МИНИСТЕРСТВО НАУКИ, ВЫСШЕГО ОБРАЗОВАНИЯ И ИННОВАЦИЙ КР
КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. И.АРАБАЕВА
ОСПО ИНСТИТУТА НОВЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

«УТВЕРЖДАЮ»
Директор ИНИТ
КГУ им. И. Арабаева
к.т.н., и.о. доц. У.Т. Керимов
_____
«09» 09 2025г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

по дисциплине: **«Программное обеспечение компьютерных сетей»**
для студентов специальности: 230109 «Программное обеспечение вычислительной техники и автоматизированных систем

форма обучения: очное

Учебно-методический комплекс составлен на основе Государственного Образовательного Стандарта среднего профессионального образования КР

Учебно-методический комплекс разработала: магистр-преподаватель отделения СПО ИНИТ КГУ имени И. Арабаева Нурлан кызы Айжамал

Соавторы: Төлөнбаев Б.А.

Бишкек 2025г.

ОГЛАВЛЕНИЕ

МИНИСТЕРСТВО НАУКИ, ВЫСШЕГО ОБРАЗОВАНИЯ И ИННОВАЦИЙ КР
КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. И.АРАБАЕВА
ОСПО ИНСТИТУТА НОВЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

«УТВЕРЖДАЮ»
Директор ИНИТ
КГУ им. И. Арабаева
к.т.н., и. о. доц. У.Т. Керибов
«___» _____ 2025г.

РАБОЧАЯ ПРОГРАММА

по дисциплине: **«Программное обеспечение компьютерных сетей»**
для студентов специальности: 230109 «Программное обеспечение вычислительной техники и автоматизированных систем»
форма обучения: очное
институт: ИНИТ
отделение: ОСПО ИНИТ
курс: 3
семестр: 5/6
аттестация (семестр): 5
экзамен (семестр): 6
всего часов по учебному плану: 120
из них:
-лекции: 44
-практические: 28
-самостоятельная работа: 48

Рабочая программа составлена в соответствии с требованиями Государственного Образовательного Стандарта среднего профессионального образования КР
Рабочую программу разработала: магистр-преподаватель отделения СПО ИНИТ КГУ имени И. Арабаева Нурлан кызы Айжамал
Соавторы: Төлөнбаев Б.А.

Рассмотрена и утверждена на заседании
ОСПО ИНИТ КГУ им. И. Арабаева
Протокол № 1
от «02» 09 2025г.
Зав. отделением: Н.С. Сейтказиева

Одобрено учебно-методическим советом
ИНИТ КГУ им. И. Арабаева
Протокол № 1
от «04» 09 2025г.
Председатель УМС ИНИТ:

Бишкек 2025г.

3
Нурлан кызы А

1. Цели и задачи изучения дисциплины, ее значение в учебном процессе

1.1. Цели дисциплины

Целями освоения дисциплины «Программное обеспечение компьютерных сетей» являются:

- Дать студентам базовые знания по программным обеспечениям компьютерной сети
- Дать навыки практического выполнения типовых операций в широком спектре относящихся к программному обеспечению компьютерных сетей.

1.2. Задачи изучения дисциплины

Обучающие:

1. Расширять знания, полученные на уроках информатики, способствовать их систематизации;
2. Знакомить с основами знаний в области программного обеспечения и веб-серверов;
3. Развитие интереса к разработке сайтов

Развивающие:

1. Подготовить сознание студентов к системно-информационному восприятию мира, развивать стремление к самообразованию, обеспечить в дальнейшем социальную адаптацию в информационном обществе и успешную профессиональную и личную самореализацию;
2. Раскрыть креативные способности, подготовить к художественно-эстетическому восприятию окружающего мира;
3. Развивать композиционное мышление, художественный вкус, графическое умение;
4. Развивать творческое воображение;
5. Развивать эмоциональную сферу, чувства, душу.
6. Развивать моторику руки, зрительную память, глазомер.

Воспитательные:

1. Формировать информационную и эстетическую культуру обучающихся;
2. Воспитывать толерантное отношение в группе.;
3. Воспитывать собранность, аккуратность при подготовке к занятию;
4. Воспитывать умение планировать свою работу;
5. Воспитывать умственные и волевые усилия, концентрацию внимания, логичность и развитого воображения;

В результате освоения дисциплины студент должен:

Знать

- Основы верстки веб-страниц
- типовой состав прикладных программ для функционирования Web- сервера;
- типы Web-серверов;
- основные принципы защиты информации, методы и средства защиты информации;
- языки программирования, используемые для разработки технической части Web-проектов;
- основные системы управления контентом Web-проектов.

Уметь

- осуществлять техническое сопровождение Web-серверов;
  участвовать в выполнении мероприятий по обслуживанию Web-серверов и компьютерных сетей;
- самостоятельно осуществлять конфигурирование Web-сервера;
- осуществлять сопровождение различных систем управления сайтами;
- разрабатывать Web-сайты на различных платформах с учетом целевой аудитории пользователей сайта.

Владеть:

- верстать веб-страницы
- разрабатывать веб-сайты

2. Компетенции по Госстандарту.

Выпускник в соответствии с целями основной профессиональной образовательной программы и задачами профессиональной деятельности, указанными в пунктах 11 и 15 настоящего Государственного образовательного стандарта, должен обладать следующими компетенциями:

а) общими (ОК):

ОК-1.	Уметь организовывать собственную деятельность, выбирать методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
ОК-2.	Решать проблемы, принимать решения в стандартных и нестандартных ситуациях, проявлять инициативу и ответственность.
ОК-3	Осуществлять поиск, интерпретацию и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.
ОК-4	Использовать информационно – коммуникационные технологии в профессиональной деятельности.
ОК-5	Уметь работать в команде, эффективно общаться с коллегами, руководством, клиентами.
ОК-6	Брать ответственность за работу членов команды (подчиненных), за результат выполнения заданий.
ОК-7	Управлять собственным личностным и профессиональным развитием, адаптироваться к изменениям условий труда и технологий в профессиональной
ОК-8	Быть готовым к организационно – управленческой работе с малыми коллективами.
ОК-9	Способен приобретать новые знания, с большой степенью самостоятельности, с использованием современных образовательных и информационных технологий.
ОК-10	Способен на научной основе оценить свой труд, оценивать с большой степенью самостоятельности, результаты своей деятельности.

230109 – «Программное обеспечение вычислительной техники и автоматизированных систем»

б) профессиональными, соответствующими основным видам профессиональной деятельности (ПК):

производственно-технологическая деятельность:

ПК-1	Владеет знаниями об архитектуре и технических характеристиках персональных компьютеров;
ПК-8	Способен осуществлять модификацию, адаптацию и настройку программных продуктов;
ПК-11	Владеет знаниями о правилах и нормах охраны труда, техники безопасности, промышленной санитарии и противопожарной защиты.
ПК-13	Способен реализовать функции сопровождения программных продуктов;
ПК-16	Способен обеспечивать эффективное применение пакетов прикладных программ;

3. Межпредметные связи. Перечень дисциплин и их разделов, усвоение которых необходимо при изучении данной дисциплины.

Пререквизиты:

№	Названия дисциплин
1	Компьютерные сети
2	ОС и среды
3	Технические средства информатизации

Постреквизиты:

№	Названия дисциплин
1	Программное обеспечение компьютерных сетей
2	Веб-дизайн

Структура и трудоемкость дисциплины

Вид работы, семестр	Трудоемкость, час	
	очное обучение	заочное обучение
№ семестров	5,6	4
кредит		
Общая трудоемкость	120	60
Аудиторная работа	72	12
Лекции	44	8
Практические занятия/семинары	28	4
Лабораторные работы		
Самостоятельная работа	48	48
Курсовые работы или проекты *(при наличии)*		
Рефераты *(при наличии)*		
Внеаудиторные самостоятельные работы *(расчетно-графические задания, типовые расчеты, и т.д.)*		
Самоподготовка *(самостоятельное изучение теоретического материала, подготовка к практическим занятиям, текущему контролю и т.д.)*		
Виды текущего контроля *(перечислить)*		
Вид итогового контроля	экзамен – 6 сем.	экз

4. Структура дисциплины с разбивкой по видам занятий, часам и модулям

№	Тематика лекционных занятий	Лекции	Примечание
5-семестр			
1	Развитие сети Интернет. Стандартизация в интернет Понятие прокси-сервер	2	
2	Протоколы Интернет прикладного уровня Стек протоколов TCP/IP	2	
3	DNS – система доменных имен	2	
4	World Wide Web (WWW) Клиент-серверные технологии Web. Протокол HTTP.	2	
5	Обеспечение безопасности передачи данных HTTP Cookie	2	
Итого по модулю №1		10	
1	Сценарии и приложения выполняющиеся на стороне клиента Программы, выполняющиеся на клиент-машине	2	
2	Программы, выполняющиеся на сервере Насыщенные интернет-приложения.	2	
3	Введение в Jscript Краткая характеристика VBScript Java-апплеты	2	
4	ActionScript – общая характеристика. XAML и Microsoft Silverlight	2	
5	Понятие о DOM. DHTML.	2	
6	Серверные web-приложения. Стандарт CGI. Сценарии Python. Ruby ASP. ISAPI	2	
Итого по модулю №2		**12**	
Итого за 5-семестр		**22**	
6-семестр			
1	Введение в Jscript Краткая характеристика VBScript Java-апплеты	2	
2	Клиентские сценарии и приложения	2	
3	BOM и DOM- объектные модели браузера	2	
4	Языки разработки сценариев Perl и PHP Язык Perl Язык PHP	2	
5	SPA, MPA и PWA	2	
6	WEB-технологии. Web-приложение: понятия, компоненты и принципы работы	2	
7	Основные принципы функционирования интернет	2	
8	Основы разработки серверной части Web-приложения	2	
9	Архитектура web-приложений ASP.NET. Разработка web-приложений на платформе .NET.	2	

	Серверные элементы управления ASP.NET		
10	Синдикация и агрегирование web-контента Web-порталы. Классификация web-порталов.	2	
11	Приложения для социальных сетей Флоксономия Семантическая web-сеть Онтология Семантические web-сервисы	2	
Итого по модулю №2		**22**	
Итого за 6 семестр		**22**	

№	Тематика практических занятий	Кол-во часов	Примечание
1	Структура HTML-документа	2	
2	Работа с отступами и шрифтами	2	
3	Работа со списками Работа со ссылками	2	
5	Графика и мультимедиа	2	
6	Работа с каскадными таблицами стилей	2	
7	Экспорт стилей и валидация	2	
8	Знакомство с синтаксисом языка JavaScript	2	
Итого 5 семестр		**14**	
9	Знакомство с основными операторами. Условный оператор	2	
11	Разработка калькулятора	2	
12	Обработка данных формы PHP	2	
13	Создание счетчика посещений	2	
14	Вычисление значения функции	2	
15	Использование массивов	2	
16	Разработка базы данных	2	
Итого 6 семестр		**14**	
ВСЕГО		**28**	

№	Тематика лекционных занятий	Кол-во часов	Примечание
4-семестр			
1	Развитие сети Интернет. Стандартизация в интернет Понятие прокси-сервер	2	
2	Протоколы Интернет прикладного уровня Стек протоколов TCP/IP	2	
3	DNS – система доменных имен	2	
4	World Wide Web (WWW) Клиент-серверные технологии Web. Протокол HTTP.	2	
Итого за 4-семестр		**8**	
№	**Тематика практических занятий**	**Кол-во часов**	**Примечание**
1	Структура HTML-документа	2	
2	Работа с отступами и шрифтами	2	
Итого за 4-семестр		4	

5. Темы заданий СРС по дисциплине

Формы обучения кол-во часов	Задания для СРС
СРС (2 ч.)	Структура Интернет. Принципы построения. Перспективы развития.
СРС (2 ч.)	Подготовка доклада по теме: Дискретный канал связи.
СРС (2ч.)	Подготовка презентации: Помехоустойчивое кодирование
СРС (2ч.)	Подготовка рефератов по теме: Пакеты. Назначение и структура. Адресация пакетов.
СРС (2ч.)	Подготовить сообщения по теме: Коммутация каналов и коммутация пакетов. Достоинства и недостатки. Постоянная и динамическая коммутация.
СРС (2ч.)	Записывать ключевые слова
СРС (2ч.)	Стандартные сети
СРС (4ч.)	Подготовка доклада по теме: Структура электронного письма.
СРС (2ч.)	Оформление мультимедийной презентации по теме: Методика проектирования сети.
СРС (2ч.)	Записывать основные термины
СРС (2ч.)	Подготовка доклада по теме: Общие сведения о физической организации Интернет.
СРС (2ч.)	Способы программной защиты персонального компьютера.
СРС (2ч.)	Подготовить сообщения по теме: Скоростные и беспроводные сети
СРС (2ч.)	Сетевое программное обеспечение в компьютерных сетях
СРС (2ч.)	HTML- редакторы; Браузеры; Антивирусные сетевые программы
СРС (2ч.)	Подготовка к лекционным и практическим занятиям
СРС (2ч.)	Получить справочную информацию по интересующему Вас вопросу.
СРС (2ч.)	Применение сетевых утилит
СРС (2ч.)	Подготовить сообщения по теме: Стандартные и беспроводные сети
СРС (2ч.)	Брандмауер Windows
СРС (2ч.)	Основы сетей. Домен и хостинг
СРС (2ч.)	Семантические web-сервисы
48ч	**ИТОГО**

6. Распределение баллов по модулям и видам учебных занятий

№	Этапы проверки	Вид средства проверки	Баллы
1	1 модуль	Проверка практических заданий. Устный, тестирование. Посещение занятий.	100
2	2 модуль	Проверка практических заданий. Тестирование. Посещение занятий.	100
3	Итоговый контроль: • Практическое занятие; • СРС.	Контрольные и графические работы, рефераты, презентации, СРС, практические задания. Тестирование. Посещение занятий.	100
	Итого средний балл		**100**

Итоговое распределение баллов по модулям

		Удовлетворительно	Хорошо	Отлично
Модуль 1 – 100 б.		60-79	80-89	90-100
Модуль 2 – 100 б.		60-79	80-89	90-100
Практическое занятие – 50 б.	Итоговый контроль	60-79	80-89	90-100

7. Список основной и дополнительной литературы

№	Библиографическое описание издания (автор, наименование, вид, место и год издания, кол.стр.)	Виды занятия в котором используется	Кол-во экз. в библиотеке университета	примечание
Основная литература				
1	П.Б. Храмцов, С.А.Брик, А.М. Русак, А.И.Сурин Основы WEB-технологий. – М.: ИТУИТ.РУ, 2003	Лекция	1	
2	В.В. Дунаев JavaScript.— СПб. : “Питер”, 2003	Лекция	2	
3	Пятибратов, А.П. Вычислительные системы, сети и телекоммуникации: учебник для студ. вузов / А. П. Пятибратов, Л. П. Гудыно, А. А. Кириченко ; ред. А. П. Пятибратов. - 4-е изд., перераб. и доп. - М. : ИНФРА-М, 2014.	Практика	2	
Дополнительная литература				
1	Проектирование web-приложений и программных систем в Open Soure	Практика		

	учебное пособие / Г. А. Лисьев, В. Г. Измайлов, М. Ю. Озерова, А. Л. Трейбач. - М.: Флинта, 2011		1	

7.1. Интернет-ресурсы

1. Основы сетей и передачи (http://intuit.ru/).
2. Основы интернет (http://psbatishev.narod.ru/).

