Федеральное агентство по образованию
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГОУВПО «АмГУ»
Факультет математики и информатики

УТВЕРЖДАЮ
Зав. кафедрой МАиМ
Т.В. Труфанова
8 сентября 2008г.

Современные информационные технологии

Учебно – методический комплекс дисциплины

Составитель: С.А.Подопригора

Благовещенск
2008

ББК
К

Печатается по решению редакционно-издательского совета факультета математики и информатики Амурского государственного университета

Подопригора С.А.

Современные информационные технологии**.** Учебно – методический комплекс дисциплины для студентов очной формы обучения специальности 010501 «Прикладная математика и информатика». – Благовещенск: Амурский гос. ун–т, 2008. − 77с.

I. РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Рабочая программа по дисциплине " Современные информационные технологии " для специальности 010501 «Прикладная математика и информатика».

Курс 5, Семестр 9, Лекции 66 час., Экзамен 9 семестр, Практические (семинарские) занятия (нет), Зачет (нет), Курсовая работа 9 час., Лабораторные занятия 34 час., Самостоятельная работа 40 час., Всего 140 часов.

Составитель: Подопригора С.А., старший преподаватель, факультет математики и информатики, кафедра математического анализа и моделирования.

1 ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

1.1 Цель преподавания учебной дисциплины

Цели дисциплины: дать студентам знания о современных информационных системах и технологиях, сетях, Internet и электронной почте. Курс современные информационные системы и технологии нацелен на изучение современных программных средств и систем, таких как текстовый процессор фирмы Microsoft Word, электронный табличный процессор Excel и знакомство с современными достижениями сетевых технологий и коммуникаций сетью Internet и электронной почтой. Поэтому весь курс условно разделен на три независимых блока. В блоке текстовые процессоры изучаются основные приемы создания и оформления сложных документов, таких как курсовые и дипломные проекты. Во втором блоке изучается электронный текстовый процессор Excel и его возможности при решении

различных научно - технических задач с построением диаграмм и графиков. Проверяется взаимодействие Word и Excel при создании сложных документов. Третий блок знакомит с основами сетевых технологий. Рассматривается использование Internet технологий для поиска и размещения информации в сетях. Завершает курс практика использования электронных коммуникаций (электронная почта E-mail).

1.2 Перечень основных навыков и умений, приобретаемых при изучении дисциплины

В результате изучения курса "Современные информационные системы и технологии" студент должен иметь представление:

о принципах организации операционной системы Windows,

об объектно-ориентированном подходе в современных языках программирования, приложениях и операционных системах,

о структуре и возможностях офисных пакетов программ (MS Office),

структуру и принципы организации глобальной вычислительной сети и ее поисковых системах,

о средствах создания и редактирования документов, предназначенных для размещения в сети Internet,

о назначении электронной почты

Должен владеть:

средствами управления операционной системой Windows, брать сетевые ресурсы, открывать и закрывать приложения, копировать и удалять файлы.

приемами создания сложных документов с использованием технологии OLE,

приемами программирования инженерных задач с применением объектно-ориентированного, событийного языка Visual Basic for Application.

способами отправки и приема сообщений электронной почтой.

Должны иметь опыт:

форматирования сложных документов путем смены шрифтов и внедрением внешних объектов, таблиц, рисунков и пр. в редакторе MS Word.

создания электронных таблиц в среде MS Excel с построением графиков и использование формул для получения решений,

написания программ на языке Visual Basic for Application для создания интерфейсов и алгоритмов управления решением прикладных задач,

поиска информации по конкретной тематике в мировой информационной сети Internet.

создания и размещения HTML документов в информационной сети Internet.

1.3. Перечень дисциплин с указанием разделов (тем), усвоение которых студентами необходимо при изучении данной дисциплины.

Предварительное усвоение иных дисциплин, предусмотренных государственным стандартом высшего профессионального образования для данной специальности, не требуется.

2 СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1 Федеральный компонент

Программа курса «Современные информационные технологии» составлена на основе авторских разработок.

2.2 *Наименование тем, их содержание, объем в часах лекционных занятий* – 66 часов

1. Операционная система Windows - (10 часов).

Общие сведения и приемы работы: файл, каталог, папка, ярлык, промежуточный буфер обмена, имена файлов.

2. Создание сложных документов - (18 часов).

Создание сложных документов с использованием текстового редактора MS WORD: возможности редактора, структура окна, структура документа,

обмен информацией, промежуточный буфер обмена данными Clipboard, редактирование таблиц, форматирование абзацев. Технология OLE: внедрение объектов, связывание объектов.

3. Табличный процессор Excel. (18 часов).

Использования электронного табличного процессора Excel: структура табличного процессора Excel, ввод информации и ее редактирование, способы адресации к ячейкам Excel, основные функции Excel. Построение графиков. Взаимодействие Excel с другими приложениями Windows. Создание макросов. Основы программирование в Excel с использованием языка Visual Basic: команды управления вычислительным процессом, операторы Visual Basic, объект, свойства, методы, подпрограммы и функции.

4. Сетевые технологии и коммуникации (20 часов).

Основы сетевых технологий: конфигурация электронных сетей, протоколы обмена, типы сетей. Internet технологии: глобальная сеть, HTML язык, Internet Explorer, поиск информации, подготовка и редактирование информации. Использование электронной почты для обмена деловой информацией: настройка почты, получение и отправка сообщений, адресная книга.

2.3 Практические занятия, их содержание и объем в часах

Практические и семинарские занятия не предусмотрены.

2.4. Лабораторные занятия, их наименование и объем в часах.

1. Создание сложных документов– 4 часа.
2. Использование технологии OLE– 5 часов.
3. Электронный табличный процессор Excel. Построение графиков и диаграмм– 5 часов

4. Электронный табличный процессор Excel. Программирование с использованием языка Visual Basic– 5 часов.
5. Знакомство с Internet– 5 часов.
6. Создание документа в формате HTML и его редактирование– 5 часов.
7. Электронная почта– 5 часов.

2.5. Курсовой проект (работа), его характеристика.

Предусмотрена курсовая работа, в которой студент демонстрирует навыки Интернет – программирования в связке PHP – MySQL – Apache. Итогом работы должен быть сайт со страницей администратора, страницей для регистрации пользователей, включающий работу с СУБД MySQL. Тематика сайта студентом подбирается самостоятельно.

2.6. Самостоятельная работа студентов

1. Изучение литературы.
2. Разработка структурных схем или схемы решения задачи. Написание программ. Проверка решения.
3. Подготовка макетов сложных документов.
4. Подготовка к промежуточной аттестации и зачету.

2.7 Вопросы к экзамену

1. Понятие информационных технологий. Приведите примеры ИТ.
2. История развития информационных технологий.
3. Современные компьютерные технологии.
4. Понятие информатизации, информационного общества.
5. Понятие документа. Реквизиты документа.
6. Понятие информации и данных. Свойства и примеры информации и данных.

7. Два подхода к внедрению информационных технологий в структуру офиса.
8. Internet-технологии.
9. Мультимедийные технологии.
10. Технология обработки текстовой информации.
11. Технология обработки табличной информации.
12. Технология доступа к данным.
13. Защищенные информационные технологии.
14. Информационные кросс-технологии.
15. Технология распределенной обработки информации.
16. Языки разметки документов.
17. Работа с папками и файлами в Windows.
18. Рабочий стол Widows XP.
19. Назначение кнопок панели Стандартная.
20. Назначение кнопок на панели Форматирование.
21. Интерфейс рабочего окна Microsoft Word.
22. Назначение и функции текстового редактора.
23. Интерфейс рабочего окна Microsoft Word.
24. Основы оформления текстового документа на компьютере.
25. Списки в текстовом редакторе Microsoft Word.
26. Вставка рисунков в текстовом редакторе Word.
27. Вставка таблиц в текстовом редакторе Word.
28. Работа с объектами WordArt.
29. Применение редактора формул в текстовом редакторе Word.
30. Копирование, вставка, поиск и замена в текстовом редакторе Word.
31. Характеристика символа в документе.
32. Назначение и функции табличного процессора.
33. Интерфейс рабочего окна Microsoft Excel.
34. Что представляет собой электронная таблица?
35. Копирование и автозаполнение ячеек в табличном процессоре Excel.

