресе узла сети верное?

а) не определяется используемым протоколом обмена данными;

б) определяется стандартом локальной сети;

в) не может быть изменен после подключения устройств к сети;

г) может быть изменен в процессе работы.

10. Поставьте соответствия между типом коммуникационного устройства и уровнями моделиOSI, на котором оно может работать

1) повторитель	**а)** физический, канальный, сетевой, транспортный
2) мост	**б)** физический, канальный
3) шлюз	**в)** физический
4) маршрутизатор	**г)** над уровнями модели OSI

11. Какое устройство изолирует трафик одной подсети от трафика другой подсети, но не может быть использован в сети с замкнутым контуром?

а) мост **б)** шлюз **в)** маршрутизатор **г)** повторитель

12. ***Что используются в одномодовом кабеле в качестве источника излучения света?*** _____

13. Какие из перечисленных пар сетевых технологий совместимы по форматам кадров и позволяют обрабатывать составную сеть без необходимости транслирования кадров?

а) Token Ring – Fast Ethernet

б) FDDI – Ethernet

в) Ethernet – Fast Ethernet

г) Token Ring - FDDI

14. ***Поставьте соответствия между названиями стандартов и технологий, которые они описывают.***

а) 802.1	**1)** технология Token Ring
б) 802.3	**2)** технология Ethernet
в) 802.5	**3)** общие определения ЛВС, связь с моделью OSI

15. ***Где располагается горизонтальная подсистема иерархической структурированной кабельной системы?***

а) в пределах этажа

б) в пределах здания

в) в пределах одной территории с несколькими зданиями

г) в пределах предприятия

16. Какими характеристиками оценивается степень искажения синусоидальных сигналов линиями связи?

а) затухание, мощность

б) пропускной способностью, полоса пропускания

в) помехоустойчивость, достоверность

г) амплитудно-частотная характеристика, полоса пропускания

17. Какие уровни модели OSI являются сетенезависимыми?

а) прикладной, представления данных

б) канальный, сетевой

в) транспортный, сеансовый

г) физический, сеансовый

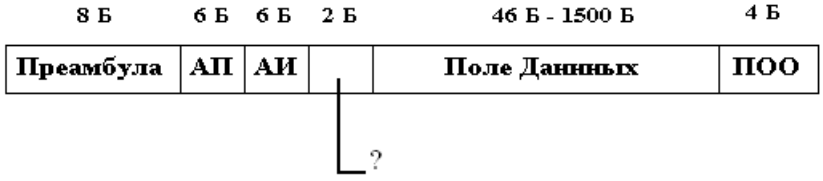
18. В каком году был принят первый стандарт сети Ethernet?

а) 1990 **б)** 1980 **в)**1970 **г)** 2000

19. Любая вычислительная система, построенная в соответствии с общедоступными спецификациями, соответствующими стандартам и принятые в результате публичного обсуждения, называется_____системой.

20. Стек коммуникационных протоколов _____ является лидером среди протоколов вычислительных сетей и использует гибкую систему адресацию, позволяющую более просто включать в интерсеть сети различных технологий.

21. На рисунке схематично изображен формат пакета стандарта IEEE 802.3. Каково название поля, отмеченного вопросительным знаком?

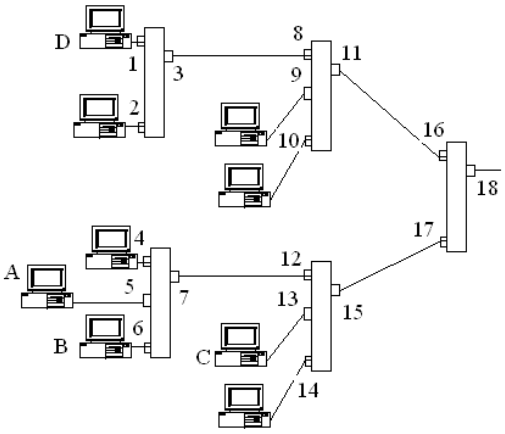


поле_____

22. Какое из перечисленных названий стандартов является общим термином для обозначения стандартов, использующих в качестве среды передачи данных витую пару?

а)100BASE-T **б)**100BASE-TX **в)** 100BASE-T4 **г)**100BASE-T2

23. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика составляет 0,01 мВт, а мощность шума в канале 0,0001 мВт



24. Если все коммуникационные устройства на схеме сети являются концентраторами, то на каких портах появится кадр, если его отправить с компьютера А компьютеру D?

а) на всех портах **б)** 5,6, 7 **в)** 4,5,6,7

г) 4,5,6,7,12, 13,14,15, 17,18

25. Если в предыдущем примере считать, что все устройства являются коммутаторами, то на каких портах появится кадр, если его отправить с компьютера А компьютеру С?

_____

ИТОГОВЫЙ КОНТРОЛЬ ЗНАНИЙ – Примеры билетов

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры	Кафедра ИУС
« » 20 г.	Факультет МиИ
Заведующий кафедрой ИУС	Курс 3
Утверждаю А.В. Бушманов	Дисциплина Сети ЭВМ и телекоммуникации

Экзаменационный билет № 1

1. Структура стандартов IEEE 802.x
2. Алгоритмы работы мостов
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика 0.063 мВт, а мощность шума в канале 0.001мВт

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры	Кафедра ИУС
« » 20 г.	Факультет МиИ
Заведующий кафедрой ИУС	Курс 3
Утверждаю А.В. Бушманов	Дисциплина Сети ЭВМ и теле коммуникации

Экзаменационный билет № 2

1. Эталонная модель взаимосвязи открытых систем
2. Амплитудно-частотная характеристика, полоса пропускания, затухание
3. Запишите значения следующих характеристик топологии 10BASE2: скорость передачи, длина сегмента кабеля, максимальное число станций в сегменте, тип кабеля.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры	Кафедра ИУС
« » 20 г.	Факультет МиИ
Заведующий кафедрой ИУС	Курс 3
Утверждаю А.В. Бушманов	Дисциплина Сети ЭВМ и теле коммуникации

Экзаменационный билет № 3

1. Классификация сетей по масштабу и территориальному признаку
2. Алгоритмы работы мостов
3. Определите пропускную способность канала связи для каждого направления дуплексного режима передачи, если известно, что полоса пропускания равна 600 кГц, а в методе кодирования используется 16 состояний сигнала.

Интерактивные технологии и инновационные методы, используемые в образовательном процессе

К ним относится электронное тестирование знаний, проводимое в качестве текущего контроля знаний, компьютерное моделирование и анализ полученных результатов.

Содержание