8. Вопросы (тесты) к модулям

№	1 полугодие	Перечень вопросов (тестов)
1	Модуль 1	Let's Start!
2	Модуль 2	Let's Start!
	2 полугодие	**Перечень вопросов (тестов)**
1	Модуль 1	Let's Start!
2	Модуль 2	Let's Start!

8.1. Примерные вопросы к экзамену по дисциплине

1 полугодие	
1.	Сети в современной жизни.
2.	Стандартизация в интернет
3.	Стек протоколов TCP/IP
4.	DNS – система доменных имен
5.	World Wide Web (WWW)
6.	Понятие прокси-сервер
7.	Протоколы Интернет прикладного уровня
8.	Использование сетей в сферах науки, образования, культуры и экономики.
9.	Локальные и глобальные сети, требования, предъявляемые к современным вычислительным сетям.
10.	Среды передачи данных в сети.
11.	Сетевые аппаратные компоненты.
12.	Программы, выполняющиеся на клиент-машине
13.	Программы, выполняющиеся на сервере
14.	Насыщенные интернет-приложения
15.	DHTML.
16.	Аппаратура для логической структуризации сети.
17.	Типы организации локальных сетей.
18.	Методы доступа в сети
19.	Протокол HTTP.
20.	Обеспечение безопасности передачи данных HTTP
21.	Cookie
22.	Сетевые программные средства.
23.	Протоколы обмена данными в сети.
24.	Работа с ресурсами в локальной сети.
2 полугодие	
1.	Сети в современной жизни.
2.	Стандартизация в интернет
3.	Стек протоколов TCP/IP
4.	DNS – система доменных имен
5.	World Wide Web (WWW)
6.	Понятие прокси-сервер
7.	Протоколы Интернет прикладного уровня
8.	Использование сетей в сферах науки, образования, культуры и экономики.
9.	Локальные и глобальные сети, требования, предъявляемые к современным вычислительным сетям.
10.	Среды передачи данных в сети.
11.	Сетевые аппаратные компоненты.
12.	Программы, выполняющиеся на клиент-машине
13.	Программы, выполняющиеся на сервере

14.	Насыщенные интернет-приложения
15.	Введение в JScript
16.	Краткая характеристика VBScript
17.	Java-апплеты
18.	ActionScript – общая характеристика.
19.	DHTML.
20.	Аппаратура для логической структуризации сети.
21.	Типы организации локальных сетей.
22.	Методы доступа в сети
23.	Протокол HTTP.
24.	Обеспечение безопасности передачи данных HTTP
25.	Cookie
26.	Сетевые программные средства.
27.	Протоколы обмена данными в сети.
28.	Работа с ресурсами в локальной сети.
29.	Средства сетевого уровня стека TCP/IP, Novell.
30.	Протоколы обмена маршрутной информацией.
31.	Определение сетевых параметров компьютера.
32.	IP маршрутизация
33.	Стандарт CGI
34.	Python
35.	Ruby
36.	Настройка IP-адресации и маршрутизации.
37.	Коммуникационное оборудование в современных вычислительных системах
38.	История возникновения и развития глобальной сети Internet.
39.	Современная структура управления Internet
40.	Сервисы и ресурсы Internet
41.	Microsoft Visual Studio .NET
42.	WWW – Всемирная паутина
43.	Организация процесса разработки web-контента.CMS/CMF.
44.	Принципы функционирования сети Internet
45.	Web-порталы. Классификация web-порталов.
46.	Приложения для социальных сетей
47.	Флоксономия
48.	Семантическая web-сеть
49.	Онтология
50.	Семантические web-сервисы
51.	Универсальный указатель ресурсов URL

9. Методическая разработка аудиторных форм работы

9.1. Краткий курс лекционных занятий

1.Развитие сети Интернет
Интернет — всемирная система объединенных компьютерных сетей, построенная на базе IP и маршрутизации IP-пакетов. Интернет образует глобальное информационное пространство, служит физической основой для Всемирной паутины (World Wide Web, WWW) и множества других систем передачи данных.

Стандартизация в интернет
Сеть Интернет строилась в полном соответствии с принципами открытых систем. В разработке стандартов этой сети принимали участие тысячи специалистов-пользователей сети из вузов, научных организаций и компаний. Результат работы по стандартизации воплощается в документах *RFC*.
RFC (англ. Request for Comments) — документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и Стандарты, широко применяемые во Всемирной сети.

Стек протоколов TCP/IP
Эти протоколы изначально ориентированы на глобальные сети, в которых качество соединительных каналов не идеально. Он позволяет создавать глобальные сети, компьютеры в которых соединены друг с другом самыми разными способами от высокоскоростных оптоволоконных кабелей и спутниковых каналов до коммутируемых телефонных линий.

DNS – система доменных имен
Несмотря на то, что аппаратное и программное обеспечение в рамках TCP/IP сетей для идентификации узлов использует IP-адреса, пользователи предпочитают *символьные имена* (*доменные имена*).

World Wide Web (WWW)
Сеть WWW образуют миллионы *web-серверов*, расположенных по всему миру. *Web-сервер* является программой, запускаемой на подключённом к сети компьютере и передающей данные по протоколу HTTP.

Понятие прокси-сервер
Прокси-сервер (proxy-server) — служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам.

2. Клиент-серверные технологии Web. Протокол HTTP.
Базовым протоколом сети гипертекстовых ресурсов Web является протокол HTTP. В его основу положено взаимодействие «*клиент-сервер*»

Протокол HTTP
HTTP (HyperText Transfer Protocol - RFC 1945, RFC 2616) - протокол прикладного уровня для передачи гипертекста.
Центральным объектом в HTTP является *ресурс*, на который указывает *URI* в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы.

Обеспечение безопасности передачи данных HTTP
Поскольку протокол HTTP предназначен для передачи символьных данных в открытом (незашифрованном) виде, то лица, имеющие доступ к каналу передачи данных между клиентом

и сервером, могут без труда просматривать весь трафик и использовать его для совершения несанкционированных действий. В связи с этим предложен ряд расширений базового протокола направленных на повышение защищенности интернет-трафика от несанкционированного доступа

Cookie

Механизм *cookie* позволяет серверу хранить информацию на компьютере клиента и извлекать ее оттуда.

Инициатором записи *cookie* выступает сервер. Если в ответе сервера присутствует поле заголовка *Set-cookie*, клиент воспринимает это как команду на запись *cookie*. В дальнейшем, если клиент обращается к серверу, от которого он ранее принял поле заголовка *Set-cookie*, помимо прочей информации он передает серверу данные *cookie*. Для передачи указанной информации серверу используется поле заголовка *Cookie.*

3. Сценарии и приложения выполняющиеся на стороне клиента

Программы, выполняющиеся на клиент-машине

Одним из типов программ, предназначенных для выполнения на клиент-машине, являются *сценарии*, например, JavaScript (VBScript). Исходный текст сценария представляет собой часть web-страницы, поэтому сценарий JavaScript передается клиенту вместе с документом, в состав которого он входит. Обрабатывая HTML-документ, браузер обнаруживает исходный текст сценария и запускает его на выполнение.

Программы, выполняющиеся на сервере

Код программы, работающей на сервере, не передается клиенту. При получении от клиента специального запроса, предполагающего выполнение такой программы, сервер запускает ее и передает параметры, входящие в состав запроса. Средства для генерации подобного запроса обычно входят в состав HTML-документа.

Насыщенные интернет-приложения

Насыщенное интернет-приложение (Rich Internet application) – еще один подход, который заключается в использовании *Adobe Flash* или *Java-апплетов* для полной или частичной реализации пользовательского интерфейса, поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью *плагинов*).

Введение в JScript

JavaScript - интерпретируемый язык программирования, стандартизированный международной организацией ECMA в спецификации ECMA-262. Языки JavaScript, JScript и ActionScript являются расширением стандарта ECMA-262.

Краткая характеристика VBScript

Visual Basic Scripting Edition (обычно просто VBScript) — сценарный язык программирования, интерпретируемый компонентом *Windows Script Host*. Он широко используется при создании *скриптов* в операционных системах семейства Microsoft Windows.

Java-апплеты

Java-апплет - это программа, написанная на языке Java и откомпилированная в байт-код. Выполнется в браузере с использованием виртуальной Java-машины (*JVM*). Апплеты используются для предоставления интерактивных возможностей web-приложений, которые не возможны в HTML. Так как байт-код Java платформо-независим, то Java-апплеты могут выполняться браузерами на многих операционных платформах.

ActionScript – общая характеристика.

ActionScript — объектно-ориентированный язык программирования, один из диалектов *EcmaScript*, который добавляет интерактивность, обработку данных и многое другое в содержимое Flash-приложений. ActionScript исполняется виртуальной машиной (ActionScript Virtual Machine), которая является составной частью приложения *Flash Player*.

XAML и Microsoft Silverlight
XAML (eXtensible Application Markup Language) - язык интерфейсов платформы *Windows Vista*.

Понятие о DOM.
DOM (Document Object Model) - объектная модель документа. Это независящий от платформы и языка программный интерфейс, позволяющий программам получать доступ к содержимому документов, а также изменять содержимое, структуру и вид документов.

DHTML.
Динамический HTML или *DHTML* представляет собой набор технологий, которые совместно позволяют создавать интерактивные web-сайты на основе статического языка разметки (*HTML*), языка создания клиентских сценариев (*JavaScript*), языка описания представления документа (*CSS*) и документной объектной модели (*DOM*).

4. Серверные web-приложения.

Стандарт CGI
Круг задач, решаемых Web-сервером, ограничен. В основном он сводится к поддержке HTTP-взаимодействия и доставке клиенту Web-документов. Любые "нестандартные" действия реализуются с помощью специальной программы, которая взаимодействует с web-сервером и клиентом. Это взаимодействие подчиняется определенным правилам.
Основной набор таких правил - *стандарт CGI* (Common Gateway Interface - интерфейс общего шлюза), который определяет порядок запуска программы на компьютере-сервере, способы передачи программе параметров и доставки результатов ее выполнения клиенту.

Сценарии
В плане быстродействия *сценарные языки* можно разделить на:
- *Языки динамического разбора* (например *command.com*). Интерпретатор считывает инструкции из файла программы минимально требующимися блоками, и исполняет эти блоки, не читая дальнейший код.
- *Предварительно компилируемые* (например *Perl*). Вначале считывается вся программа, затем компилируется либо в машинный код, либо в один из внутренних форматов, после чего получившийся код исполняется.

Python
Python — высокоуровневый язык программирования общего назначения с акцентом на производительность и читаемость кода. *Язык Python* сочетает в себе минимализм синтаксиса ядра и большой объём полезных функций в стандартной библиотеке.

Ruby
Ruby — интерпретируемый язык высокого уровня для быстрого и удобного объектно-ориентированного программирования. Ruby обладает независимой от операционной системы реализацией *многопоточности*, строгой *динамической типизацией*, «*сборщиком мусора*» и многими другими возможностями. Многие особенности синтаксиса и семантики языка *Perl* заимствованы в Ruby.

ASP

ASP (Active Server Pages) — технология, разработанная компанией *Microsoft*, позволяющая легко создавать приложения для Web.

Программирование на *ASP* дает разработчикам доступ к интерфейсу программирования приложений *Internet Information Server* с помощью языка сценариев *VBScript* и *JScript*.

ISAPI

Для web-сервера IIS (Internet Information Server) Был разработан специальный программный интерфейс для создания приложений расширяющих стандартные возможности web-сервера.

ISAPI (Internet Server Application Programming Interface) – многозвенный API для IIS.

5. Языки разработки сценариев Perl и PHP

Язык Perl

Язык *Perl* (Practical Extraction and Report Language) — это язык программирования, сильными сторонами которого считаются его богатые возможности для работы с текстом, в том числе реализованные при помощи регулярных выражений. Также язык известен тем, что имеет огромную коллекцию дополнительных модулей CPAN.

Язык PHP

Язык PHP (*PHP:Hypertext Preprocessor*) - один из наиболее популярных сценарных языков ввиду своей простоты, скорости выполнения, богатой функциональности и распространенности исходных кодов на основе лицензии PHP.

6. Введение в C# и платформу Visual Studio.Net

Microsoft Visual Studio .NET

Microsoft Visual Studio .NET - это интегрированная среда разработки для создания, документирования, запуска и отладки программ, написанных на языках .NET.

Эта среда разработки является открытой языковой средой. Наряду с языками программирования, изначально включенными в среду - *C++, C#, J#, Visual Basic,* - в нее могут добавляться любые языки программирования, компиляторы которых создаются сторонними разработчиками. Необходимым условием для включения языков в среду Visual Studio .Net является использование единого каркаса – платформы *Framework.Net*.

Основы C#

- В программах на C#, как правило, нет необходимости в работе с указателями (при сохранении этой возможности), поскольку в нем реализовано автоматическое управление памятью.
- Предусмотрены встроенные синтаксические конструкции для работы с перечислениями, структурами и свойствами классов.
- Имеется полная поддержка программных интерфейсов. Использование двоичных модулей .NET позволяет передавать объекты (по ссылке или по значению) через границы программных модулей.
- Полная поддержка объектно-ориентированных технологий.

7. Архитектура web-приложений ASP.NET. Разработка web-приложений на платформе .NET.

Платформа .NET Framework предоставляет возможность разработки и интеграции web-приложений. ASP.NET является одной из составляющих инфраструктуры .NET Framework и фактически является платформой для создания web-приложений и web-сервисов, работающих под управлением IIS.

Серверные элементы управления ASP.NET
Важной особенностью ASP.NET является использование *серверных элементов управления* на web-странице (элементы *WebForm*), которые являются фактически тэгами, понятными web-серверу. Эти элементы определены в пространстве имен *System.Web.UI.WebControls*.

8. Интерфейсы взаимодействия web-приложений с СУБД.
Сегодня большинство информационных систем в той или иной степени используют базы данных. Не составляют исключение и системы, основанные на web-технологиях. Поэтому организация взаимодействия web-приложений с СУБД является неотъемлемой составной частью web-технологий.
До начал 90-х годов существовало несколько разных поставщиков баз данных, каждый из которых имел собственный интерфейс. Если приложению было необходимо обмениваться данными с несколькими источниками данных, для взаимодействия с каждой из баз данных было необходимо написать отдельный код. С целью решения этой проблемы Майкрософт и ряд других компаний создали *стандартный интерфейс* для получения и отправки данных источникам данных различных типов. Этот интерфейс получил название *open database connectivity* (ODBC).