36. Вставка формул в таблицах Excel. Абсолютные и относительные адреса.
37. Форматирование ячеек в табличном процессоре Microsoft Excel.
38. Виды диаграмм, применяемых в табличном процессоре Excel.
39. Функции, применяемые в табличном процессоре Excel.
40. Создание эффективной презентации в MS PowerPoint.
41. Способы создания презентаций.
42. Настройка анимации при создании презентации в PowerPoint.
43. Какова последовательность действий при демонстрации презентаций?
44. Вставка управляющих кнопок на слайд при создании презентации.
45. Вставка гиперссылок на слайд при создании презентации.
46. Текстовые редакторы, применяемые для создания HTML-документов.
47. Обзор команд языка HTML.
48. Создание таблиц в HTML-документах.
49. Списки в HTML-документах.
50. Внешние и внутренние гиперссылки в HTML-документах.
51. Правила описания фреймов в HTML-документах.
52. Браузеры, используемые для просмотра HTML-документа.
53. Тэги, применяемые для форматирования текста.
54. Создание карт-изображений в HTML-документах.
55. Параметры фона в HTML-документах.
56. Вставка рисунков в HTML-документ.
57. Вставка рисунков в HTML-документ. Атрибуты тэга <IMG>.
58. Вставка в HTML-документ звука, видео-информации. Бегущая строка.
59. Параметры фона документа в HTML.
60. Преимущества и недостатки применения фреймов в HTML-документах.
61. Преимущества, недостатки и область применения карт-изображений.
62. Понятие атрибута в HTML-документах. Какие атрибуты вы знаете?
63. Назначение языка HTML, правила записи текста HTML-программы.

Структура HTML-документа.

3 ТРЕБОВАНИЯ К ЗНАНИЯМ СТУДЕНТОВ, ПРЕДЪЯВЛЯЕМЫЕ НА ЭКЗАМЕНЕ

Оценка «отлично» ставится при полном изложении теоретического материала экзаменационного билета, ответах на дополнительные вопросы со свободной ориентацией в материале и других литературных источниках, при правильно выполненной практической части.

Оценка «хорошо» ставится при твердых знаниях студентом всех разделов курса (в пределах конспекта лекций) и при преимущественно правильно выполненной практической части (допускаются ошибки вычислительного характера, небольшие недочеты или неточности).

Оценку «удовлетворительно» студент получает, если дает неполные ответы на теоретические вопросы билета, показывая поверхностное знание учебного материала, владение основными понятиями и терминологией; при неверном ответе на билет или на дополнительные вопросы, при этом, по крайней мере, одно из практических заданий должно быть выполнено верно.

Оценка «неудовлетворительно» выставляется за незнание студентом одного из разделов курса, если студент не дает ответы на теоретические вопросы билета, показывая лишь фрагментарное знание учебного материала, незнание основных понятий и терминологии, либо если не решена ни одна задача из предлагаемых в билете.

4 РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

4.1 Основная литература

1. Гаврилова Т.И., Логинов В.И. Табличного процессор Excel. Решения инженерных задач. Часть 1.- Н.Новгород: Изд-во ВГАВТ, 2001. - 54 с.
2. Электронная помощь в используемых пакетах.
3. Internet http://aqua.sci-nnov.ru/informatika (/index.htm) или Intranet сайт А363-2.

4.2 Дополнительная литература

1. Персон Р., Роуз К. Microsoft Word 97 в подлиннике: Пер. с англ.- СПб.: BHV - Санкт - Петербург, 1997.- 1120 с.
2. Борланд Р. Эффективная работа с Word 7.0 для Windows 95.: Пер. с англ.- СПб: Питер, 1997.- 1104 с.
3. Хомоненко А. Д. Самоучитель Microsoft Word 2000.- СПб.: БХВ - Санкт - Петербург, 1999.- 560 с.
4. Беленький Ю. М., Власенко С. Ю. Microsoft Word 2000.- СПб.: БХВ - Санкт - Петербург, 1999.- 992 с.
5. Уокенбах Д. Библия пользователя Excel 97.: Пер. с англ.- К.: Диалектика, 1997.- 624 с.
6. Долженков В. А., Колесников Ю. В. Microsoft Excel 2000. СПб.: БХВ - Санкт - Петербург, 1999.- 1088 с.
7. Реселман Б., Писли А., Пручняк В., Смит Э. Использование Visual Basic 6.: Пер. с англ.- К.; М.; СПб.: Изд. дом "Вильямс", 1999.- 608 с.
8. Клименко С., Уразметов В. Internet - среда обитания информационного общества.- Протвино:РЦФТИ, 1995.- 327 с.
9. Хоффман П. Internet.- К.: Диалектика, 1995.- 160 с.

II. ГРАФИК САМОСТОЯТЕЛЬНОЙ УЧЕБНОЙ РАБОТЫ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ НА КАЖДЫЙ СЕМЕСТР С УКАЗАНИЕМ ЕЕ СОДЕРЖАНИЯ, ОБЪЕМА В ЧАСАХ, СРОКОВ И ФОРМ КОНТРОЛЯ

График самостоятельной работы определен пунктом 2.6 рабочей программы данной дисциплины.

III. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ СЕМИНАРСКИХ И ПРАКТИЧЕСКИХ ЗАНЯТИЙ (РЕКОМЕНДУЕМАЯ ТЕМАТИКА И ВОПРОСЫ, ФОРМЫ ПРОВЕДЕНИЯ), САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Самостоятельная работа студентов состоит в подготовке к лабораторным работам, практическим занятиям, контрольным работам. Вся рекомендуемая для этих целей литература указана в п. 4 Рабочей программы данной дисциплины. Также самостоятельная работа включает изучение темы «Метод динамического программирования»; в рабочей программе указана литература, в которой представлено изложение данной темы. Возможно использование другой литературы. В случае возникновения вопросов проводятся консультации по обозначенному преподавателем графику.

IV. КРАТКИЙ КОНСПЕКТ ЛЕКЦИЙ (ПО КАЖДОЙ ТЕМЕ) ИЛИ ПЛАН-КОНСПЕКТ

В этом курсе вы узнаете элементарные принципы HTML, все что нужно чтобы создать свою страничку, какие программы вам необходимы, как опубликовать ваши страницы в интернет. Не можете разместить страницу у вашего провайдера ? Есть много других сайтов которые предоставляют место для страниц бесплатно. Не хотите тратить время на изучение HTML, но хотите создать свою страницу? Попробуйте сделать это с помощью интерактивного волшебника.

Создание **веб страницы** не очень сложное занятие. Начальный **HTML** удивительно прост .Веб страница это просто текстовый файл, HTML файл. HTML это компьютерный язык или лучше сказать описательный язык. Набор кодов определяющих как ваша страница будет выглядеть в браузере. HTML переводиться как **Я**зык **Р**азметки **Г**ипертекста. HTML файлы имеют .html или .htm расширения.

Гипертекст это специальный вид текста. Если вы хотите что то найти в книге очень быстро, вы обычно смотрите в оглавление. Гипертекст намного более эффективен в этом отношении. Вы можете связать определенные слова и иллюстрации с другими страницами. Нажав на такую ссылку , вы попадете на другие страницы которые размещены на сервере расположенном на другом конце мира. Гиперссылки это основная сила интернета.

Слово **Разметка** появилось из принципа работы принтера. Разметка говорит принтеру как разместить текст и картинки, какие шрифты использовать, размер шрифта, пробелы и так далее. HTML делает что то похожее. Он говорит вашему браузеру как отобразить гипертекст на экране монитора.

Итак, вы готовы освоить азы HTML. Я обещаю что это не будет очень трудным делом. Все что вам нужно сделать так это провести несколько часов с этим учебником. Вы готовы ? Тогда давайте начнем .

Чтобы создавать веб страницы вам понадобятся некоторые программы. В основном это бесплатные программы , которые можно скачать из интернета. Все программы которые будут упоминаться дальше , вы можете найти на странице списка программ .

Редактор

Первая вещь которая вам понадобиться это программа чтобы писать и редактировать HTML. Предпочтительно использовать самый простой текстовый редактор и писать код полностью самим. Таким образом вы будете понимать любую строчку в HTML. Также можно использовать специально ориентированный HTML редактор. Но советую не использовать функции до тех пор пока вы не освоите азы.

Нельзя использовать текстовый процессор (типа Word) . Программы такого вида создают такого типа файлы которые ваш браузер не сможет прочитать. Также не соблазняйтесь чтобы использовать так называемые WYSIWYG редакторы. Результаты обычно скудные, файлы больших размеров и как результат ваши страницы будут загружаться очень медленно. HTML код, который они производят , настоящий мусор.