9. Организация процесса разработки web-контента.CMS/CMF.
Система управления контентом (Content management system, CMS) - компьютерная программа, используемая для создания, редактирования, управления и публикации контента некоторым систематическим образом.
Обычно такие системы используются для хранения и публикации большого количества документов, изображений, музыки или видео.
Система управления web-контентом (Web content management system, WCMS или Web CMS) - програмное обеспечения *CMS* класса, реализованное обычно в виде web-приложения, и предназначенное для создания, и управления HTML содержимым.

10. Синдикация и агрегирование web-контента
Web-синдикация - форма синдикации при которой содержимое web-сайта предоставляется другим многочисленным web-сайтам. Иначе говоря, *web-синдикация* означает создание доступных с сайта *web-потоков* (feed), предоставляющих всем пользователям в форме краткой сводки информацию о новом содержимом, появившемся на сайте (это могут быть новости, сообщения из форума и др.).
Создание набора web-потоков, которые доступны одновременно в одном месте называется *аггрегированием*. Для этого используются специальные *аггрегаторы*.

11. Web-порталы. Классификация web-порталов.
Создание набора web-потоков, которые доступны одновременно в одном месте называется *аггрегированием*. Для этого используются специальные *аггрегаторы*.
Портлеты - подключаемые программные компоненты пользовательского интерфейса, управляемые и отображаемые в *web-портале*. *Портлеты* генерируют фрагменты кода разметки, которые внедряются на страницу портала. Страница портала представляет собой набор непересекающихся окон портлетов.

12. Введение в Web 2.0.
Термин *Web 2.0* используется для обозначения новых тенденций в использовании технологий WWW, направленных на расширение творческих возможностей пользователей, более безопасный обмен информацией и взаимодействие между ними.
При этом больший акцент делается на формирование web-сообществ и социально-ориентированных сайтов таких как, например, *блоги* и *видеоблоги*, *фолксономии*, *википедии* и др.

13. Приложения для социальных сетей

Понятие "*Социальный Web*" (*Social Web*) используется для описания того, как происходит социализация пользователей и их взаимодействие друг с другом с помощью сети *WWW*. Основой для объединения пользователей служат самые разнообразные общие интересы.

Фолксономия

Фолксономия (*folksonomy*) — практика и методика совместной категоризации контента (ссылок, фото, видео клипов и т.п.) посредством произвольно выбираемых меток (тегов). Она основана на спонтанном сотрудничестве группы людей с целью организации контента и полностью отличается от традиционных формальных методов классификации на основе *индексных терминов*. Как правило, этот феномен возникает только в неиерархических сообществах, например на общедоступных web-сайтах. Так как участники *фолксономии* контента обычно являются и основными же ее потребителями, использование методики фолксономии приводит к результатам, более точно отражающим *совместную концептуальную модель контента* всей группы.

Семантическая web-сеть

Семантическая web-сеть (Semantic Web) — часть глобальной концепции развития сети Интернет, целью которой является реализация возможности машинной обработки информации, доступной в сети WWW. Основной акцент в этой концепции делается на работе с *метаданными*, однозначно характеризующими свойства и содержание ресурсов WWW, вместо используемого в настоящее время *текстового анализа* документов.

Онтология

Онтология - это попытка всеобъемлющей и детальной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из иерархической структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области.

Семантические web-сервисы

В то время как совокупность ресурсов и их метаданных можно считать статической частью *семантической паутины*, её динамическую часть представляют *семантические web-сервисы* - законченные элементы программной логики с однозначно описанной семантикой, доступные через Web и пригодные для поиска, композиции и выполнения.

14. Введение в Web 3.0.

Web 3.0, согласно определению Джейсона Калаканиса (Jason Calacanis), руководителя Netscape.com, создателя поискового стартапа Mahalo.com и сети сайтов Weblogs — это высококачественный контент и сервисы, которые создаются талантливыми профессионалами на технологической платформе Web 2.0.

Главная идея Web 3.0 состоит в том, что пользователь, который до этого единолично был вовлечён в процесс формирования контента, отныне творит коллективно, и его партнерами, помимо других пользователей, являются эксперты направлений, причём статус пользователя может быть изменён на экспертный, равно как и форма сотрудничества создателя контента и портала. Эксперт должен выступить своеобразным модератором публикуемого контента.

9.2. Содержание практических занятий

Практическая работа №1
«Структура HTML-документа»

Цели:

1. *создать HTML-страничку;*
2. *использовать комментарии.*

HTML содержит типы элементов, представляющих параграфы, гипертекстовые ссылки, списки, таблицы, изображения и т.д. Каждое *объявление типа элемента* обычно описывает три части: начальный тег, содержимое и конечный тег.

Название элемента появляется в *начальном теге* (`<название-элемента>`) и в *конечном теге* (`</название-элемента>`). Некоторые элементы HTML допускают отсутствие конечного тега. Некоторые типы элементов HTML не имеют содержимого. Такие *пустые* элементы никогда не имеют конечного тега. **Названия элементов всегда нечувствительны к регистру.** Элементы могут иметь ассоциированные свойства, называемые *атрибутами*, которые могут иметь значения (по умолчанию или устанавливаемые автором). Пары атрибут/значение появляются перед конечным символом ">" начального тега элемента. Любое количество (допустимое) пар значений атрибута, разделённых пробелами, может появляться в начальном теге элемента. Они могут появляться в любом порядке. По умолчанию требуется, чтобы все значения атрибутов были ограничены с использованием двойных или одинарных кавычек, однако в некоторых случаях можно устанавливать значение атрибута без использования кавычек, но рекомендуется использовать знак кавычек даже тогда, когда можно обойтись без него. Названия атрибутов всегда нечувствительны к регистру.

Комментарии HTML имеют следующий синтаксис:

```
<!-- это комментарий -->
<!-- и это тоже комментарий,
занимающий более одной строки -->
```

Основные теги.

Элемент `HTML`

Описание: определяет начало и конец HTML-документ.

Начальный тег: ***не обязателен***

Конечный тег: ***не обязателен***

Пример:

```
<HTML>
...элементы head, body и т.п. идут здесь...
</HTML>
```

Элемент HEAD

Описание: содержит информацию о текущем документе, такую как название, ключевые слова и другие данные, не являющиеся содержимым документа.

Начальный тег: ***не обязателен***

Конечный тег: ***не обязателен***

Элемент TITLE

Описание: **определяет** заголовок страницы или название окна. Каждый документ HTML **обязан** содержать элемент `TITLE` в разделе `HEAD`. Авторы должны использовать элемент `TITLE` для идентификации содержимого документа. Поскольку пользователи часто обращаются к документам вне контекста, авторы должны предоставлять осмысленные заголовки. Заголовок не может содержать разметку (в том числе и комментарии).

Начальный тег: ***необходим***

Конечный тег: ***необходим***

Пример:

<HTML> <HEAD><TITLE>Гуманитарный колледж </TITLE> *...другие элементы заголовка...* </HEAD> ... </HTML>	В заголовке браузера отображается надпись: Гуманитарный колледж.

Метаданные.

HTML позволяет авторам специфицировать метаданные - информацию о самом документе, а не о его содержимом - различными способами.

Элемент META

Описание: Элемент **META** можно использовать для идентификации свойств документа (напр., автора, конечной даты использования, списка ключевых слов и т.д.) и установки значений этих свойств. Каждый элемент **META** определяется в разделе HEAD и определяет пару свойство-значение. Атрибут `name` идентифицирует свойство, а атрибут *content* определяет значение свойства. Например, следующее объявление устанавливает значение для свойства `Author(автор)`: <META name="Author" content="Студент группы …">. Обычно META специфицирует ключевые слова, которые используются поисковыми машинами для повышения качества и скорости поиска. Например, следующее объявление устанавливает значения для свойства keywords (ключевые слова): <META name="keywords" content="Chelaybinsk, КПиЭ">

Начальный тег: ***требуется***

Конечный тег: ***запрещён***

Определения атрибутов:

`name` = строка – устанавливает имя свойства

`content` = строка – определяет значение свойства

Пример:

<HTML> <HEAD> <TITLE> Гуманитарный колледж </TITLE> <META name="author" content="Студент"> <META name="keywords" content="Колледж, студент, Челябинск"> </HEAD> ... </HTML>	В заголовке браузера отображается надпись: Гуманитарный колледж. Автором документа является «Студент» Ключевые слова: Колледж, студент, Бишкек

Элемент BODY

Описание: тело документа. В теле документа находится содержимое документа.

Начальный тег: ***не обязателен***

Конечный тег: ***не обязателен***

Определения атрибутов:

`background` = "url" (url – это строка, задающая путь в структуре каталогов до файла) – установка фоновой картинки

text = color (значение цвета может быть или 16-ричным числом (предваряемым знаком # в следующем формате #RRGGBB, где RR – градация красного цвета, GG – зеленого, BB – синего), или одним из следующих 16 названий цвета) - устанавливает цвет текста (для визуальных браузеров).

Black (Черный) = "#000000"	Green (Зеленый) = "#008000"
Silver (Серебро)= "#C0C0C0"	Lime (Известь) = "#00FF00"
Gray (Серый) = "#808080"	Olive (Оливковый) = "#808000"
White (Белый) = "#FFFFFF"	Yellow (Желтый)= "#FFFF00"
Maroon (Темно-бордовый) = "#800000"	Navy (Темно-синий) = "#000080"
Red (Красный) = "#FF0000"	Blue (Синий) = "#0000FF"

Purple (Фиолетовый) = "#800080" Teal (Чирок) = "#008080"
Fuchsia (Фуксия) = "#FF00FF" Aqua (Аква) = "#00FFFF"

bgcolor = color – установка цвета фона документа

Пример:

<HTML> <HEAD> <TITLE> Гуманитарный колледж </TITLE> </HEAD> <BODY bgcolor="white" text="#000000"> *... тело документа...* </BODY> </HTML>	Текст на странице будет отображаться черным цветом на белом фоне.

Задания:

1. *На жестком диске создать папку с именем HTML.*
2. *Открыть текстовый редактор "Блокнот".*
3. *В окне блокнота создать документ, печатающий в качестве заголовка документа название Вашей специальности.*
4. *Сохранить документ под именем index.html, обязательно с расширением* ***html*** (или ***htm***) *в папке HTML.*
5. *Запустить любой из браузеров, установленный в системе, например, Internet Explorer (Пуск – Программы - Internet Explorer).*
6. *Используя меню Файл – Открыть, открыть в окне браузера свой файл и убедиться, что в строке заголовка напечатано название Вашей специальности.*
7. *Перейти в окно редактора Блокнот и добавить вывод в окне браузера «Гуманитарный колледж».*
8. *Сохраните изменения.*
9. *Перейти в окно браузера. Сейчас, что бы просмотреть внесенные изменения, необходимо нажать кнопку "Обновить".*
10. *Используя метаданные, определить автора документа и ключевые слова: название Вашей специальности и название Вашего колледжа.*
11. *Установить цвет фона документа на свой вкус.*
12. *Установить фоновую картинку, для этого:*
    - 12.1. *на жестком диске найти файл с расширением jpg или gif;*
    - 12.2. *скопировать найденный файл в свою папку;*
    - 12.3. *установить фоновую картинку;*
13. *Изменить цвет текста документа на свой вкус.*
14. *Сохранить результаты, так как следующие задания опираются на результаты предыдущих.*

Практическая работа №2
«Работа с отступами и шрифтами»

Цели:

1. *повторить использование заголовков различных уровней;*
2. *повторить использование абзацы, горизонтальные линии, «бегущие» строки;*

Заголовки: элементы `H1, H2, H3, H4, H5, H6`

Описание: описывают шесть уровней заголовков: от H1 (самый верхний) до H6 (самый нижний).
Начальный тег: ***необходим***
Конечный тег: ***необходим***
Определения атрибутов:

align = left|center|right|justify – выравнивание текста: left – выравнивание по левому краю, center – по центру (по умолчанию), right – по правому краю, justify – выравнивание по правому и левому краям.

Примеры:

<h1>Заголовок первого уровня</h1> <h2 align=”center”>Заголовок второго уровня</h2>	Печатается заголовок первого уровня. Печатается заголовок второго уровня, выровненный по центру.

Параграфы: элемент P

Описание: представляет параграф

Начальный тег: ***необходим***

Конечный тег: ***не обязателен***

Определения атрибутов:

align = left|center|right|justify – выравнивание текста

Примеры:

<p> Первый абзац. <p align=”left”>второй абзац.	Печатается первый абзац. Печатается второй абзац, выровненный по левому краю.

Форсирование обрыва строки: элемент BR

Описание: принудительно обрывает (оканчивает) текущую строку текста, но абзац не заканчивается.

Начальный тег: ***необходим***

Конечный тег: ***запрещён***

Пример:

<p> Первая строка абзаца Вторая строка абзаца.	Печатает абзац такого вида: Первая строка абзаца Вторая строка абзаца.

Изменение шрифта текста: элемент FONT

Описание: определяет вид, размер и цвет шрифта для текста.

Начальный тег: ***необходим***

Конечный тег: ***необходим***

Определения атрибутов:

size = *number* – устанавливает размер шрифта. Возможные значения:

- Целое число от 1 до 7. Устанавливает шрифт в определённый фиксированный размер, представление которого зависит от браузера пользователя.
- Относительное увеличение размера шрифта. "+1" означает: на один размер больше. "-3" означает: на три размера меньше. Все размеры находятся в пределах шкалы от 1 до 7

color = *color* – устанавливает цвет текста

face = string – задает имя шрифта

Пример:

 Строка	Печатает слово «Строка» черным цветом, шрифт «Tahoma» размером 1.

Элементы стиля шрифта: TT, I, B, BIG, SMALL, STRIKE, S и U

Описание:

TT: моноширинный текст, телетайп.

I: курсив.

B: полужирный.

BIG: "большой" шрифт.

SMALL: "малый" шрифт.

STRIKE и S: зачёркнутый текст.

U: подчёркнутый.

Начальный тег: ***необходим***

Конечный тег: ***необходим***

Примеры:

bold italic bold italic <tt>teletype text</tt> <big>big</big> small	**полужирный** *курсив* ***полужирный курсив*** телетайп "большой" шрифт "малый" шрифт

Подиндекс и надиндекс: элементы SUB и SUP

Описание: переводит текст в нижний и верхний регистр

Начальный тег: ***необходим***

Конечный тег: ***необходим***

Примеры:

H₂O E = mc²	H_2O $E=mc^2$

Бегущая строка: элемент MARQUEE

Описание: создания бегущей строки

Начальный тег: ***необходим***

Конечный тег: ***необходим***

Определения атрибутов:

bgcolor = color –цвет фона бегущей строки

loop = number – число повторов анимации бегущей строки (значение может быть целым или infinite – текст будет продолжать бегать пока читатель не перейдет на новую страницу)

direction = left|right|up|down – определяет направление: left (по умолчанию) – справа налево, right – слева направо, up – снизу вверх, down – сверху вниз.

scrollamount = number – скорость движения строки. (Рекомендуется ставить скорость "1", в этом случае строка выглядит более удобочитаемо и не дёргается).

scrolldelay = number – задаёт временной интервал между шагами бегущей строки.

width = number – ширина бегущей строки в пикселях.

height = number – высота бегущей строки. (Если вы делаете бегущую строку в одну строчку, то можно высоту не указывать, она сама подбирается под размер букв).