Поэтому забудьте обо всех этих навороченных текстовых редакторах и используйте простой.В Windows есть встроенный текстовый редактор Блокнот. Это один из самых хороших вариантов.

Браузер

Следующее что вам понадобиться это браузер. Глагол to browse дословно в переводе с английского значит пастись . Фигурально это значит пролистывать книгу, читать бессистемно . Лучшие из браузеров это Netscape Navigator и Microsoft Internet Explorer. Установите оба из них и тестируйте ваши страницы в обоих браузерах.

FTP программы

Потом вам понадобиться программа с помощью которой вы сможете

размещать выши страницы в интернете.Вам понадобится **FTP** клиент. FTP переводится как протокол передачи файлов.С помощью FTP клиента вы можете загрузить ваши файлы на сервер вашего провайдера. Наиболее часто используемая программа это CuteFTP. Также можно использовать FTP Voyager.

Файловый менеджер

Дальше вам понадобится файловый менеджер. Конечно вы можете использовать Explorer но по моему мнению Windows Commander намного удобнее. Так же есть замечательная программа написанная нашими соотечественниками и она к тому же бесплатна FAR .

Большинство файлов в Интернете находится в сжатом виде. Иногда это может быть самораспаковывающийся файл или установщик. В этом случае вам понадобиться распаковщик. Windows Commander может распаковывать файлы. Winzip одна из самых удобных программ, которая может распаковывать файлы.

Библиотека HTML

Также вы можете многое узнать из справочных библиотек HTML.. Лучшая написана Стефоном Ле Гунте (Stephen le Hunte), справочная библиотека HTML . Веб дизайнерская группа (The Web Design Group) имеет богатую библиотеку HTML.

В заключении хотелось бы сказать несколько слов об ошибках. Невозможно научиться создавать веб страницы только читая о них. Не бойтесь ошибок. Просто учитесь на них. Ну что , приступим ? Список программ можно найти здесь.

HTML – теги

В языке HTML все связанно с **тегами**. Тег это ярлык с надписью , типа ярлыка на вашей рубашке или брюках. HTML тег чем -то напоминает переключатель, с помощью которого вы что-то включаете или выключаете в вашем браузере.

<TAG>

Теги всегда располагаются между значками "меньше чем" (<) и "больше чем" (>). Теги могут быть написаны заглавными и незаглавными буквами. Наиболее мудро в тегах использовать заглавные буквы , что бы они выделялись в тексте. Хотя в этой ситуации , это ваш выбор. И первый и второй вариант дает одинаковые результаты.

<TAG>...</TAG>

Теги в основном используются попарно. **Открывающий тег** и **закрывающий тег**; **включить** и **выключить** . Единственная разница между ними это "слеш" в закрывающем теге. Запомните , что вы будете часто забывать поставить слеш , в результате чего ваши страницы будут сильно

отличаться от того , что вы ожидали там увидеть. Открывающий тег включает, закрывающий тег выключает.

<TAG ATTRIBUTE>...</TAG>

Иногда теги имеют один или несколько **атрибутов**. Атрибуты предоставляют дополнительную информацию для браузера. При размещении картинки в вашей странице, используется специальный тег. Какую картинку вы разместите , указывается в атрибуте. Название файла этой картинки всегда помещается в **двойные ковычки**. В кратце: тег говорит браузеру , ЧТО делать , а атрибут говорит КАК это делать.

<TAG ATTRIBUTE="...">...</TAG>

Всегда обращайте внимание на порядок расположения тегов. Размещайте значки "больше чем" и "меньше чем" в правильном порядке. Размещайте название тега сразу после них, без пробелов. Не используйте пробелы в атрибутах. Когда добовляете информацию для атрибута , после знака "равно" , всегда помещайте ее в двойные кавычки. Всегда должен быть следующий порядок : **<A><B></B></A>**. Если вы разместите теги вот так: **<A><B></A></B>** , все может пойти кувырком.

Запустите Блокнот и напечатаете текст, расположенный ниже. Окройте ваш браузер . Не забываете обновлять вашу страницу, что бы вы могли видеть изменения , которые вносите.

<HTML>
</HTML>

С помощью этих двух тегов , вы говорите браузеру , что этот файл - HTML файл. Вы говорите: HTML файл начинается здесь (<HTML>), и заканчивается здесь (</HTML>).Эти теги используются только один раз. До <HTML>и после </HTML> тегов, обычно никакого текста нет

<HTML>
<HEAD>
</HEAD>
</HTML>

Заголовок

Каждый HTML файл имеет тег HEAD. Он тоже появляются только один раз в вашей странице. Здесь вы располагаете название вашей страницы. Текст между тегами TITLE показывается в самом верху окна вашего браузера. Не используйте другие теги внутри тегов HEAD. Есть теги , которые вы можете использовать внутри тегов HEAD, но эта информация выходит за рамки данного учебника.

<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
</HTML>

Название

Убедитесь, что ваша страница имеет ясное название. Когда кто-то посетит вашу страницу и сделает закладку на нее, единственное, что будет видно в

закладке так, это заглавие. Заглавие также важно для поисковых машин. Оно должно содержать только простой текст. Здесь нельзя использовать другие теги. Заглавие всегда расположено внутри тега HEAD .

```
<HTML>
<HEAD>
<TITLE>простое название</TITLE>
</HEAD>
</HTML>
```