Примеры:

<marquee>Бегущая строка</marquee>	Надпись «Бегущая строка» двигается влево
<marquee direction=”right” scrollamount=1 scrolldelay=20>Бегущая строка</marquee>	Надпись «Бегущая строка» двигается вправо со скоростью равной 1 и с временным интервалом 20.

Горизонтальные линии: элемент HR

Описание: описывает горизонтальную линию

Начальный тег: ***необходим***

Конечный тег: ***запрещён***

Определения атрибутов:

align = `left|center|right` – определяет горизонтальное выравнивание линии.

size = number – определяет высоту линии.

width = number – определяет ширину линии. Значение может быть числовым и в процентах от ширины окна браузера.

color = *color* – устанавливает цвет линии.

Примеры:

<HR width="50%" align="center">	Линия, расположенная по центру и шириной 50% от размера окна.
<HR size="5" width="100" align="left" color=”#ff0000”>	Красная линия, расположенная слева, высотой 5 точек и шириной 100 точек.

Задания.

1. *Запустить программу Блокнот и открыть документ index.html.*
2. *Изменить документ так, чтобы надпись «Колледж «Права и экономики» была заголовком первого уровня и выровнена по центру, надпись специальности – заголовком второго уровня, также выровнена по центру.*

3. *Добавить в документ параграф: Привет, меня зовут …. Сейчас мы изучаем язык HTML (Hypertext Markup Language).*
4. *Добавить в документ параграф: Мой адрес …, выровненный по правому краю.*
5. *Установить для слова Привет размер 5, цвет – желтый.*
6. *Для всего оставшегося параграфа: размер 4, цвет – maroon.*
7. *После слова привет вставить принудительный обрыв строки*
8. *Для последнего абзаца установить шрифт: размер 2, цвет – зеленый.*
9. *Для слова Привет установить шрифт Impact.*
10. *Для своей фамилии, имен, отчества установить шрифт Arial Black.*
11. *Подчеркнуть фамилию, имя, отчество.*
12. *Слово Привет написать курсивом.*
13. *Большим шрифтом выделить специальность.*
14. *Слово HTML зачеркнуть.*
15. *После слова HTML добавить предложение: Еще мы знаем немного математики, например, разность кубов вычисляется по формуле: $a^3-b^3=(a-b)*(a^2+ab+b^2)$, а сумма членов арифметической прогрессии равна: $S_n=(a_1+a_n)/2$.*
16. *Сделать строку Фамилия, имя, отчество бегущей, установив цвет фона gray, направление слева направо.*
17. *Нарисовать линию на всю ширину экрана под надписью Гуманитарный колледж.*

Практическая работа №3
«Работа со списками»

Цели:

1. *научиться работать со списками;*
2. *научиться структурировать документ.*

Организация текста внутри документа. HTML позволяет определять внешний вид целых абзацев текста. Абзацы можно организовывать в списки, выводить их на экран в отформатированном виде, или увеличивать левое поле. Разберем все по порядку.

Ненумерованные списки: <UL> ... </UL>.

Описание: описывают ненумерованный список.
Начальный тег: ***необходим***
Конечный тег: ***необходим***
Определения атрибутов:

type *= disc|circle|square* – задает стиль меток для данного списка: circle (кружок), disc (закрашенный кружок, по умолчанию) или square (квадрат).

Нумерованные списки: <OL> ... </OL>.

Описание: описывают нумерованный список.
Начальный тег: ***необходим***
Конечный тег: ***необходим***
Определения атрибутов:

type *= 1|a|A|i|I* – задает стиль меток для данного списка: 1(арабские цифры, по умолчанию), a (латинский алфавит, нижний регистр) или A (латинский алфавит, верхний регистр), i (римский алфавит, нижний регистр), I (римский алфавит, верхний регистр).

start *= number* – устанавливает номер первого элемента упорядоченного списка. По умолчанию это "1". Заметьте, что, хотя значением этого атрибута является целое число, соответствующие метки могут быть нечисловыми. Так, если стиль элемента списка это латинские буквы в верхнем регистре (A, B, C, ...), start=3 означает "C". Если стиль – это римские цифры в нижнем регистре, start=3 означает "iii" и т.д.

Каждый элемент обоих списков должен быть определен тэгом <LI>.

Элемент нумерованного и ненумерованного списков: <LI>.

Описание: описывают элемент списка.
Начальный тег: ***необходим***
Конечный тег: ***не обязателен***
Определения атрибутов:

value = *число* – устанавливает номер текущего элемента списка. Заметьте, что, хотя значением этого атрибута является целое число, соответствующие метки могут быть нечисловыми (см. атрибут **start**).

Примеры:

<UL type="disc"> <LI> *первый элемент списка...*</LI> <LI> *второй элемент списка...* </UL>	• *первый элемент списка...* • *второй элемент списка...*
<OL type="1"> <LI value=10> *первый элемент списка...* <LI> *второй элемент списка...* </OL>	10. *первый элемент списка...* 11. *второй элемент списка...*
<OL type="i" start="5"> <LI> *первый элемент списка...*</LI> <LI value="10"> *второй элемент списка...* <LI> третий *элемент списка...* </OL>	*V.* *первый элемент списка...* *X.* *второй элемент списка...* третий *элемент списка...* *XI.* третий *элемент списка...*

Задания.

1. *Запустить программу Блокнот.*
2. *В новом документе оформить документ следующим образом (картинку можно вставить любую. Замечание: 1) найдите на винчестере нужную Вам картинку и скопируйте ее в свою папку, 2) файл назовите obomne.html и сохраните его в своей папке):*

Кратко о себе:
1.Фамилия
2.Имя
3.Отчество
4.Дата рождения.
5.Место рождения.
6.Факультет.
7.Группа.
8.Хобби:
- Первый интерес
- Второй интерес
- Третий интерес

9.Знание компьютера:
- первая программа
- вторая программа
- третья программа

3. *Для своей странички установить фон.*
4. *Строка «Кратко о себе» должна быть написана 7 шрифтом, цвет установить по своему усмотрению, шрифт – полужирный курсив и оформить ее в виде бегущей строки.*
5. *Строки, обозначенные цифрами, должны быть написаны 5 размером шрифта, цвет выбрать по своему усмотрению.*
6. *Хобби и название известных Вам компьютерных программ должны быть написаны 4 размером шрифта, цвет выбрать по своему усмотрению.*
7. *Под строкой Отчество провести линию синего цвета, ширина которой равна 3, во всю страницу.*

8. *Внизу страницы провести линию, под которой написать справа адрес своей электронной почты.*

Практическая работа №4
«Работа со ссылками»

Цели:
научиться работать со ссылками;
научиться структурировать документ.

Основные сведения. Существует три типа ссылок: *внутристраничные* – они задают переходы в пределах одной страницы; *внутрисистемные* – ссылки между страницами в пределах одного и того же сервера; и *межсистемные* – ссылки на страницы, расположенные на удаленных узлах Web. Для определения ссылок предназначен специальный тег, который называется Anchor (якорь).

Элемент А.

Описание: определяет ссылку или якорь.
Начальный тег: ***необходим***
Конечный тег: ***необходим***
Определения атрибутов:

name = "строка" – именует текущий якорь, который может стать якорем назначения для другой гиперссылки. Значением этого атрибута должно быть уникальное имя якоря. Областью видимости этого имени является текущий документ.

href = "строка" – определяет размещение ресурса Web, определяя таким образом ссылку с текущего элемента на якорь назначения, определённый этим атрибутом.

Примеры:

Информация ЧГПУ	Создается метка с именем info. Создается ссылка на сайт ЧГПУ.

Цвет ссылки можно задать при помощи атрибутов тэга **BODY**:

link = "color" – устанавливает цвет непосещённых гиперссылок.
vlink = "color" – устанавливает цвет посещённых гиперссылок.
alink = "color" – устанавливает цвет гиперссылок при выборе пользователем.

Пример:

<body link="#ff0000" vlink="#00ff00">	Цвет непосещенных гиперссылок – красный, посещенных – синий.

Для указания ссылки на электронный ящик в значение атрибута href должно быть "mailto:имя_электронного_ящика".

Пример:

Напишите мне	Создается ссылка «Напишите мне», щелкнув по которой, можно написать и отправить письмо.

1.1. **Внутристраничные ссылки**:

1.1.1. Создать имя (метку) для точки назначения, в которую должен осуществляться переход. Метка создается с помощью тега якоря, используя его атрибут NAME (например, <A NAME="info"> Дополнительная информация </A>). Фраза "Дополнительная информация" при этом никак не будут выделены в тексте документа.

1.1.2. Для создания гиперссылки на эту точку документа используется тег <A> с атрибутом HREF=, при этом к имени якоря присоединяется знак #: <A HREF="#info"> Просмотр дополнительной информации </A>. Такой фрагмент HTML-текста приведет к появлению в документе выделенного фрагмента (в нашем случае фразы Просмотр дополнительной информации), при нажатии на который произойдет переход к строчке Дополнительна информация.

1.2. Внутрисистемные ссылки:

1.2.1. В файле, в который мы хотим перейти необходимо создать метку аналогично п. 1.1.1.

1.2.2. Создать гиперссылку аналогично п. 1.1.2, только имя якоря присоединяется к имени файла через знак # (Кратко о моих увлечениях и хобби можно посмотреть <A HREF=”obomne.html#info”> здесь </A>). При нажатии на выделенный фрагмент произойдет переход строчке #info в файле с именем obomne.html.

1.2.3. ***Задание:*** *В файл index.html добавить абзац «Кратко о моих увлечениях и хобби можно посмотреть здесь» после абзаца со словами «Еще мы изучаем математику …».*

1.2.4. В начале файла obomne.html фразу «Кратко о себе» заключить в теги <A NAME=”info”> и <A>.

1.2.5. В документе main.html слово «здесь» оформить гиперссылкой на документ obomne.html на якорь «info»

1.2.6. Изменить цвет непосещенных гиперссылок на красный, а посещенных на черный.

1.2.7. Опробовать действие гиперссылки.

1.2.8. В файл obomne.html внизу страницы поместить абзац, состоящий из одного слова «Назад», выровненного по центру.

1.2.9. Организовать обратный переход.

1.3. **Межсистемные ссылки.** Используя эти ссылки можно установить связь с любой страницей на любом узле Web.

1.3.1. В этом случае необходимо создать только гиперссылку, например, ссылка на сервер фирмы Microsoft выглядит следующем образом: *<A HREF="http://www.microsoft.com"> Переход на сервер компании Microsoft </A>*. При нажатии на выделенный фрагмент произойдет переход на сервер фирмы Microsoft.

1.3.2. ***Задание****: в документ main.html добавить абзац «Здесь вы можете посмотреть сервер Челябинского государственного педагогического университета».*

1.3.3. Оформить абзац по своему усмотрению.

1.3.4. Оформить абзац как гиперссылку на адрес http://www.preco.ru.

1.3.5. Адрес e-mail внизу страницы сделать ссылкой. Теперь пользователь сможет отправлять электронные письма по этому адресу.

1.3.6. В документе obomne.html добавить гиперссылку на колледжа.

Практическая работа №5
«Графика и мультимедиа»

Цели:

1. *научиться работать с рисунками;*
2. *научиться структурировать документ с элементами мультимедиа.*

1. Изображения *в HTML-документе.*

Вставка изображения в HTML-страницу осуществляется с помощью тэга <IMG>.

Тэг <IMG>.

Описание: внедряет изображение в текущий документ в месте определения элемента.

Начальный тег: ***необходим.***

Конечный тег: ***запрещен.***

Определения атрибутов:

src = “строка” – задающая путь в структуре каталогов до файла с изображением (наиболее распространённые форматы изображений: GIF, JPEG и PNG);

alt = “строка” – определяет альтернативный текст (который появляется при наведении курсора мыши на изображение);

width = “число” – определяет ширину объекта (изображения);

height = “число” – определяет высоту объекта. Для уменьшения времени загрузки страницы с графикой полезно указывать размер изображения. Если он известен еще до загрузки страницы, то браузер может отвести рамку для картинки, а затем загружать текст на страницу. Пока загружается графика, посетитель страницы может начать читать текст.

border = “число” – определяет ширину рамки вокруг объекта;

align = “bottom|middle|top|left|right” – определяет позицию объекта по отношению к окружающему тексту (*bottom* – низ объекта должен быть выровнен вертикально по текущей базовой линии (по умолчанию), *middle* – центр объекта должен быть выровнен вертикально по текущей базовой линии, *top* - верх объекта должен быть выровнен вертикально по верхней границе текущей строки, *left* – прижимает объект к левому краю, *right* – прижимает объект к правому краю).

Примеры:

<img src=”pic1.jpg” width=”100” height=”100” align=”right”>	Вставить картинку pic1.jpg размером 100x100 и обтеканием текстом слева
<img src=”pic1.jpg” border=”2”>	Вставить картинку с рамкой шириной 2 пикселя.

1.1. Задание:

1.1.1. В графическом редакторе нарисовать эмблему Вашей специальности или колледжа и сохранить его под именем emblem.jpg в Вашей папке.

1.1.2. В файл index.html вверху страницы по середине вставить картинку emblem.jpg и ее подписать «Эмблема колледжа (или специальности) …», ширина рамки – 5 пикселов.

1.1.3. Внизу страницы расположить фотографию Колледжа (файл College.jpg). Установить соответствующую высоту, ширину и толщину рамки, выравнивание по левому краю и рядом по середине фотографию, сделать подпись «Гуманитарный колледж».

1.1.4. В броузере отключить отображение графики (используя систему помощи броузера) и заново просмотреть созданный документ.

1.1.5. Нарисовать в графическом редакторе рисунок, в котором красиво написать «Назад».

1.1.6. Создать новый документ, назвав который photos.html, где разместите свою фотографию и сделать подпись к ней. Внизу страницы по середине сделать надпись «Назад», которую оформить в виде гиперссылки на документ index.html.

1.1.7. В документе index.html сделать гиперссылку на файл photos.html, добавив абзац «Здесь вы можете посмотреть мою фотографию».

1.1.8. В документе photos.html внизу страницы рядом с надписью «Назад» расположить картинку, которую оформить в виде гиперссылки на главную страницу.

2. Создание анимированных gif-файлов на примере Corel Photo-Paint.