Тело

Каждый HTML файл имеет тело, обозначенное тегом BODY. Внутри этого тега располагается содержание вашей страницы, то что будет видно в браузере. Тег BODY используется тоже один раз. Тег<BODY> располагается сразу после тега </HEAD > ; Тег</BODY> располагается последним перед тегом </HTML>

```
<HTML>
<HEAD>
<TITLE>простое название</TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Файлы

Теперь вы можете сохранить вашу страницу. Сохраните ее как page.html. На некоторых компьютерах нельзя сохранять файлы с расширением, большим чем три буквы. В этом случае вы можете сохранить вашу страницу как page.htm. Поместите этот файл в отдельную директорию , вместе с другими файлами, которые будут частью вашей страницы

Очень часто **название файла** может быть причиной проблем. Большинство веб-серверов используют операционную систему **UNIX** . UNIX различает заглавные и не заглавные буквы. File.html , FILE.HTML или FiLe.HtMl являются для UNIX тремя разными вещами. Все три могут одновременно сосуществовать в одной директории. Поэтому лучше не использовать заглавные буквы в названиях ваших файлов. В Windows допустимы пробелы в названиях файлов, а в UNIX нет. Поэтому не используйте их.

Ваша первая страница

Ну вот вы и создали вашу первую страницу. Можете посмотреть ее в вашем браузере и так же можете разместить на сервере вашего провайдера. Это настоящая веб страница, соответствующая всем правилам. Вы можете дважды нажать на файл в вашем файловом менеджере и он откроется в браузере. Страница пуста, за исключением названия страницы в самом верху окна браузера.

Можете использовать этот файл, как **заготовку**. Любая страница, неважно насколько она сложна, состоит из трех элементов. ВСЕ теги, упоминавшиеся

выше, используются в странице единожды. Если вы измените этим правилам, ваша страница будет выглядеть совершенно иначе, чем вы ожидали.

В любой странице есть текст. Иногда текст и несколько картинок. Текст это самый простой элемент веб страницы. Видимая часть текста должна быть расположена между тегами . Просто добавьте следующее слово к странице, которую вы создали раньше.

Текст появился

Вы можете заполнить вашу страницу простым текстом, но такая страница будет выглядеть довольно скучно. В действительности вы можете изменить ваш текст как хотите, как в любом текстовом редакторе. Шрифт может быть больше или меньше, может быть выделенным, наклонным, цветным. С помощью тега B вы можете выделить ваш текст

<B>выделенный текст</B>

выделенный текст

Наклонный текст может быть сделан с помощью тега I . Советую использовать наклонный текст только с большими шрифтами . Наклонный тест с маленьким шрифтом выглядит довольно ужасно. Только на компьютерах с высоким разрешением такой текст смотрится нормально.
<I>наклонный текст</I>

наклонный текст

Подчеркнуть текст можно с помощью тега U . Однако будьте аккуратны. Подчеркнутый текст выглядит как ссылка. И если использовать довольно часто может действовать на нервы.
<U>подчеркнутый текст</U>

<u>подчеркнутый текст</u>

Все эти теги можно комбинировать
<B><I><U>выделенный подчеркнутый наклонный текст</U></I></B>

<u>выделенный подчеркнутый наклонный текст</u>

Заголовок

Заглавие страницы или параграфа можно выделить с помощью заглавного тега H с номером. Номер определяет размер шрифта, начиная с 1 (наименьший) до 6 (наибольший). Заглавный тег автоматически переносит текст до и после него.

<H1>H1 Огромный</H1>
<H2>H2 Большой</H2>

<H3>H3 Нормальный</H3>
<H4>H4 Маленький</H4>
<H5>H5 Поменьше</H5>
<H6>H6 Малюсенький</H6>

H1 Огромный

H2 Большой

H3 Средний

H4 Маленький

H5 Поменьше

H6 Малюсенький

Обычно используется теги с H1 до H3. Остальные теги слишком маленькие. Не используйте заглавные теги очень часто. Они как бы являются эквивалентом акцента или крика в разговорной речи.
Когда вы хотите начать ваш текст с новой строчке, вы обычно нажимаете клавишу [Enter]. Вы можете попробовать сделать тоже самое в ваше веб странице.
Это
курс
HTML
Однако то, что вы увидите будет выглядеть следующем образом:
Это курс HTML.

Переносы

HTML не начинает текст автоматически с новой строки. Он просто игнорирует их, до тех пор пока строка текста не достигнет края вашего браузера. На первый взгляд это может показаться странным. На самом же деле это сделано с определенным намерением. Окно браузера может иметь совершенно разные размеры. Шрифты бывают разных размеров. Если вы пишите в широко открытом окне , и потом просматриваете этот текст в узком окне, это иногда может привести к совершенно непредсказуемым результатам. Типа чередующихся коротких и длинных строк.

Создатели HTML подумали об этом. Браузер обрывает линию только у края. Или когда он встречает специальный тег , который говорит ему что здесь необходимо произвести перенос линии. Для этого используется простой тег
 или обрыв линии. Если бы вы захотели что бы ваш текст выглядел следующем образом, нужно бы было сделать следующее:

Это
курс
HTML

HTML совершенно не обращает внимание на переносы линии. Вы можете создать целую страницу из одной линии. Она будет выглядеть в вашем браузере.

Это
курс
HTML.

Удобочитаемый код

Для небольшого отрывка HTML это вполне приемлемо. Но написание целых страниц в таком стиле это наверное мазохизм. Такой код невозможно читать. Старайтесь, что бы код выглядел похожим на ваши страницы. Когда используете тег
 делайте перенос линии в вашем редакторе.

Если вы хотите пропустить целую линию, тег BR не всегда помогает. Для этой цели подходит другой тег. Тег параграфа : <P>. Он пропускает линию и начинает текст с новой строки.

Поля

Текст не выглядит очень привлекательно, если он не имеет полей. Также его труднее читать. Поля создаются с помощью тега <BLOCKQUOTE> . Текст будет иметь отступы с обоих сторон.

Если вы хотите что бы ваш текст имел отступы только с левой стороны, вы можете использовать тег <UL> . С помощью этого тега вы можете сделать левое поле шире, чем правое.

<BLOCKQUOTE>
<UL>
<UL>
Расположите ваш текст здесь.
</UL>
</UL>
</BLOCKQUOTE>

Можете посмотреть как такая страница выглядит и найти там дополнительные советы.

Табличная разметка

Создание страницы подобно этой может быть трудновато для начала, для этого необходимо познакомиться с тегом TABLE. Трудновато для начинающих, я думаю. Но если вы хотите увидеть, как это было сделано , посмотрите на страницу посвященную созданию страниц с помощью таблиц. Если хотите научиться использовать таблицы для создания ваших страниц, ознакомьтесь с курсом по использованию тега TABLE.

До сего момента мы рассматривали только текст. Очень хорошо, но ваша страница выглядит очень скучно. Давайте добавим картинки к вашей странице. В HTML для этого используется тег **<IMG>** . Картинка это файл. Тег только указывает на этот файл с помощью атрибута **SRC** . Чтобы избежать проблем, скопируйте файл картинки в ту же самую директорию, где

расположены файлы вашей веб страницы. Иначе ваш браузер не найдет этот файл соответственно вы не увидите его в браузере.

<IMG SRC="pam.jpg">

Размер рисунка

В принципе достаточно указать только название файла. Однако будет лучше, если вы добавите размеры картинки в пикселях. В этом случае браузер высвободит место для картинки и продолжит заружать остальную часть страницы. В противном случае браузер попытается определить размер картинки и только потом продолжит загружать текст. В этом случае вы не увидите текст до тех пор пока все картинки не будут загружены.

Размер определяется атрибутами **WIDTH** и **HEIGHT** (ширина и высота). Если вы не знаете размер вашей картинки , откройте ее в вашем браузере. В Netscape выберете Вид|Информация. Откроется новое окно с информацией о картинке. Вверху будет указан размер в пикселях. Определить размер картинки также можно в большинстве графических редакторов и программ для просмотра графических файлов.

<IMG SRC="pam.jpg" WIDTH="144" HEIGHT="192">

Границы

Большинство браузеров показывают границы вокруг картинки. Ширина границы определяется с помощью атрибута **BORDER** . В большинстве случаев вам не понадобятся границы. Установите значение атрибута равным нулю.

Альтернативный текст

И последний тег **ALT** (альтернативный текст). Некоторые отключают загрузку картинок в браузере, чтобы ускорить загрузку страниц. Существуют также браузеры, которые показывают только текст (например Lynx). В обоих случаях можно указать название картинки.

<IMG SRC="pam.jpg" WIDTH="144" HEIGHT="192" BORDER="0" ALT="Pamela Anderson - pretty as a picture">

Типы файлов

Используйте только GIF или JPEG (JPG) файлы. Не все браузеры поддерживают другие типы файлов.Но их можно конвертировать в jpg или

gif с помощью графических программ типа Paint Shop Pro, Irfanview и т.д.. Убедитесь чтобы ваши файлы были не больше, чем 25 килобайт. Иначе ваша страница будет слишком долго загружаться

Где взять рисунки

А где взять эти картинки ? Это очень просто. Любая картинка, которая отображена в вашем браузере может быть сохранена на ваш жесткий диск. Наведите мышью на картинку, нажмите правую кнопку мыши и выберете пункт меню Сохранить рисунок. Не используйте все картинки подряд. Некоторые картинки имеют авторские права. Есть много сайтов с коллекциями бесплатных картинок.

Если вы хотите скопировать ваш рабочий стол просто нажмите кнопку Print Scrn. Если нажать Alt и Print Scrn одновременно только верхнее окно будет скопировано. Потом вы можете вставить его в любой графический редактор типа IrfanView и сохранить как jpg или gif файл.

Тег BODY обромляет содержание вашей страницы. Однако , с помощью этого тега можно делать еще кое-что. Для примера, вы можете изменить цвет фона . Для этого существует два варианта. Вы можете езменить цвет фона или использовать рисунок для фона .

```
<BODY>
some text
</BODY>
```

some text

Цвет обоев

Некоторые браузеры используют серый цвет для фона. Вы можете изменить начальные установки. Для этого используется тег BGCOLOR . Черный текст на белом фоне - это один из лучших вариантов.

```
<BODY BGCOLOR="White">
текст
</BODY>
```

текст

Цвет фона помещается в двойные кавычки. Кавычки обязательны. Ниже расположена таблица с названиями шестнадцати цветов, которые стандартизированы для основных браузеров.

Название цветов

	aqua		navy
	black		olive

Рисунки для фона

Фон также может быть заполнен рисунком. Можно использовать любой JPEG или GIF файл. Будьте осторожны с использованием больших файлов, которые могут привести к тому, что ваша страница будет загружаться очень долго. Предпочтительно использовать рисунки размером 5000 байт максимум. Сохраняете вашу страницу удобочитаемой. Чем цветастее ваша страница, тем труднее ее читать. Иногда можно бороться с этим делая шрифты выделенными и больше.

<BODY BACKGROUND="venetian.gif">
текст
</BODY>
текст

Размер рисунка вверху всего 124 байта. Он дает эффект жалюзи. Всегда указывайте цвет фона, когда используете фоновый рисунок . Желательно, чтобы цвет фона был близок к цвету фонового рисунка, на случай если у кого-то отключена загрузка рисунков. В этом случае вид вышей странице не сильно изменится.

Можно изменить **цвет фона** на любой, который вам нравится. Например черный. Но тогда текст черного цвета будет не видно. Наверное вы видели белый текст на черном фоне. Как это делается ? С помощью атрибута тега BODY.

Атрибут **BGCOLOR** определяет цвет фона . Атрибут TEXT определяет цвет текста, также как атрибут **LINK** определяет цвет ссылки и атрибут **VLINK** цвет ссылки, которую уже посещали. Атрибут **ALINK** показывает цвет активной ссылки, когда вы нажимаете на нее мышкой .Эти атрибуты задают цвета для всей страницы . Хорошая комбинация это черный с тремя основными цветами .

<BODY BGCOLOR="black" TEXT="yellow" LINK="cyan" VLINK="fuchsia" ALINK="white">

Удобдчитаемость

Вы можете использовать любые цвета, однако помните о том, чтобы ваш текст было удобно читать. Не используйте слишком яркие цвета. Также следите, чтобы не было слишком сильных или слабых конрастов. Черный текст на белом фоне один из лучших вариантов. Или темный текст на светлом фоне. Цвет текста может определяться теми же словами, что и цвет

фона .

Внимание: Эти атрибуты влияют на цвет целой страницы. Их нельзя использовать для части текста или отдельных слов. Эти атрибуты не предназначены для определения цвета отдельных частей. Для этого можно использовать тег **FONT**.

Наиболее важный тег, который вам придется использовать это гипертекстовая ссылка. **<A>** (anchor или якорь). Более распространенное название - гиперссылка или просто ссылка. Ссылки на другие страницы, вот наиболее частое использование этого тега. Также иногда используют название - гипертекстовая ссылка, имея ввиду ссылку на другой гипертекстовый документ. Этот документ указывается в атрибуте **HREF**.

<A HREF="test.html">test</A>

test

Ссылка закрывается тегом </A> . Между открывающим и закрывающим тегом указывается, то что вы увидите на экране. По умолчанию цвет ссылки голубой и подчеркнутый.

Наведя мышкой на такой тег, вы можете нажать на нее. Вы будете перенесены на другую страницу, часто содержащую новые ссылки. Этот тег связывает веб страницы друг с другом. Гипертекстовые ссылки это как бы клей, который связывает веб страницы. Это просто основа интернета.

Использование рисунков

вы можете разместить не только текст между эитми тегами. Вероятно вы видели кнопки и иконки на веб страницах. Это делается с помощью рисунка окруженнонго гиперссылкой. Рисунок становится видимым с помощью тега IMG.

<A HREF="test.html">
<IMG SRC="button.gif" HEIGHT="18" WIDTH="40" ALT="button">
</A>

Email

Почтовая гиперссылка, которую вы будете использовать достаточно часто, предназначена для того, чтобы посылать электронное письмо. Например, если кто-то захотел послать письмо после того, как он посетил вашу страницу. Ссылка внизу может быть использована для того, чтобы послать письмо мне. Если вы нажмете на нее, появится ваша программа, для отправки электронных писем с адресом в поле to:

Email:

<A HREF="mailto:weballey@gmx.net"> gerben@forfree.at</A>

Email: gerben@forfree.at

И конечно, вы можете комбинировать рисунок и ссылку на ваш электронный адрес.
Email:
<A HREF="mailto:weballey@gmx.net"> <IMG SRC="postbus.gif" WIDTH=40 HEIGHT=40 BORDER=0 ALT="gerben@forfree.at"> </A>

В основном вы можете использовать любой рисунок. Нужно только позаботиться чтобы было ясно, что это действительно ссылка, а не просто рисунок. Для этого можно выделить рисунок с помощью границ. Не очень симпатично, но зато понятно, что это ссылка. Также может помочь, если вы добавите к рисунку текст.

Теги

Теги всегда располагаются между скобками "меньше чем" (<) и "больше чем" (>). Большинство тегов имеют закрывающий тег с наклонной чертой (с слешем). Некоторые теги могут использоваться с атрибутами, который влияют на их поведение. Значение атрибутов заключаются в двойные кавычки. Теги всегда как бы обнимают друг друга. Используйте верхний регистр для тегов.

<TAG>
<TAG>...</TAG>
<TAG ATTRIBUTE>...</TAG>
<TAG ATTRIBUTE="...">...</TAG>
<A><B>...</B></A> правильно
<A><B>...</A></B> не правильно

Простая веб страница

Тег HTML начинает и заканчивает страницу. Секция HEAD содержит заглавие (TITLE) страницы, которое появляется в верхней части окна браузера. Между тегами BODY размещается текст страницы. Все эти теги используются в странице только один раз. Все другие теги могут быть использованы только внутри тега BODY.

<HTML>
<HEAD>
<TITLE>Разместите заглавие страницы здесь</TITLE>
</HEAD>
<BODY>
Текст размещается здесь.
</BODY>
</HTML>

Текст

Текст может быть выделенным с помощью тега B , наклонным с помощью тега I , и подчеркнутым с помощью тега U . Эти теги можно комбинировать.

<B>выделенный</B>
<I>наклонный</I>
<U>подчеркнутый</U>

Заголовок

Заголовки создаются с помощью тега Hn , где n = 1 наибольший размер шрифта, n = 6 наименьший размер шрифта.

<H1>Огромный</H1>
<H2>Большой</H2>
<H3>Нормальный</H3>
<H4>Маленький</H4>
<H5>Еще меньше</H5>
<H6>очень маленький</H6>

Переносы

Переносы осуществляются с помощью тега BR, параграфы с помощью тега P.

 перенос
<P> параграф

Поля

Отступ слева делается с помощью тега UL. Тег BLOCKQUOTE создает отступы с обоих сторон. Оба тега добавляют пустую линию сверху и снизу. Комбинация этих тегов может быть использована для создания простых полей.

<BLOCKQUOTE>
<UL>
<UL>
Текст.
</UL>
</UL>
</BLOCKQUOTE>

Рисунки

Рисунок можно добавить с помощью тега IMG. Рисунок это отдельный файл, который вызывается с помощью атрибута SRC. Атрибуты WIDTH и HEIGHT определяют размер картинки и позволяют странице загружаться быстрее. Атрибут ALT описывает рисунок, а атрибут BORDER добавляет границы к рисунку. Используйте только GIF или JPEG файлы.

<IMG SRC="file" WIDTH="size" HEIGHT="size" BORDER="0" ALT="description">

Фон

Фон страницы задается с помощью атрибута BGCOLOR тега BODY. Атрибут BACKGROUND создает фон черепичного цвета. Используйте только стандартные цвета. Используйте цвет фона похожим на цвет рисунков. Этот атрибут используется в странице только один раз.

<BODY BGCOLOR="color" BACKGROUND="image">
Ваш текст здесь
</BODY>

Цвет текста

Общий цвет текста может быть изменен с помощью атрибута TEXT тега BODY, атрибута **LINK** для ссылок, атрибута **VLINK** для ссылок, которые уже посещали и атрибута **ALINK** для активных ссылок. И конечно используйте только стандартные цвета. Эти атрибуты используются один раз в странице.

```
<BODY BGCOLOR="color" TEXT="color" LINK="color"
VLINK="color" ALINK="color">
Ваш текст здесь
</BODY>
```

Гиперссылки

Чтобы соеденить веб страницы друг с другом используется гиперссылки. Для этого используется тег A (anchor или якорь) . Атрибут HREF определяет имя файла на который вы делаете ссылку. Используйте префикс 'http://' если вы делаете ссылку на другой сервер .Вместо текста вы можете использовать рисунок для ссылки. Это делается с помощью тега IMG. Префикс 'mailto:' создает ссылку на ваш электронный почтовый адрес.