2.1. Создать ролик, в котором буква за буквой появляется надпись «Группа …» (название Вашей группы), для этого:

2.1.1. Запустите графический редактор Corel Photo-Paint.

2.1.2. Выберите пункт меню Файл|Новый.

2.1.3. Установите размеры изображения 400x50 пикселов, режим – 8-ми битная палитра, цвет фона установите по своему усмотрению, поставьте флажок «Создать фильм», количество кадров оставьте равным 1.

2.1.4. Выбрать инструмент текст и в левом нижнем углу написать заглавную букву «Ф». Установить цвет и размер буквы.

2.1.5. Выделить букву и выполнить команду меню Объект|Комбинировать->Склеить объекты с фоном.

2.1.6. Выполнить команду меню Фильм|Вставить кадр. Отметить опцию «Копировать текущий кадр» и нажать ОК. Таким образом появится второй кадр точно такой же как первый.

2.1.7. На новый кадр добавить следующую букву другого цвета (можно ее расположить по дуге, относительно других букв). Перейти к п. 2.1.5.

2.1.8. Аналогично выполнить действия для каждой буквы.

2.1.9. После создания последнего кадра. Сохранить документ как GIF.

2.2. Добавьте изображение на все страницы вверху.

3. **Звук и видео.** Вы можете добавить на свою страничку звуки или видеоклипы. Они могут запускаться автоматически при загрузке странички. Для этого используется тег <EMBED>, который предназначен для встраивания объектов в документ.

Описание: внедряет объект текущий документ в месте определения элемента.
Начальный тег: ***необходим.***
Конечный тег: ***необходим.***
Определения атрибутов:

src = "строка" – задающая путь в структуре каталогов до файла с изображением (наиболее распространённые форматы изображений: GIF, JPEG и PNG);
autostart = "true|false" – определяет автозапуск звука или видеоклипа;
repeat = "true|false" – определяет после проигрывания записи начинать ли заново;
width = "число" – определяет ширину объекта (изображения);
height = "число" – определяет высоту объекта.
border = "число" – определяет ширину рамки вокруг объекта;
align = "center|left|right" – расположение пульта управления.

Примеры:

<EMBED SRC="media.wav" WIDTH=145 HEIGHT=55 autostart=true repeat=true></EMBED>	Проигрывать WAV файл со стандартным пультом управления, автостартом и с разрешением повтора

3.1.1. Задание:

3.1.1.1. На жестком диске найти файлы с разрешением avi и скопировать один из них в свою папку.

3.1.1.2. Поместить на страничку выбранное видео.

Практическая работа №6
«Работа с каскадными таблицами стилей»

Цели:

1. *научиться работать с каскадными таблицами стилей;*
2. *изучить основные методы работы с CSS, селекторы, атрибуты.*

Задание

Оформить подготовленный структурированный гипертекст, представленный в файле index.html в соответствии со стилем представленным на рисунке index.bmp
При выполнении работы допускается использовать «песочницу» http://cssdesk.com/
и справочник http://htmlbook.ru/css
Для подробного освоения темы по «margin» и «padding» рекомендуется прочитать:
http://habrahabr.ru/blogs/css/121810/
http://habrahabr.ru/blogs/css/123250/
Исходный HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Template</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

```
<style type="text/css" media="all">
@import "style.css";
</style>
</head>
<body>
<div class="content">
  <div class="toph"></div>
  <div class="right">
    <div class="title">GREY</div>
    <div class="nav">
      <ul>
        <li><a href=#>HOME</a></li>
        <li><a href=#>ARTICLES</a></li>
        <li><a href=#>GALLERY</a></li>
        <li><a href=#>AFFILIATES</a></li>
        <li><a href=#>SUPPORT</a></li>
        <li><a href=#>CONTACT</a></li>
      </ul>
    </div>
    <h2>Top Articles:</h2>
    <ul>
      <li><a href=#>NoHeader Template</a></li>
      <li><a href=#>Consectetuer adipiscing elit</a></li>
      <li><a href=#>Lorem ipsum dolor sit amet</a></li>
      <li><a href=#>dolor sit amet consectet</a></li>
    </ul>
    <hr />
    <h2>Links</h2>
    <ul>
      <li><a href=#>any.com</a></li>
      <li><a href="htmlbook.ru/samcss">htmlbook.ru/samcss</a></li>
    </ul>
    <hr />
  </div>
  <div class="center">
    <h2><a href=#>Try sNews 1.4!</a></h2>
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer euismod ante non diam. Sed eleifend odio sed quam. Sed vulputate, <a href=#>turpis at tincidunt</a> porttitor, est elit consequat metus, non dignissim augue mauris quis arcu. Phasellus faucibus blandit eros. Curabitur porttitor ante non est. Maecenas dolor. Aenean egestas sem. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Sed suscipit, nisi sit amet pharetra malesuada, sem velit laoreet sem, vitae iaculis diam neque consequat est. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Pellentesque tincidunt eros non quam. Mauris a magna sit amet libero accumsan auctor. Aenean nec urna non dui lobortis viverra...
    <p class="date">Posted by Avenir <img src="more.gif" alt="" />
 <a href=#>Read more</a> <img src="comment.gif" alt="" />
<a href=#>Comments (2)</a> <img src="timeicon.gif" alt="" /> 21.02.</p>
    <br />
    <h2><a href=#>Heading Item</a></h2>
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer euismod ante non diam. Sed eleifend odio sed quam. Sed vulputate, <a href=#>turpis at tincidunt</a> porttitor, est elit consequat metus, non dignissim augue mauris quis arcu. Phasellus faucibus blandit eros.
```

```
    <p class="date">Posted by James <img src="more.gif" alt="" />
<a href=#>Read more</a> <img src="comment.gif" alt="" />
 <a href=#>Comments (7)</a> <img src="timeicon.gif" alt="" /> 18.01.</p>
    <br />
    <div class="boxad"> Your Ads here...sNews is a completly free PHP and MySQL driven tool for publishing and maintaining news articles on a website.</div>
  </div>
  <div class="footer"></div>
</div>
</body>
</html>
```

Требуемый стиль:

Ход работы

1. Для селектора «body»:
- установить цвет фона тела страницы #7D8085 используя свойство «background» селектора «body»;
- установить шрифт тела страницы 74% Arial, Sans-Serif используя свойство «font»;
2. Для селектора (класса) «.toph»:
- установить неповторяющийся фоновый рисунок «top.jpg», без полей, высотой 39px, с выравниванием по центру используя свойства «background», «height», «margin», «padding»;
3. Для селектора (класса) «.content»:
- установить повторяющийся фоновый рисунок «bg.jpg», без полей, шириной 800px, с выравниванием по центру используя свойства «background», «width», «margin», «padding»;
4. Для селектора (класса) «.title»:
- установить неповторяющийся фоновый рисунок «logo.jpg» с выравниванием по левой стороне используя свойство «background»;
- с полями от верхнего края 10px, от левого 40px, высотой 28px используя свойства «height», «padding»;
- установить размер шрифта 140%, полужирный, цвет #F29900 используя свойство «font»;

5. Для селектора (класса) «.right»:
- установить обтекание слева используя свойство «float»;
- с отступом от правого края 15px, и полем от правого края 1em используя свойства «margin», «padding»;
- установить размер шрифта 95%, полужирный, используя свойство «font»;
- установить ширину слоя 170px, используя свойство «width»;
6. Для селектора (класса) «.footer»:
- установить запрет на обтекание одновременно с правого и левого края используя свойство «clear»;
- установить неповторяющийся фоновый рисунок «bot.jpg» с выравниванием по центру стороне используя свойство «background»;
- установить выравнивание текста по центру используя свойство «text-align»;
- установить высоту слоя в 37px, используя свойство «height»;
- установить автоматическую ширину слоя «auto», используя свойство «width»;
7. Для селектора (класса) «.center»:
- установить обтекание справа, используя свойство «float»;
- установить ширину слоя 530px, используя свойство «width»;
- установить размер шрифта 95%, полужирный, используя свойство «font»;
- установить цвет текста #FFF;
- установить поля и отступы соответственно «margin: 0px 0 5px 35px; padding: 0;»;
8. Установить цвет ссылок для центрального блока:
- цвет основной ссылки #F29900 используя селекторы «.center a»;
- цвет ссылки под курсором #FFF используя селекторы «.center a:hover»;
9. Установить для блока «date»:
- цвет основного текста #ccc
- выравнивание текста по правому краю, используя свойство «text»
- поля и отступы соответственно «margin: 4px 0 5px 0; padding: 0.4em 0 0 0;»
- верхнюю границу блока толщиной в 1px цветом #555, используя свойство «border»;
10. Установить цвет ссылки в #ccc, используя свойство «color» селектора «.date a»;
11. Установить цвет ссылки в #7D8085, используя свойство «color» селектора «.right a»;
12. Установить цвет тегов параграфа и ссылок в #888, используя свойство «color» селекторов «p» и «a»;
13. Для селектора «a»:
- установить наследование фона, используя свойство «background» с параметром «inherit»;
- выключить стили текста, используя свойство «text-decoration»;
14. Для селектора «p» установить отступы и поля соответственно «margin: 0 0 5px 0; padding: 0;»
15. Для селектора «hr»:
- установить высоту 1px;
- установить основной и фоновый цвет в #eee;
- убрать «border»;
16. Для селектора заголовка «h1»:
- убрать отступы и поля;
- установить цвет в #FFF;
- установить жирный шрифт размера 1.8em, гарнитур Arial, Sans-Serif;
- установить наследование фона, используя свойство «background» с параметром «inherit»;
-установить свойство letter-spacing равным «-1px»;
17. Установить цвет для ссылок «a» находящихся в заголовке «h1» в #FFF, и установить для них наследование фона.
18. Для селектора заголовка «h2»:
- установить наследование цвета фона, используя свойство «background» с параметром «inherit»;

- установить отступы и поля согласно «margin: 10px 0 10px 0; padding:0;»
- установить основной цвет в #F29900;
- установить размер шрифта 140%, жирный;

19. Установить цвет для ссылок «a» находящихся в заголовке «h2» в #F29955. Для ссылок под курсором установить тот же цвет и убрать подчёркивание.
20. Применить стили элементов списка согласно инструкциям и вставить комментарии действия свойств:

```
ul { margin: 5px 0 20px 15px;
      padding: 0;
      list-style: none;
}
li { list-style-type: square;
    color: #F29900;
    margin: 0 0 0px 0;
    padding: 0 0 0 0px;
}
li a { color: #7D8085; }
li a:hover { color: #F29900;  }
```

21. Выполнить применение указанных стилей тремя методами: связанным, глобальным и внутренним
22. **Дополнительно**:

- рассмотреть работу базовых макетов построения сайта представленных в http://htmlbook.ru/layout
- ознакомиться с материалами статьи http://habrahabr.ru/blogs/css/126207/

Практическая работа №7
«Экспорт стилей и валидация»

Цели:

1. *научиться экспортировать стили;*
2. *научиться структурировать документ.*

Ход работы

Используя материал предыдущей работы выполнить следующее задание:

1. Сменить заголовок с «GRAY» на «CSS», подобрать логотип.
2. Перенести из учебника материал, по числу пунктов меню начиная с введения.
3. Выполнить хотя бы одно абсолютное и относительное позиционирование (http://ru.html.net/tutorials/css/lesson14.php).
4. Сделать хотя бы один плавающий блок – картинку (см. http://ru.html.net/tutorials/css/lesson13.php)
5. Сделать наслоение (http://ru.html.net/tutorials/css/lesson15.php)
6. Указать у всех используемых шрифтов родовые имена и объяснить зачем это нужно. (http://ru.html.net/tutorials/css/lesson4.php)
7. Выполнить экспорт стилей в файл и осуществить его подключение
   Выполнить валидацию файла стиля (см. http://ru.html.net/tutorials/css/lesson16.php)

Практическая работа №8
«Знакомство с синтаксисом языка JavaScript »

Цели:

1. *ознакомиться с синтаксисом языка;*
2. *иметь представление о методах ввода/вывода информации;*
3. *знать способы объявления переменных;*
4. *уметь применять полученные знания на практике.*

Способы объявления переменных.

Объявление переменной делается оператором **var**:

```
var a;
```

Переменные бывают трех основных типов:

- **number** - любое число (12; -123; 12.3; 1.23e2)
- **boolean** - принимает всего два значения: правда и ложь ("2+2=4" - true; "2*5=-9" - false)
- **string** - текстовая строка ("Любая фраза")

JavaScript сам определит тип переменной при присвоении ей значения. А присвоить значение можно любым из следующих способов:

```
<script>
var a=6363;
var s;
s="Это текстовая строка";
YourName=prompt ("Ваше имя", "Вводить здесь");
</script>
```

Т.е. для объявления переменной не всегда необходимо пользоваться оператором var.

Названием переменной может быть не любая комбинация. Имя переменной должно состоять из латинских букв, цифр, лучше не использовать символы (кроме, символа нижней черты _), нельзя использовать пробелы, имя не должно начинаться с цифры и не должно совпадать с зарезервированным словом JScript.

Упражнение1. Написать программу, которая спрашивала бы у пользователя его имя (например, Маша), а потом говорила «Привет, Маша».

Специальные символы

При выводе информации иногда бывает необходимо вывести и специальные символы. Вот некоторые из них:

\b - возврат на один символ
\f - переход на следующую страницу
\n - переход на новую строку
\r - возврат каретки
\t - табуляция
\' - одинарные кавычки (апостроф)
\" - двойные кавычки
\\ - обратная косая черта
\000 - вставка любого символа по восьмеричному коду (например: \123)
\x00 - то же самое только по шестнадцатеричному коду ASCII (например: \xAC)

Упражнение 2. Известна такая довольно старая шутка. Вы спрашиваете у вашего друга:
"Где ты больше всего любишь обедать?" (Например, он отвечает "В столовой")
"Твоя любимая еда?" ("Торт")
"Твой любимый музыкальный инструмент?" ("Барабан")
"Твоя любимая геометрическая фигура?" ("Круг")
"Твоя любимая скорость?" ("140 км/ч")
После этого вы рассказываете другу такую историю:
"Вы сидите в столовой. Играет романтическая музыка, горят свечи... Вы кладете в рот кусочек вкуснейшего торта, и вдруг на вас падает барабан, и со скоростью 140 км/ч вы превращаетесь в круг!.."

Осуществите эту шутку в JavaScript. Используйте методы prompt(), alert() и переменные.