```
<A HREF="file">описание</A>
<A HREF="file"><IMG SRC="file"></A>
<A HREF="mailto:email">email</A>
```

Последние заметки

Веб страницы сохраняются как простой текстовый файл с расширением .html. Используйте только нижний регистр для имен файлов, чтобы избежать проблем на UNIX веб серверах. Используйте простой текстовый редактор для создания ваших страниц.

<HTML>...</HTML>

Эти теги начинают и заканчивают каждый файл. Они показывают, что данный файл это HTML файл. Другими словами, что это веб страница. Используйте эти теги только раз на страницу.

<HEAD>...</HEAD>

Эти теги окружают заглавие вашей страницы. Они располагаются сразу после тега HTML. Здесь вы можете разместить название страницы. Эти теги используются тоже один раз на странице.

<TITLE>...</TITLE>

Тег title размещается внутри тега HEAD. Название появляется в верхней части окна вышего браузера. Старайтесь давать ясное название каждой странице. Эти теги используются тоже один раз на странице.

<BODY>...</BODY>

Тег body окружает содерхание страницы. Открывающий тег располагается сразу после тега HEAD , а закрывающий тег сразу перед закрывающим тегом HTML.Эти теги используются тоже один раз на странице.Тег BODY имеет несколько атрибутов, которые влияют на цвет текста и фона.

BGCOLOR="..."

цвет фона (по умолчанию серый)

BACKGROUND="..."

фоновый рисунок

TEXT="..."

цвет текста(черный)

LINK="..."

цвет ссылки (голубой)

VLINK="..."

цвет ссылки, которую посещали (фиолетовый)

ALINK="..."

активная ссылка(красный)

<B>...</B>

Выделенный текст

<I>...</I>

Наклонный. Не используйте слишком маленький размер шрифта, может выглядеть ужасно.

<U>...</U>

Подчеркнутый текст.Может иногда выглядеть как ссылка , внося путаницу.

<Hn>...</Hn>

Заглавие. Не используйте слишком часто. n значит размер от 1 до 6 .

**
**

Перенос:обрывает текущую линию и начинает новую с новой строки.

<P>

параграф: прерывает строку текста, пропускает строку и начинает новый параграф с повой строки.

<BLOCKQUOTE>...</BLOCKQUOTE>

Поля создают некоторое свободное пространство вокруг текста. Можно использовать несколько тегов, чтобы расширить поля.

<UL>...</UL>

UL: Сходен с тегом BLOCKQUOTE, только создает отступ слева. Может использоваться несколько раз, чтобы регулировать ширину поля.

<IMG>

Рисунок показывает рисунок на странице. Используйте только GIF или JPG, не бльше, чем 25 kB. Много атрибутов, только SRC обязателен.

SRC="..."

Source (источник) , указывает на файл рисунка. Указывайте правильное имя файла и путь к файлу. Файл также может быть размещен на другом сервере.

BORDER="n"

Border (границы) : Линия вокруг рисунка. Число указывает ширину границы в пикселях. При значение равном 0 , границы не видно.

WIDTH="n"

Ширина рисунка в пикселях. Позволяет странице загружаться быстрее.

HEIGHT="n"

Высота рисунка в пикселях. Позволяет странице загружаться быстрее.

ALT="..."

Альтернативный текст. Появляется если рисунок не загружен или в случае когда указатель мыши наведен на картинку.

<A>...</A>

Anchor(Якорь): (гипертекстовая) ссылка, наиболее важный тег, потому что соеденяет разные документы друг с другом.Текст между этими тегами отоброжается подчеркнутым, тем самым показывая что это ссылка.Тег имеет один обязательный атрибут, содержащий веб адрес ссылки.

HREF="..."

Гипертекстовая ссылка, веб адрес страницы, на которую делается ссылка. Указывайте полный адрес, если делаете ссылку на другой сервер.Включает в себя "http://" префикс и имя сервера.Иначе броузер будет искать этот документ на своем сервере, что закончится документом об ошибке (404 not found error). Можно также использовать почтовые адреса ('mailto:' префикс); и комбинированные с тегом IMG.

Итак ваша страница готова и вы хотите, чтобы ее увидел свет. Вы можете разместить ее у своего провайдера.

Место у провайдера

Иногда вы не можете разместить ее у своего провайдера. В этом случае есть простое решение. Есть много провайдеров предлагающих место для домашних страниц бесплатно. Я собрал некоторые из них на link page. странице с ссылками. Сервис на этих серверах сильно отличается друг от друга. Иногда позволено разместить только небольшую страничку, а в некоторых местах вам предоставляется 25 MB или более. Способы, с помощью которых вы можете загружать страницы на сервер провайдера , существуют совершенно разные. Я опишу самые расптространенные.

В дейсвительности для домашней страницы не нужно много места. К примеру, этот сайт размером всего 2 мегабайта. Только если вы собираетесь использовать много рисунков или звуковых файлов, тогда вам понадобится много места. Хороший фотограф легко может заполнить 150 MB.

FTP программы

Ваша страница будет размещена на веб сервере. Для этого вам понадобяться некоторые программы. Для этого можно использовать Netscape Navigator (Файл | Редактировать страницу | Публикация) , но в этом случае у вас не будет достаточно свободы. Лучший способ, это использовать FTP программы. FTP переводится как Протокол Передачи Файлов. FTP программы позволяют удалять и размещать программы в вашей дериктории

на сервере.

FTP программы имеют два окна. Левое показывает ваш компьютер, правое - веб сервер. Выглядит как Norton Commander или Windows Commander. Вы видите, что просходит на сервере и на вашем компьютере . **WS_FTP** наиболее часто используемая программа. Это программа бесплатна и вы можете найти ее на странице со списком программ.

Соеденяемся с провайдером

После того, как вы запустили WS_FTP, появляется окно логина. Если это не так, нажмите кнопку Connect. Впишите имя вашего FTP сервера. Часто это просто имя сервера, где "www." заменено на "ftp.".Потом нужно ввести ваш логин . Отметьте галочкой "Save password(Сохранить пароль)". Во вкладке StartUp вы прописываете путь к вашим страницам на сервере и вашем компьютере. Потом нажмите "Ok" и WS_FTP соеденит вас с провайдером.

Если все пройдет нормально, внизу окна появится сообщение "connected". Кнопка Connect изменится на 'close', нажав которую вы рассоеденитесь с вашим провайдером. С левой стороны вы увидите дерикторию с вашей домашней страницей, а с правой вашу дерикторию на сервере. Иногда в правом окне вам нужно будет выбрать другую дерикторию типа "html", "public_html" или "www"

Копирование файлов

Двойное нажатие на файл приводит к копированию в противоположное окно. Можно выделить все файлы и нажать правую стрелку.WS_FTP сообщит, когда все будет сделано.

Посмотрите на имена ваших файлов. Если вы видите, что какие-то имена содержат буквы верхнего регистра, переименуйте их. Иначе, могут быть проблеммы. Главная страница должна иметь следующие имена: index.htm, index.hmtl или default.html. Большинство веб серверов настроенны открывать этот файл автоматически.

Проверяем

Потом, запускайте ваш браузере, введите адрес вашей страницы в поле 'Location:' и нажмите Enter. Вы должны увидеть вашу страницу. Для моей страницы, например, должен быть введен следующий адрес: http://www.weballey.net/. Адрес должен быть написан правильно.

Если все было сделано правильно, вы должны увидеть вашу страницу в браузере. Если что-то не так, проверьте все. Ввели ли вы правильный логин и пароль? Как насчет регистра букв в названии файлов ? Если все правильно, но проблеммы по прежнему остаются, пошлите письмо вашему системному администратору.

Если все работает, не думайте, что это все. Проверьте и перепроверьте, все что вы разместили в интернете. Проверьте каждый рисунок, каждую ссылку. Если что-то не так , обязательно исправьте это. Если ошибок слишком много,

подумайте, может стоит убрать вашу страницу на время, до тех пор, пока не исправите все. Ничто так не достает, как ссылки, которые не работают.

Ну вот, ваша новая страница в интернете. Теперь вам хотелось бы привлечь как можно больше посетителей к вашей странице. Ничто так не растраивает, как слои пыли на счетчике посещаемости вашей страницы. Чтобы этого избежать, можно кое-что предпринять.

Ссылки

Как пользователи находят вашу страницу. Правильно: с помощью гиперссылок. Ссылки, это как клей, который соеденяет страницы интернета друг с другом. Поэтому единственный путь, чтобы кто-то нашел вашу страницу, это, чтобы кто-то разместил ссылку на вашу страницу. И чем больше таких ссылок, тем лучше. Есть несколько способов достичь этого.

- поисковые сервера
- обмен банерами
- ссылки на сходные страницы
- гостевые книги
- нелегальная деятельность

Поисковые сервера

Большинство находят их путь в интернете с помощью поисковых серверов. Несмотря на их неуклюжесть, неточность, на даныый момент это лучший способ найте что-то. Вы вводите несколько слов, и после этого исследуете ссылки. Большинство страниц будет не по теме или просто скучными. Но в конце концов, вы найдете страницы, где захотите остаться подольше и изучить их.

Как попасть в список поисковых серверов. Идете на их домашнюю страницу и найдите там 'Add URL' ссылку или кнопку. Нажав на эту кнопку, вы попадете на страницу, где нужно будет заполнить форму в которой вы можете указать все детали, касательно вашей страницы. Подтвердив эту информацию, вы попадете в их базу данных, где другие люди смогут найти вас.