Практическая работа №9
«Знакомство с основными операторами»

Цели:

1. *ознакомиться с математическими операторами;*

2. *иметь представление об объектах языка (объект Math);*
3. *уметь применять полученные знания на практике.*

Математические операторы
+ сложение
- вычитание
***** умножение
/ деление
% остаток от деления

Реализуйте следующий пример:

```
<script>
alert (45+56);
alert (45-56);
alert (45*56);
alert (45/56);
alert (56%45);
</script>
```

Объекты

Объекты позволяют получить дополнительный набор операторов. Начнем с объекта **Math**. Этот объект предоставляет дополнительный набор математических операторов. Вот наиболее распространенные:

- **LN2; LN10: LOG2E; LOG10E** - различные логарифмы
- **E; PI** - число *e* и число π соответственно
- **SQRT2; SQRT1_2** - корень квадратный из двух и 0,5 соответственно

Реализуйте следующий пример:

```
<script>
alert (Math.Ln10);
a=Math.E;
alert (a);
</script>
```

- **sin(); cos(); tan(); asin(); acos(); atan()** - синус, косинус, тангенс, арксинус, арккосинус, арктангенс
- **abs()** - модуль числа
- **sqrt()** - квадратный корень числа
- **pow()** - степень числа (в скобках через запятую основание и показатель степени)
- **exp()** - экспонента числа
- **log()** - логарифм числа
- **random()** - псевдослучайное число
- **round()** - округление числа
- **ceil()** - округление по избытку
- **floor()** - округление по недостатку (целая часть от деления двух целых чисел)
- **max()** - максимальное значение из тех чисел, которые через запятую перечислены в скобках
- **min()** - минимальное значение из тех чисел, которые через запятую перечислены в скобках

Упражнение 1. Написать программу, вычисляющую площадь треугольника по формуле Герона.

Упражнение 2. Написать программу, вычисляющую объем усеченного конуса.

Упражнение 3. `Math.random()` дает псевдослучайное число в диапазоне от 0 до 1. Подумайте, как составить генератор псевдослучайных чисел в диапазоне от 1 до 100.

Практическая работа №10
«Условный оператор»

Цели:

1. *ознакомиться с синтаксисом условного оператора;*
2. *получить навыки работы с операторами if и else.*

Существует такой тип переменной как **Boolean**. Операторы условия работают через него. Т.е. если некоторое выражение или переменная принимает значение **true**, то идет процесс выполнения определенного набора операторов. А если же **false**, то выполняется другой набор операторов (или не выполняется ничего).

Для получения булеановских выражений используются специальные **операторы сравнения**:

== равно (5==5)
< меньше (-1<1)
> больше (1>-2)
<> не равно (1<>2)
>= больше или равно (10>5 5=5)
<= меньше или равно (5<10 10=10)

Логические связки:

!= логическое НЕ (1!=2)
&& логическое И (2>1 && 3>1)
|| логическое ИЛИ (2>1 || 0>1)

Если требуется проверить несколько условий, то каждое, помимо общих скобок, берется в свои, отдельные.

Задание 1. Разработать программу, которая говорила бы пользователю «Спасибо!» только в том случае, если он введет ноль, иначе она будет говорить «Упитанный, а не воспитанный».
Используйте оператор условия и операторы сравнения.

Задание 2. Разработать программу, которая бы проверяла, чтобы пользователь ввел число четное и не больше 10. Используйте оператор условия, операторы сравнения, логические связки.

Практическая работа №11
«Разработка калькулятора»

Цели:

1. *ознакомиться с основными подходами в разработке программ;*
2. *получить навыки разработки калькулятора.*

Основы подхода к программированию на JavaScript.

Идея JavaScript очень проста. Все операции, которые можно исполнять в программе на JavaScript, описывают действия над хорошо известными и понятными объектами, которыми являются элементы рабочей области браузера и контейнеры языка HTML. Собственно объектная ориентированность JavaScript на этом и кончается. Никаких классов объектов, а тем более, наследования в JavaScript нет. Есть только объекты с набором свойств и набор функций над объектами, которые называются методами. Кроме методов существуют и другие функции, больше похожие на функции из традиционных языков программирования, которые позволяют работать со стандартными математическими типами или управлять процессом выполнения программы. Еще в JavaScript есть события - аналог программных прерываний. Эти события также ориентированы на работу в World Wide Web, например загрузка страницы в рабочую область Navigator или выбор гипертекстовой ссылки. Используя события, автор гипертекстовой страницы и программы, ее отображающей, может организовать просмотр динамических объектов, например бегущей строки, или управление многооконным интерфейсом.

Для встраивания скриптов в тело HTML-документа используется контейнер SCRIPT. Не все браузеры способны распознавать и исполнять скрипты, поэтому само тело скрипта помещается в контейнер комментария.

Пример

<html> <head> <!—- <script language="JavaScript"> Код сценария </script> --> </head> <body> Код HTML </body> </html>	Пример встраивания скрипта в тело HTML-документа.

В этом примере в заголовок документа (контейнер HEAD) включен контейнер SCRIPT. Далее, в тексте страницы определен комментарий, в который включен текст скрипта.

Необходимо запомнить следующие вещи:

1. Поместив сценарий на JavaScript в разделе <head> документа, вы делаете так, что весь сценарий будет загружен до того, как потребуется его выполнить.
2. Код сценария должен быть заключен в теги комментария HTML для того, чтобы старые броузеры, не понимающие JavaScript, не отображали его на экране.
3. Регистр, которым написаны буквы, в JavaScript имеют значение.

Отступление. *Для того чтобы можно было обратиться к какому-то объекту, при его описании используется атрибут ID или name, **например**, <P ID=passage>, теперь к параграфу можно обратиться, используя passage.*

Новые понятия

Прежде чем обратиться к объектам, составляющим объектную модель броузера, следует рассмотреть ряд новых понятий, которые мы будем использовать на протяжении ряда следующих занятий.

Объект — совокупность свойств, методов, событий и коллекций, предоставляемая броузером в рамках объектной модели. Все объекты создаются самим броузером, и доступ к ним осуществляется через экземпляр:

Объект**.**метод | событие | коллекция | свойство.

Свойство — переменная в рамках объекта, которая может использоваться для получения каких-то значений или установки новых. Ряд свойств может быть доступен только для чтения.

Метод — процедура или функция, предоставляемая объектом для выполнения каких-либо действий или управления *свойствами* объекта.

Событие — какое-либо действие пользователя или момент работы броузера. Для реакции на события создаются *обработчики событий.*

Коллекция — упорядоченный набор свойств, похожий на массив, доступ к которому осуществляется специальными средствами.

Запуск JavaScript. Что необходимо сделать, чтобы запускать скрипты, написанные на языке JavaScript? Вам понадобится браузер, способный работать с JavaScript - например, Netscape Navigator (начиная с версии 2.0) или Microsoft Internet Explorer (MSIE - начиная с версии 3.0).

События. События и обработчики событий являются очень важной частью для программирования на языке JavaScript. События, главным образом, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит событие *onClick*. Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие *MouseOver*. Существует несколько различных типов событий. Мы можем заставить нашу JavaScript-программу реагировать на некоторые из них. И это может быть выполнено с помощью специальных программ обработки событий. Так, в результате щелчка по кнопке

может создаваться выпадающее окно. Это означает, что создание окна должно быть реакцией на событие щелчка - *Click*. Программа - обработчик событий, которую мы должны использовать в данном случае, называется *onClick*. И она сообщает компьютеру, что нужно делать, если произойдет данное событие. Приведенный ниже код представляет простой **пример** программы обработки события *onClick*:

<form> <input type="button" value="Щелкни" onClick = "alert('Yo')"> </form>	Пример обработки события: при нажатии на кнопке выводится сообщение «Yo».

Данный пример имеет несколько новых особенностей: рассмотрим их по порядку. Вы можете здесь видеть, что мы создаем некую форму с кнопкой. Первая новая особенность - `onClick="alert('Yo')"` в тэге <input>. Как мы уже говорили, этот атрибут определяет, что происходит, когда нажимают на кнопку. Таким образом, если имеет место событие *onClick*, компьютер должен выполнить вызов `alert('Yo')`. Обратите внимание, что в этом случае мы даже не пользуемся тэгом <script>. Функция *alert()* позволяет Вам создавать выпадающие окна. При ее вызове Вы должны в скобках задать некую строку. В нашем случае это *'Yo'*. И это как раз будет тот текст, что появится в выпадающем окне. Таким образом, когда Вы щелкаете на кнопке, наш скрипт создает окно, содержащее текст *'Yo'*. В этом примере мы написали `onClick="alert('Yo')"` - то есть мы использовали и двойные, и одинарные кавычки. Если бы мы написали `onClick="alert("Yo")"`, то компьютер не смог бы разобраться в нашем скрипте, поскольку становится неясно, к которой из частей конструкции имеет отношение функция обработки событий onClick, а к которой - нет. Поэтому Вы и вынуждены в данном случае перемежать оба типа кавычек. Не имеет значения, в каком порядке Вы использовали кавычки - сперва двойные, а затем одинарные или наоборот. То есть Вы можете точно так же написать и `onClick='alert("Yo")'`.

Вы можете использовать в скрипте множество различных типов функций обработки событий. Сведения о некоторых из них мы получим в данном описании, однако не о всех.

Функции. В большинстве наших программ на языке JavaScript мы будем пользоваться функциями. В большинстве случаев функции представляют собой лишь способ связать вместе нескольких команд. Давайте, к **примеру**, напишем скрипт, печатающий некий текст:

```
<html>
<script language="JavaScript">
function myFunction()
{
 document.write("Добро пожаловать на мою страницу!<br>");
 document.write("Это JavaScript!<br>");
}
</script>
<body>
<form>
<input type="button" value="Вызов функции" onClick="myFunction()">
</form>
<body>
</html>
```

В этом скрипте мы определили некую функцию, состоящую из следующих строк:

```
function myFunction() {
 document.write("Добро пожаловать на мою страницу!<br>");
 document.write("Это JavaScript!<br>");
}
```

Все команды скрипта, что находятся внутри фигурных скобок - {} - принадлежат функции *myFunction()*. Это означает, что обе команды document.write() теперь связаны воедино

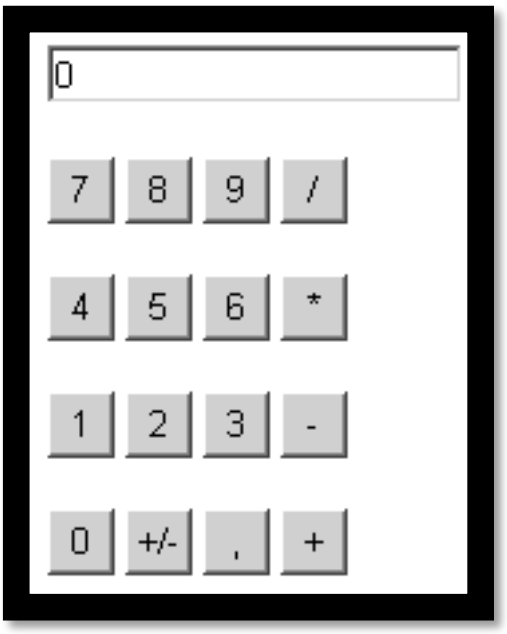
и могут быть выполнены при вызове указанной функции. Можно увидеть, что здесь при нажатии на кнопку осуществляется вызов функции *myFunction().*

Замечание:

1. Обратите внимание на строку onClick="myFunction()" при описании кнопки. Событие onClick возникает, когда пользователь щелкает левой кнопкой по объекту, и мы обрабатываем данное событие, вызывая функцию myFunction, описанную выше.
2. Обратите внимание, что даже если мы описываем функцию без параметров, например, *myFunction(),* как при описании, так и при вызове после имени функции мы должны указывать круглые скобки.

<u>Практическая часть, написание калькулятора.</u>

1. Используя программу Блокнот создать следующую форму. Замечания: используя атрибут ID, присвоить форме идентификатор calc, имена кнопок можно задать произвольно, чтобы все кнопки были одинакового размера (25x25), можно описать соответствующий класс CSS.



2. Описать функцию с именем zero(), в теле которой написать следующий код: идентификатор_формы.имя_текстовой_строки.value+=’0’, это означает, что мы хотим добавить в строку цифру 0; ту же запись можно было записать так: идентификатор_формы.имя_текстовой_строки.value= идентификатор_формы.имя_текстовой_строки.value+’0’. Можно пользоваться любой из записей, первая из них – это особенность языка Си.
3. Для кнопки с цифрой нуль обработать событие onClick, вызвав описанную выше функцию следующим образом: <input type="button" value="0" name="B10" onClick=zero()>.
4. Опробовать в действии Вашу страничку.
5. Аналогичным образом описать все цифры, описывая соответствующие функции и обрабатывая событие onClick для соответствующей кнопки, можно пользоваться любой из записей, но первая предпочтительней.
6. Немножко уменьшим объем нашей странички. Все 10 описанных выше функций выполняют одну и туже последовательность действий, а именно добавляет в строку один символ, значит все их можно объединить в одну функцию, в которую в качестве аргумента будем передавать символ. Для этого опишите новую функцию с именем add(d), где d – это аргумент, в котором мы будем передавать символ. В теле функции написать оператор, который в строку добавляет значение переменной d (смотри выше).

7. Изменить обработчики событий для кнопок с цифрами следующим образом, например для кнопки с цифрой 1 нужно написать onClick=add(‘1’). Для кнопки с запятой обработать событие onClick, вызвав функцию add, передавая в качестве параметра ‘,’.
8. Опишем функцию с именем znak(c), которая будет запоминать знак операции, первое число и очищать строку. Для этого выше всех функций описать три переменные с именами oper (знак операции), pervoe (первое число), vtoroe (второе число) следующим образом: var oper, pervoe, vtoroe, то есть мы описали переменные, значения которых будет известно во всех функциях. Теперь опишем функцию znak(c), где c – параметр для хранения знака операции. Сначала в переменную oper помещаем значение переменной c, затем, используя функцию eval для преобразования строки в число, в переменную pervoe помещаем значение текстовой строки и очищаем текстовую строку, помещая в нее значение, равное нулю. Обработать событие щелчок по кнопкам со знаками операций, вызвав описанную выше функцию.
9. Опишем функцию chet(), которая будет подсчитывать результат. Сначала переменной vtoroe присвоим значение, хранящееся в текстовой строке (смотри предыдущий пример), затем используя условный оператор можем вычислить результат, например, если знак «+», то условный оператор выглядит следующим образом: if (op = = '+') идентификатор_формы.имя_текстовой_строки.value=first+second;. Аналогично написать для всех знаков.
10. Сохранить документ и опробовать в действии.
11. Самостоятельно: добавить кнопку, которая бы очищала текстовую строку, кнопочку, которая бы меняла знак числа.