Если бы были только один поисковый сервер, все было бы довольно просто. Но их буквально сотни. И заполнение форм на каждом из них заняло бы слишком много времени. Поэтому умные люди уже подумали об этом. Идея заключается в том, что вы заполняете форму только один раз, которая рассылается на множество серверов.

Вы можете предложить вашу страницу основным поисковым машинам прямо на этом сайте. Посмотрите на мою веб рекламную страницу. Есть также много сайтов, где вы можете сделать тоже самое. Сервис типа AddURL, Submit-it, Promote-it предлагается бесплатно для ограниченного числа поисковых серверов. Для большего числа поисковых машин вам придется платить или вам придется потрудится самому. Посетите мою

страницу списка утилит.

Другое решение это использованеи спциализированного програмного обеспечения. Демо версия , позволяет подтвердить вашу страницу на 10 или 20 сайтах. Если заплатить за эту программу, это позволит разместить информацию о вашей странице на несколько сотнях серверов. На странице списка программ вы можете найти некоторые из них .

Обмен банерами

Другую способ это использование банеров (рекламных символов). Это реклама, показываемая на вашей странице. В обмен, ваш собственный банер, будет показан на других сайтах. Наибольшего эффекта можно достичь, разместив банеры на каждой странице. Частные банеры иногда смотрятся ужасно, некоторые замедляют загрузку страницы, но они приносят определенный эффект.

Создавать банеры не очень просто. Для тех, кто не хочет тратить на это время, есть сайты, где можно найти целые библиотеки банеров. Вам только понадобиться добавить ваш текст. Существуют даже онлайн редакторы, для создания банеров. На странице списка утилит есть некоторые примеры.

Ссылки на родственные сайты

Бороздя просторы интернет, вы можете встретить сайты, сходные по теме или освещающие спецефическую тему вашего сайта. Вы можете послать создателю этого сайта письмо, с предложением сделать ссылку на его сайт в обмен на взаимною ссылку на ваш сайт. Я делал это много раз, и иногда это приводило к повышению трафика. Однако, сайты со списком ссылок, часто ведут к другом списку ссылок. Многие люди знают об этом и вероятно никогда не воспользуются ссылками на таком сайте.

Заполнение гостевых книг

Я никогда ни делал этого, но некоторые люди посоветовали мне использовать гостевую книгу. Причем в двух направлениях: разместив гостевую книгу на вашем сайте и заполняя гостевые книги на других сайтах. Некоторые повидимому читают их и после этого посещают вашу страницу. Если вам нравится эта идея, посетите эту страницу , разместите гостевую книгу и веселитесь.

Нелегальная активность

Может быть один из наиболее эффективных способов привлеч посетителей на ваш сайт, это предоставить нелегальную информацию. Вещи, типа порнографии, нелегальное програмное обеспечение, крэки (программы, которые ломают ограничение для использования некоторых программ), серийные номера, MP3 музакальные файлы и т.д..

Но только веселье не продлиться очень долго. Ваш провайдер вышвырнет вас через несколько дней, после того, как они заметят вашу страницу. Даже если информация не совсем нелегальная, они попытаются сделать это, чтобы

избежать большого трафика, который может привести к перегрузке сервера и сбою в его работе. Поэтому, мой совет, не делать таких страниц.

Ну вот вы и дошли до конца этого курса. Вы ознакомились с основами HTML. Я думаю это было не слишком трудно. Если вы запомнили, что было написано, то вы сможете создать простую страницу. Вы можете разместить вашу страницу в интернете. В основном вам не нужно знать боьше, чтобы создать веб сайт.

Тестирование

Один последний совет: тестируйте, проверяйте все, что вы делаете. Убедитесь, что ваш HTML код абсолютно безупречен. Убедитесь, что все рисунки присутствуют. Проверяйте ваши страницы в разных браузерах. Если страница смотрится хорошо в одном браузере, это не значит, что она будет выглядеть также хорошо в другом. В первый раз **никогда** не бывает без ошибок. Не размещайте наполовину законченные сайты. Это только раздражает посетителей.

Сделайте ссылку на этот сайт

Если вам понравился этот сайт, можете порекомендовать его друзьям. Это можно также сделать с помощью рекомендательной страницы. Также вы можете сделать ссылку на этот сайт или добавить баннер или кнопку, доступные на этой странице.

А что дальше

HTML намного сложнее, чем было описано выше. Я хотел бы дать вам совет. Сделайте страницу, используя то, чему вы научились здесь. Практикуйтесь на том, что вы узнали здесь до тех пор, пока вы не будете знать все на зубок. И только потом вы можете углубляться в HTML дальше.

Если вы знаете простые основы HTML, попробуйте мой курс по использованию тега TABLE. Почему таблицы, вы можете спросить. Таблицы, это наиболее важный инструмент по разметке веб страниц. Почти все профессиональные сайты используют их. Достаточная причина попробовать самому.

PHP. Урок 1. Что такое PHP?

PHP расшифровывается в настоящее время как PHP Hypertext Preprocessor (да, да, именно так - в расшифровке снова присутствует слово PHP!). Раньше же PHP расшифровывался как Personal Home Pages. Но в результате развития языка решили, что "персональные страницы" в названии не соответствуют языку, и его переименовали.

PHP - это язык для написания скриптов для WEB-сервера. Здесь важно то, что скрипты эти будут выполнятся именно на сервере, т. е. посетитель WEB-страницы никакого кода на PHP не увидит. Вместо вставок с кодом на PHP

посетитель увидит обычный код HTML (или DHTML, или HTML с вкраплениями какого-нибудь скрипового языка). В этом, кстати, одно из отличий PHP от других скриптовых языков - например, код JavaScript может передаваться прямо с HTML-кодом браузеру посетителя WEB-странички, и при желании этот код можно увидеть.

Файлы с скриптами на PHP имеют расширение *.php3 (для третьей версии php) или просто *.php (для четвертой версии). При этом код PHP встраивается с помощью специальных тегов в HTML-страничку, и как такую страничку интерпретировать (HTML с вкраплениями PHP или наоборот) - дело вкуса.

Основной конкурент PHP - это ASP (Active Server Pages). Но у ASP есть существенный недостаток - работают они, как правило, только на серверах IIS от Microsoft, а последних в Интернете примерно раза в три меньше, чем серверов Apache. PHP же может устанавливаться как на Apache под различные версии UNIX'a, так и под Apache под Windows, так и под IIS под WIndows. И, так как у большинства в качестве рабочей ОС стоит все-таки Windows, то менять ОС не нужно - PHP просто устанавливается и под эту операционную систему. Но этот процесс мы рассмотрим на следующем занятии.

PHP - продукт бесплатный. Так что совесть ваша будет чиста ;).

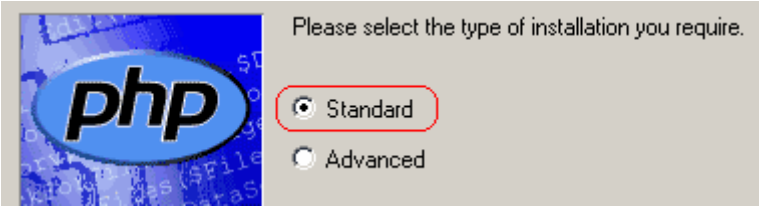
PHP. Урок 2. Установка PHP

В данном уроке мы рассмотрим процесс установки PHP на компьютер с Windows в качестве операционной системы. Предполагается, что на компьютере уже установлен WEB-сервер Apache.

Вопрос первый. Где взять PHP? Так как это продукт бесплатный, то его можно легально скачать из Интернета. По адресу http://www.php.net находятся все дистрибутивы. На момент написания этих строк последняя версия была 4.2.1. Но вы должны скачать ту версию, которую поддерживает ваш хостинг (если вы не изучаете PHP просто для собственного удовольствия). Сейчас почти на каждом хостинге (на платном, разумеется) поддерживают версии 3 и 4. Лучше скачать версию 4 - она полностью работоспособна, новейшие же версии могут иметь некоторые недоработки. В наших уроках речь будет идти о версии 4 или выше.

Так как мы работаем под Windows, то находим дистрибутив под эту ОС. Нужный нам файл будет называться приблизительно php-4.2.1-installer.exe (завист от номера версии). Он относительно небольшой (меньше мегабайта), так что проблем со скачиванием быть не должно.

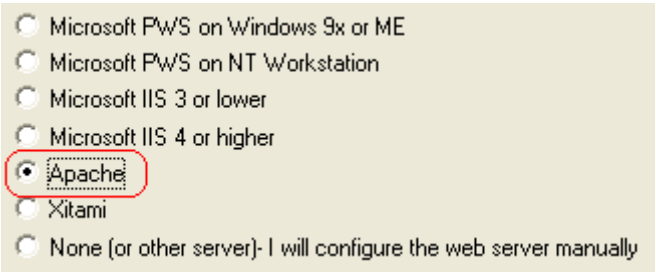
После скачивания запускаем скаченный exe-шник. Нажимая на кнопку Next, добираемся до вопроса о типе инсталляции - Standard или Advanced. Выбираем Standard - установки по умолчанию нас вполне удовлетворяют:



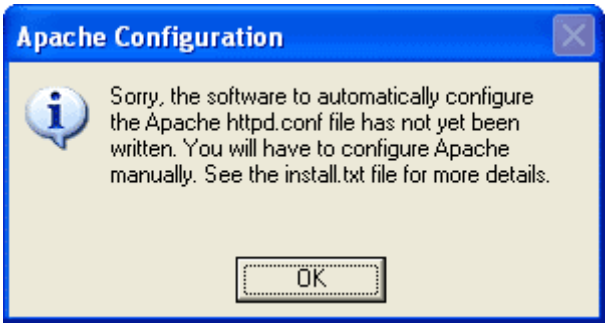
На очередном шаге мастера мы должны определить папку для установки. По умолчанию предлагается папка C:\php, которую, по свей видимости, логичнее заменить на C:\Program Files\php (Чем меньше в корневом каталоге папок, тем лучше). В дальнейшем мы будем предполагать, что PHP установлен в папку C:\Program Files\php. Нажимаем на Next.

На следующем шаге мастера вы должны определить параметры для электронной почты. Если вы оставите предложенные параметры без изменения, то функции PHP по работе с почтой будут недоступны. Можете задать и действительные значения для сервера SMTP (берется из настроек вашего почтового клиента) и для вашего e-mail'a.

Переходим к следующему шагу мастера, на котором вы должны определить для какого WEB-сервера мы устанавливаем PHP. Выбираем, естественно, Apache:



После того, как программа установки скопирует все нужные файлы, появится окно, информирующее нас о том, что необходимо вручную внести изменения в файл httpd.conf:



Это файл конфигурации сервера Apache. Найдите его. Он должен находится в папке conf вашего WEB-сервера (скорей всего это будет путь C:\Program Files\Apache Group\Apache\conf). Внесите в него следующие изменения.

Раскомментируйте сторочку