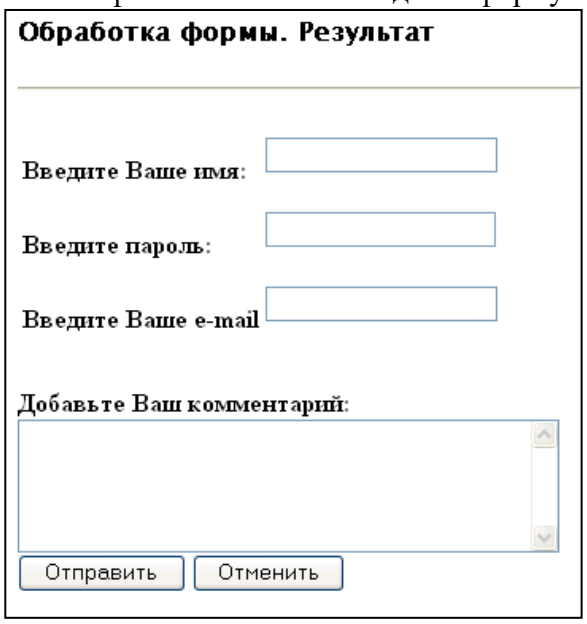
Практическая работа №12
«Обработка данных формы PHP»

Цели:

1. *приобретение навыков обработки данных текстовых полей;*
2. *ознакомление со структурой стандартной формы.*

Порядок выполнения работы:

1. Создайте каталог php2 и в нем файл forma.html. Создайте форму по следующему образцу:



Создайте файл обработчик формы, который должен выполнять следующие действия:

- Проверку, все ли поля заполнены:
  If ($_GET[‘nm’]==’ ‘ or ($_GET[‘pass’]==’ ‘ or ($_GET[‘email’]==’ ‘)
  {echo “Вы ввели не все данные”;
  Exit; }
- Выводить приветствие с указанием имени;
- Формировать и отправлять письмо:

```
$komu=”xxx.ru”;
$tema=”Вопрос от “. $_GET[‘nm’].” “.$_GET[‘email’];
$text_p=$_GET[‘quest’];
mail($komu, $tema, $text_p);
echo “Ваш вопрос отправлен администратору”;
```

Вопросы для самоконтроля

1. Какие типы переменных поддерживает язык PHP?
2. В чем отличие php-страницы и html-страницы?

Практическая работа №13
«Создание счетчика посещений»

Цели:

1. *приобретение навыков использования функций обработки файлов;*
2. *ознакомление со структурой стандартной формы.*

fopen(имя файла, режим работы) – возвращает число – дескриптор открытого файла, по которому можно обращаться к открытому файлу.

Режим работы имеет 2 составляющие:

- способ работы с информацией (текстовый (t) и бинарный(b));
- способ работы с файлом:

Способ	Описание
R	Файл открывается для чтения, указатель текущей позиции в начале файла. Если файла не существует, возникает ошибка
R+	Файл открывается для чтения и записи, указатель текущей позиции в начале файла. Если файла не существует, возникает ошибка
W	Создается пустой файл и открывается для записи, указатель текущей позиции в начале файла. Если файл существует, он перезаписывается
W+	Создается пустой файл и открывается для чтения и записи, указатель текущей позиции в начале файла. Если файл ∃, он перезаписывается
A	Файл открывается для записи, указатель текущей позиции в конце файла. Если файла не существует, он создается
A+	Файл открывается для записи и чтения, указатель текущей позиции в конце файла. Если файла не существует, он создается

flock (дескриптор файла, режим блокировки) – блокирует файл для использования других пользователей.

Режимы блокировки:

2- устанавливает блокировку;

3- снятие блокировки;

fgets(дескриптор файла) – считывает данные из файла.

ftruncate(дескриптор файла, размер) – обрезает данные из файла до заданного размера (указывается в байтах). Возвращает TRUE (при успешном выполнении) или FALSE.

fputs(дескриптор файла, данные) – осуществляет запись данных в файл.

fclose(дескриптор файла) – закрытие файла.

die(текст сообщения об ошибке) – выводит текст, переданный в качестве параметра, и осуществляет выход из программы.

Одновременное использование двух функций:

fopen() or die() – если результат выполнения первой функции FALSE, то в этом случае выполняется вторая функция.

Порядок выполнения работы

Количество посещений любой страницы хранится в текстовом файле с именем counter.txt.

1. Ввести код программы-счетчика посещений counter.php

```
<?php
$f=fopen(“counter.txt”, “a+t”) or die(“Невозможно открыть файл”);
flock( $f, 2);
```

```
$s = fgets($f);
$s+=1;     // $s=$s+1;
ftruncate ($f, 0);
fputs ($f, $s);
flock ($f, 3);
fclose($f);
echo $s;
?>
```

2. Открыть код страницы forma.html первой практической работы.
3. Добавить код для подключения счетчика в нижней части левой панели:

```
<?php
echo “Количество посещений – “; require_once(“counter.php”);
?>
```

Require_once(имя файла) – подключает модуль, имя которого указано в параметре. В качестве модуля используют программы PHP или HTML.

4. Заменить расширение файла: forma.php

Практическая работа №14
«Вычисление значения функции»

Цели:

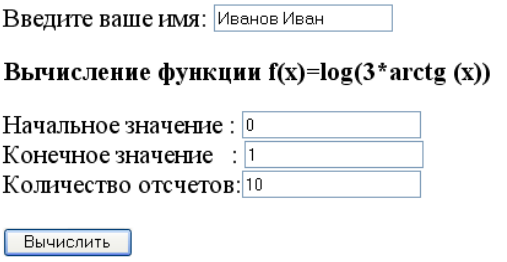
1. *закрепление навыков отладки PHP-кода;*
2. *ознакомление со структурой стандартной формы.*

Описание и вызов функций:

```
<?php
function first_function() {
  echo "<h4>Первая пользовательская функция</h4>";
    function second_function() {
     echo "<h5>Вторая пользовательская функция</h5>";
   }}
first_function();
second_function();
?>
```

3. Порядок выполнения работы:

1. Создать форму следующего содержания:



2. Написать обработчик данных формы, который представляет результаты в виде следующей таблицы:

Задание выполнил: Иванов Иван

Таблица значений функции f(x)=log(3*arctg (x))

x	f(x)
0	-INF
0.1	-1.20729178113
0.2	-0.523933356729
0.3	-0.134251213424
0.4	0.13235994667
0.5	0.329981810013
0.6	0.483202700004
0.7	0.605505364795
0.8	0.705185837594
0.9	0.787750431167
1	0.857047813398

Практическая работа №15
«Использование массивов»

Цели:

1. *закрепление навыков* обработки данных формы*;*
2. *закрепление навыков использования массива в программе.*

В PHP существуют различные методы инициализации массивов:

1. простое присвоение значений

```
<?
  $car[] = "passenger car";
  $car[] = "land-rover";
  echo($car[1]); // выводит "land-rover"
?>
```

2. явное указание индекса массива:

```
<?
  $car[0] = "passenger car";
  $car[1] = "land-rover";
  echo($car[1]); // выводит "land-rover"
?>
```

3. использование конструкции array():

```
<?
  $car = array ("passenger car", "land-rover");
  echo($car[1]); // выводит "land-rover"
?>
```

4. явное указание индексов (в этом случае применяется оператор =>)

```
<?
  $car = array ("passenger car", 5 => "land-rover", "station-wagon","victoria");
  echo($car[0]); echo("<br>"); // выводит "passenger car"
  echo($car[5]); echo("<br>"); // выводит "land-rover"
  echo($car[6]); echo("<br>"); // выводит "station-wagon"
  echo($car[7]); // выводит "victoria"
?>
```

5. индексами массива могут быть и строки:

```
<?
  $car = array ("pc" => "passenger car", "lr" => "land-rover");
  echo($car["lr"]); echo("<br>"); // выводит "land-rover"
  echo($car["pc"]); // выводит "passenger car"
?>
```

Для обработки элементов массива используют:

1. цикл FOREACH

```
foreach (array as [$key =>] $value)
{
  statements;
}
```

Пример:

```
<?
  $car = array ("passenger car", "land-rover", "station-wagon", "victoria");
  foreach ($car as $index => $val)
     {
    echo ("$index -> $val <br>");
     }
?>
```

Как видно из синтаксиса, переменная $key необязательна и может быть опущена:

```
<?
  echo (
    "available cars: <br> <ul>"
  );
  $car = array ("passenger car", "land-rover", "station-wagon","victoria");
  foreach ($car as $val)
     {
    echo ("<li>$val</li>\n");
     }
     echo ("</ul>");
?>
```

Порядок выполнения работы

1. Создать новую страницу с формой следующего вида:


```
<h2> Форма для регистрации студентов</h2>
<form action=”1.php” method=POST>
Имя <br> <input type=text name=”first_name” value=”Введите ваше имя”> <br>
```

```
Фамилия <br> <input type=text name=”last_name” ><br>
E-mail <br> <input type=text name=”email” ><br>
<p> Выберите курс, который будете посещать: <br>
<input  type=checkbox name=’kurs[]’ value=’PHP’> PHP <br>
<input  type=checkbox name=’kurs[]’ value=’LISP’> LISP <br>
<input  type=checkbox name=’kurs[]’ value=’C++’> C++ <br>
<input  type=checkbox name=’kurs[]’ value=’UNIX’> UNIX <br>
<p> Что Вы хотите, чтобы мы знали о Вас? <br>
<textarea name=”comment” cols=32 rows=5></textarea>
<input type=submit value=”Отправить”>
<input type=reset value=”Отменить”>
</form>
```

В файле action.php, обрабатывающем эту форму, можно написать следующее:

```
<?php
$str = "Здравствуйте,".$_POST ["first_name"]." ".$_POST ["last_name"] ."!<br>";
$str.= "Вы выбрали для изучения курс по ". $_POST["kurs"];
echo $str;
?>
```

2. Написать обработчик формы 1.php для регистрации участников заочной школы программирования и после регистрации отправить участнику сообщение. По полученным сведениям от зарегистрировавшегося человека, скрипт генерирует соответствующее сообщение. Если человек выбрал какие-то курсы, то ему выводится сообщение о времени их проведения и о лекторах, которые их читают. Если человек ничего не выбрал, то выводится сообщение о следующем собрании заочной школы программистов.

```
<?
// создадим массивы соответствий «курс-время» и «курс-лектор»
$time = array(“PHP”=>”14.30”, “LISP”=>”12.00”, “C++”=>”15.00”, “UNIX”=>”14.00”);
$lector= array(“PHP”=>”Васильев”, “LISP”=>”Иванов”, “C++”=>”Петров”,
“UNIX”=>”Сидоров”);
define(“SIGN”, “С уважением, администрация”);//  определяем подпись как константу
define(“MEETING_TIME”, “18.00”);          //  задаем время собрания
$date=”12 мая”;
$str=”Здравствуйте, уважаемый “.$_POST[“first_name”]. “  “. $_POST[“last_name”].”!<br>”;
$str .= “<br> Сообщаем Вам, что  “;
$lect =””;
$kurses=$_POST[“kurs”];
If (!isset($kurses)) {
  $event= “следующее собрание студентов”;
  $str .= “$event состоится $date “. MEETING_TIME . “<br>”;
} else {
      $event= “выбранная Вами лекция состоится $date <ul>”;
      For ($i=0; $i < count($kurses); $i++) {
          $k=$kurses[$i];
          $lect = $lect  . “<li> лекция по $k  в $time[$k]”;
          $lect  . =”  ваш лектор, $lector[$k])”
}
$event = $event . $lect . “</ul>”;
$str . =”$event”;
}
$str .= “<br>” . SIGN;
echo $str
?>
```

Практическая работа №16
«Разработка базы данных»

Цели:

1. *приобретение навыков создания и управления базой данных с помощью программы phpMyAdmin;*
2. *закрепление навыков использования MySQL.*

Структура базы данных TOVARS:

1. Таблица tovar, содержит учетные записи товаров

№	Название поля	Описание	Тип
1	id	Поле-счетчик	INT
2	name	Название товара	VARCHAR (20)
3	cost	Стоимость	INT
4	kol	Количество товара	INT
5	date	Дата реализации	DATE

Пример записей:

id	name	cost	kol	date
1	Хлеб столовый	24	100	25.03.10
2	Хлеб ржаной	20	50	27.03.10

Выбор данных:
SELECT column1,… FROM table WHERE definition ORDER BY col_name
Добавление данных:
INSERT INTO table VALUES (value1, …)
Удаление данных:
DELETE FROM table WHERE definition

Основные функции для работы с MySQL:

int mysql_connect(string hostname, string username, string password) - создать соединение с *MySQL*. Функция возвращает параметр типа *int*, который больше 0, если соединение прошло успешно, и равен 0 в противном случае.

hostname – имя хоста, на котором находится база данных.

Username – имя пользователя.

Password – пароль пользователя.

int mysql_select_db(string database_name, int link_identifier) - выбрать базу данных для работы. Функция возвращает значение true или false

Database_name – имя базы данных.

link_identifier – ID соединения, которое получено в функции *mysql_connect*. (параметр необязательный, если он не указывается, то используется *ID* от последнего вызова *mysql_connect*)

int mysql_query(string query, int link_identifier) - функция выполняет запрос к базе данных. Функция возвращает *ID* результата или 0, если произошла ошибка.

query – строка, содержащая запрос *link_identifier* – см. предыдущую функцию.

int mysql_result(int result, int i, column) - функция возвращает значение поля в столбце *column* и в строке *i*.

int mysql_close(int link_identifier) - функция закрывает соединение с *MySQL*. Функция возвращает значение *true* или *false*.

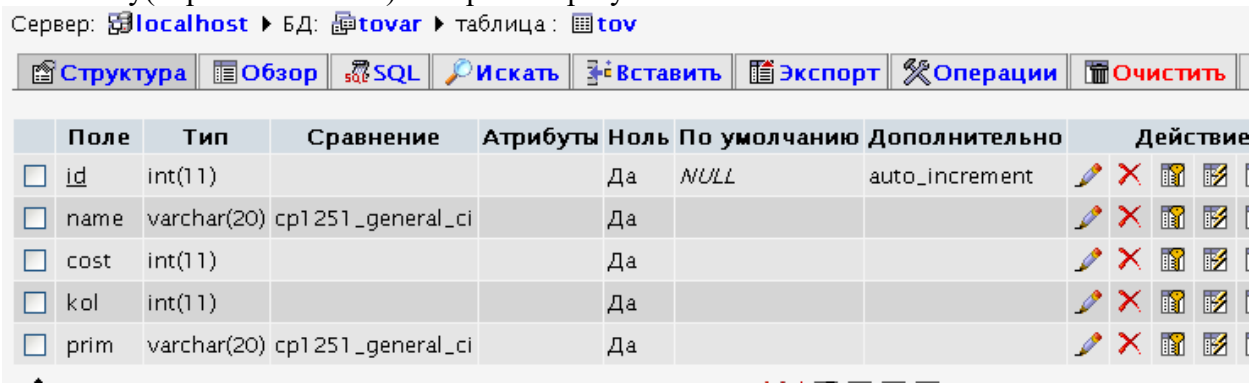
link_identifier – см. выше.

3. Порядок выполнения работы

2. Открыть программу phpMyAdmin: набрать в строке браузера localhost и выбрать пункт phpMyAdmin.
3. В разделе Привелегии добавить нового пользователя homeuser со всеми правами.
4. В поле Создать новую БД ввести имя базы TOVAR – Создать (Create Database)
5. Создать таблицу tov:

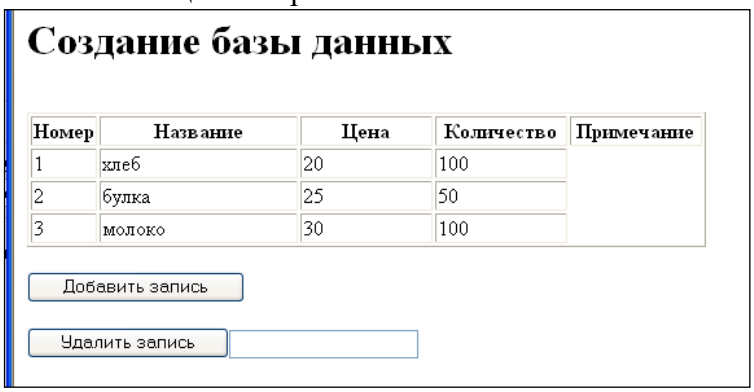
a. В поле Имя ввести tov;
b. В поле Поля – число 5;
c. Кнопка Пошел.

6. Указать название полей таблицы и их тип. В поле Дополнительно указать auto_increment (автоматическое присвоение). Установить переключатель primary key(первичный ключ). Сохранить результат.



	Поле	Тип	Сравнение	Атрибуты	Ноль	По умолчанию	Дополнительно	Действие
☐	id	int(11)			Да	*NULL*	auto_increment	
☐	name	varchar(20)	cp1251_general_ci		Да			
☐	cost	int(11)			Да			
☐	kol	int(11)			Да			
☐	prim	varchar(20)	cp1251_general_ci		Да			

7. Ввести 4 записи для данных базы.
8. Создать папку lab4. В ней файл index.php, в котором описать код вывода данных таблицы на экран.



Пример кода:

```
<?php
echo "<h1>Создание базы данных</h1><br>";
echo "<table border=1><tr align=center><td width=10%><b>Номер</td><td
width=30%><b>Название</td><td width=20%><b>Цена</td><td
width=20%><b>Количество</td><td width=20%><b>Примечание</td></tr>";

$sqlhost="localhost"; $sqluser="homeuser"; $sqlpass=""; $bd="TOVARS";
// соединение с базой данных
mysql_connect($sqlhost,$sqluser,$sqlpass) or die ("нет доступа!".mysql_error());
// выбирает базу для последующей работы
mysql_select_db($bd) or die ("нет соединения".mysql_error());
$zap="SELECT * FROM tovar ORDER BY id";
// выполнение SQL-запроса выбора данных из БД
$zap_res=mysql_query($zap);
while (list($id, $name, $cena, $kol, $prim)=mysql_fetch_row($zap_res))
{
 echo "<tr>
 <td>$id</td>  <td>$name</td>  <td> $cena</td>  <td>$kol</td>  <td>$prim</td>  </tr>";
}
echo "</table>"; ?
```

7. Разместить 2 кнопки Добавить запись и Удалить запись № и текстовое поле для указания № удаляемой записи.
9. Создать файл insert.php, в котором разместить форму для ввода данных в таблицу.

Пример кода:

```
<?php
if (isset($_REQUEST))
{
 foreach($_REQUEST as $key=>$val)
 {$key=$val;}
 }
$sqlhost="localhost"; $sqluser="homeuser"; $sqlpass=""; $bd="TOVARS";
mysql_connect($sqlhost,$sqluser,$sqlpass) or die ("нет доступа!".mysql_error());
mysql_select_db($bd) or die ("нет соединения".mysql_error());
$zap="INSERT INTO tovar( name, cost, kol, prim) VALUES ($name, $cena, $kol, $prim)";
$zap_res=mysql_query($zap);
if (zap_res==true)
   echo "Запись успешно добавлена";    else  echo "Ошибка при записи данных";
?>
```

10. Создать файл delete.php, в котором описать код для удаления записи по заданному номеру.

Пример кода:

```
<?php
if (isset($_REQUEST))
{
 $num=$_REQUEST[num];
 }
$sqlhost="localhost"; $sqluser="homeuser"; $sqlpass=""; $bd="TOVARS";
mysql_connect($sqlhost,$sqluser,$sqlpass) or die ("нет доступа!".mysql_error());
mysql_select_db($bd) or die ("нет соединения".mysql_error());
$zap="DELETE FROM ‘tovar’ WHERE id = $num ";
$zap_res=mysql_query($zap);
if (zap_res==true)
   echo "Запись успешно удалена";
 else  echo "Ошибка при удалении данных";
?>
```

Описание функций работы с БД – http://rusphp.chat.ru/34_MySQLFunctions.html или на сайте PHP.SU

10.Глоссарий

11. **Internet (Интернет):** Всемирное сообщество отдельных компьютерных сетей. Все компьютеры в Интернет взаимодействуют друг с другом согласно определённым правилам (протоколам). Подключившись к Интернет, пользователь получает доступ к широкому спектру разнообразных услуг (сервисов), предоставляемых ему различными серверами (специальными компьютерами) сети. К основным направлениям использования Интернет относятся электронная почта (e-mail), передача файлов от одного компьютера к другому (FTP), удалённый доступ к компьютерам, службы мгновенных сообщений, WWW (World Wide Web или всемирная паутина) и др.
12. **WWW (World Wide Web - всемирная паутина):** Наиболее популярная услуга Internet, интерактивная гипертекстовая информационно-поисковая система. Мультимедийные (т.е. содержащие помимо текста видео-, аудио- и графические ресурсы) блоки данных WWW ("страницы") размещаются на специальных компьютерах, называемых WWW-серверами (или Web-серверами) и принадлежащих отдельным организациям или частным лицам. В основе системы WWW лежит протокол передачи гипертекстовых сообщений HTTP (Hypertext Transfer Protocol), с помощью которого web-документы передаются с главного компьютера (сервера) на компьютеры пользователей. Для доступа к WWW используются специальные программы - web-браузеры (browser - обозреватели).
13. **Applet** – небольшая программа или приложение, обычно написанное на Java, которое запускается браузером пользователя и активирует объекты, например, анимацию или интерактивную таблицу.
14. **ASP** (Active Server Pages) - web-страница (page.asp), созданная с использованием технологии Active Server Pages. Данная технология включает набор средств для формирования на основе скриптовых языков содержимого web-страниц, создания гибких и удобных интерфейсов доступа к базам данных и динамических приложений на web-страницах.
15. **CMS** (Content Management System) - система управления содержанием сайта. Представляет собой программные средства для подготовки, редактирования и публикации информации на сайте, а также средства для управления функциональностью сайта.
16. **CGI** (Common Gateway Interface) - общий шлюзовой интерфейс, с помощью которого происходит запуск CGI-скрипта и взаимодействие с ним. CGI-скрипт является программой, которая выполняется на web-сервере по запросу клиента.
17. **DNS** (Domain Name Service) - Служба доменных имен. Осуществляет преобразование символьного доменного имени в числовой IP-адрес. Построена на принципе распределенного администрирования (делегирования полномочий), когда каждый компьютер или сам “знает” ответ на вопрос, или “знает”, в каком направлении передать данный запрос. Система замкнута и если запрошенная информация имеется на каком-либо компьютере, она будет найдена и передана клиенту. В случае, если вопрос не имеет ответа, клиент получает сообщение о невозможности получения ответа на вопрос. См. также DNS, DNS-сервер.
18. **DNS** (Domain Name System) - система доменных имен. Текстовая система адресации в Интернете, сопоставляющая имя домена и числовой IP-адрес.
19. **DNS-сервер** (Domain Name Server) - сервер доменных имен, в задачу которого входит преобразование текстовых доменных имен в IP-адреса.
20. **GIF** (Graphic Interchange Format) - формат представления графических изображений. Получил наибольшее распространение в Интернет, за счет возможности хранения изображений, имеющих до 256 цветов, поддерживания прозрачности, анимации и способности сохранения в одном файле нескольких изображений. GIF имеет хороший алгоритм сжатия, что крайне важно для создания компактных графических файлов.

21. **HTML** (Hyper Text Markup Language) - язык разметки гипертекста, позволяющий с помощью управляющих меток (тэгов) определять структуру и внешний вид HTML-документа (web-страницы) при отображении в браузере, а также создавать ссылки на другие файлы.
22. **HTTP** (Hyper Text Transfer Protocol) - протокол, обеспечивающий взаимодействие пользователя, запрашивающего доступ к web-документам, с сервером, предоставляющим возможность такого доступа.
23. **IDSN** (Integrated Service Digital Network) - цифровая сеть с интеграцией услуг, позволяющая одновременно передавать по обычным медным телефонным проводам цифровые данные и голос со скоростью до 128 Кбит/с.
24. **ISP** (Internet Service Provider) - поставщик доступа к Интернет. Провайдер является посредником между пользователями и телекоммуникационным оборудованием, необходимым для доступа к различным линиям связи (телефонные кабели, волоконно-оптические кабели, спутниковые каналы). При заключении договора, провайдер предоставляет доступ к различным сервисам Интернет. Услуги провайдера, как правило, являются платными.
25. **URL** (Uniform Resource Locator) - Интернет-адрес, присвоенный каждой web-странице. Каждый URL в Интернет уникален.
26. **Usenet** (USENET, UseNet) – приложение Интернет для обмена сообщениями в пределах групп новостей по интересам, «всемирная доска объявлений». Одно из старейших приложений Интернет (существует с 1979 года).
27. **Web-сервер** - компьютер со специальным программным обеспечением, обеспечивающий доступ многих пользователей к расположенной на нем информации.
28. **Web-страница** (HTML-документ) - логическая единица Интернет (точнее, Всемирной паутины), однозначно определяемая адресом (URL). Физически представляет собой HTML-файл. Может содержать текст, изображения, аудио- и видеофрагменты, Java-апплеты и другие элементы. Web-страница может быть статической или динамически сгенерированной (примеры динамических страниц - перечни результатов, выдаваемые поисковыми машинами). В случае использования фреймов, каждый фрейм рассматривается в качестве отдельной страницы. Страницы загружаются и просматриваются пользователем на свой компьютер с помощью браузера. Логически связанная совокупность web-страниц образует сайт.
29. **Баннер** - статичное или динамичное изображение, размещаемое на страницах сайта с целью рекламы (продвижения) чего-либо. Стандартный размер баннера 468 на 60 пикселов.
30. **Браузер** (Browser) - клиентская программа для работы во Всемирной Паутине (WWW). Позволяет пользователю просматривать содержание web-страниц. Браузер обращается к web-серверу (сайту), запрашивает HTML-документ, интерпретирует полученную информацию и отображает документ на экране компьютера. Браузеры делятся на графические и текстовые. Последний вариант браузеров, примером которого является Lynx, в настоящее время практически полностью вышли из употребления. Примеры браузеров: Mosaic, Netscape Navigator, Internet Explorer, Opera, Mozilla.
31. **Всемирная паутина** (World Wide Web - WWW) - приложение Интернет, в основе которого лежит гипертекст. Позволяет пользователю получить доступ к огромному массиву документов, расположенных на web-серверах по всему миру, и легко перемещаться между ними с помощью гиперссылок. Наполнение Всемирной паутины составляют текстовые материалы и все виды объектов мультимедиа (изображения, аудио- и видеофайлы, анимация и др.). В настоящее время Всемирная паутина представляет собой место, где все общественные институты и частные граждане размещают собственные электронные представительства, многие из которых выполняют традиционно присущие им функции в условиях цифровой среды.

32. **Гиперссылка (**Hyperlink) - слово или изображение в электронном документе, содержащие ссылку на другие файлы, Щелчок "мышью", по гиперссылке позволяет перейти к другому файлу или фрагменту электронного документа. Как правило, гиперссылки выделяются цветом. При наезде на них "мышью", вместо стрелки появляется изображение руки с указательным пальцем.
33. **Гипертекст** (Hypertext) - электронный текст, содержащий в своей структуре ссылки на адреса других файлов.
34. **Главная страница** (Home page) - начальная (титульная) страница web-сайта. На главной странице размещаются общие сведения о сайте с указанием того, что представлено во всех его разделах. Внешние ссылки на ресурс, как правило, делаются именно на главную страницу, поэтому число ее посещений намного больше, чем любых других страниц сайта.
35. **Доменное имя** (доменный адрес) - уникальный текстовый идентификатор компьютера (хоста), подключенного к Интернет. Состоит из слов, написанных латинскими буквами и разделённых точками. Пробелов и других знаков препинания в доменных именах нет. Каждому доменному имени соответствует определенный IP-адрес или несколько IP-адресов. Например, доменному имени www.rbc.ru соответствует IP-адрес 194.186.36.150. Доменные имена являются составляющей частью URL, указывающих на конкретные web-страницы. Доменные имена преобразовываются в IP-адреса службой DNS. Система доменных имен создана для удобства пользователей, которым легче запомнить доменный адрес (например, www.harvard.edu, www.fbi.gov, www.louvre.fr или www.ddt.ru), чем числовые значения IP-адресов. Регистрацией доменных имен занимается InterNIC (представитель в России - РОСНИИРОС). Регистрация доменного адреса означает внесение его и соответствующего ему IP-адреса в базу данных DNS-сервера.
36. **Интернет (Internet)** – глобальная компьютерная сеть, объединяющая компьютерные сети, взаимодействующие посредством протоколов TCP / IP.
37. **Клиент** - компьютер, который потребляет ресурсы других компьютеров сети, прежде всего, серверов. Также - программа, вырабатывающая запросы на доступ к удаленным ресурсам и передающая их по сети на определенный компьютер.
38. **Контекстное меню** - список возможных действий, который появляется при нажатии на объект правой кнопкой “мыши”. Для каждого объекта список свой.
39. **Контент** - (Content) - содержание. Под данным термином чаще всего понимается содержательное наполнение электронных ресурсов, например, web-сайтов.
40. **Куки** (Cookies) - элемент данных, которыми web-сервер помечает конкретный браузер при его посещении. При следующем визите сервер уже “узнает” пользователя и может предложить ему информацию с учетом заявленных прежде пристрастий или, наоборот, не показывать клиенту те данные (например, рекламный баннер), которые он уже видел. Cookies не способны читать диск компьютера пользователя. Некоторые их значения хранятся только в течение одного сеанса работы с сервером и удаляются после закрытия браузера. Другие записываются в файл и хранятся на жестком диске в специальных директориях.
41. **Кэш** – (Cache - "тайный запас") - системная папка, в которую компьютер записывает все документы, полученные пользователем из Сети. При запросе документа вторично, показывается содержимое кэша. Наиболее эффективно кэширование, производимое прокси-сервером, который хранит документы, полученные из Интернет всеми сотрудниками организации. Обращение к кэшу в случае повторного запроса одного и того же документа позволяет не только снизить трафик, но и увеличивает скорость предоставления данных клиенту. Единственным недостатком кэширования является возможность получения старой версии документа в случае, если документ на удаленном сервере изменился, а кэш еще содержит старую версию. ежую версию документа.