```
...
#AddType application/x-httpd-php .php
...
```

для чего просто уберите знак # перед ней. Таким образома эта строка превратится в

```
AddType application/x-httpd-php .php
```

Раскомментирвав эту строку, мы говорим WEB-серверу, то файлы с расширением *.php надо обрабатывать особым образом. Теперь осталось указать WEB-серверу, кто будет обрабатывать файлы с расширением php. Для этого добавьте после только что измененной строки следующие 2 строчки:

```
...
AddType application/x-httpd-php php
ScriptAlias /_php4/ "c:/Program Files/php/"
Action application/x-httpd-php "/_php4/php.exe"
...
```

В первой из этих строк мы задаем псевдоним _php4 для расположения интерпретатора php (т. е. мы указываем имя той папки, в которую мы php установили). Обратите внимание на все слеши, в частности на слеш в конце строки. В последней добавленной строке мы указываем, кто будет обрабатывать файлы с расширением php, при этом мы используем определенный нами псевдоним.

Сохранита файл httpd.conf и закройте его.

Перезапустите WEB-сервер Apache (т. е. остановите его и запустите опять).

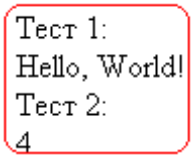
С установкой PHP все!

PHP. Урок 3. Первая программа на PHP

Назначение первой программы - просто проверить правильность установки PHP. Для чего создайте в папке для html-файлов вашего сервера (она, скорей всего, будет называться htdocs) файл test.php следующего содержания:

```
<body>
Тест 1:<br>
<?
echo "Hello, World!<br>"
?>
Тест 2:<br>
<?=2+2?>
</body>
```

Как вы видите, код php располагается прямо на WEB-странице, прямо посреди html-тегов. Как правило, код php расположен между тегами <? и ?>, хотя возможны и другие варианты. Все, что расположено между такими специальными тегами, обрабатывает наш интерпретатор php (не зря же мы его устанавливали). Результат будет, как несложно догадаться, таким:



Если вы получили именно такой результат, то с установкой PHP все в порядке.

PHP. Урок 4. Переменные

Как и в любом другом нормальном языке программирования, в PHP есть понятие переменной. С технической точки зрения переменная - это просто именованная область в памяти (адресном пространстве). Если мы заводим переменную, то мы можем обращаться к содержимому соответствующих ячеек памяти по имени переменной.

Переменные в PHP бывают следующих типов: целые числа, вещественные (дробные) числа, строки. Есть еще массивы и объекты, которые тоже можно рассматривать как особый тип переменных, но на этом занятии мы их рассматривать не будет, а отложим это на будущие уроки.

Для того, чтобы определить в вашей программе переменную, мы должны написать что-то вроде

```
...
$var=31;
...
```

Тем самым мы заводим переменную $var целого типа. Обратите внимание на целый ряд моментов. Во-первых, мы сразу присваиваем нашей переменной некоторое значение. Автоматически никакие значения типа нуля

присваиваться не будут. Во-вторых, мы не задаем явно типа переменной. Тип определяется тем, что мы в эту переменную пишем. В данном случае мы пишем туда целое число и именно по этому тип нашей переменной $var - целый. Обратите внимание также на то, что имя любой переменной должно начинаться со знака доллара ($). Сначала это кажеться странным и слегка раздражает - в "нормальных" языках программирования такого нет. Но потом привыкаешь. Знак доллара надо использовать почти во всех случаях - например, если мы показываем значение переменной на HTML-страничке, то мы пишем что-то вроде:

```
...
echo $var;
...
```

Переменная может менять свой тип в процессе работы программы. Вот пример:

```
<?
$var=31;
echo $var;
$var="Hi";
echo $var;
$var=3.14;
echo $var;
?>
```

В данном примере переменная $var сначала имеет целый тип, затем - строкомый, и потом - вещественный. Значения всех трех типов будут выведены на экран.

Переменные строкового типа можно заключать как в двойные, так и в одинарные кавычки. Разница между ними в том, что одинарные кавычки интерпретатор воспринимает буквально, т. е. никаких замен и подстановок не производится. При использовании же двойных кавычек именая переменных заменяются на значения переменных. Вот пример:

```
<?
$var=22;
$str="Number                    $var
";
echo $str;
$str='Number                    $var
';
echo $str;
?>
```

А вот результат выполнения указанного фрагмента:

```
Number 22
Number $var
```

Как вы видите, в двойных кавычках переменная $var заменилась на 22, а в одинарных - нет.

В строках PHP, как и в других C-подобных языках, можно использовать специальные символы - \n (новая строка), \r (переход к началу строки), \t (табуляция), \\ (печать символа \), \" (символ двойной кавычки) и некоторые другие.

PHP. Урок 5. Константы

Константы - это то, что в вашей программе меняться не должно. Константы мы заводим с помощью define (обратите внимание, что в имени константы не надо писать в начале знак доллара). В следующем примере мы заводим константу и выводим ее на HTML-страницу:

```
define ("pi", "3.1415926");
echo pi;
```

В PHP существует целый ряд предопределенных констант. Вот самые распространенные из них:

- __FILE__ - имя файла, в котором расположен скрипт PHP.
- __LINE__ - номер текущей выполняемой строки в файле сценария.
- PHP_OS - операционная система, на которой выплняется сценарий.
- PHP_VERSION - номер версии интерпретатора PHP.
- TRUE - истина.
- FALSE - ложь.

Применение этих констант может быть самое разное - например, константы PHP_OS и PHP_VERSION могут быть использованы в том случае, если ваш скрипт использует некоторые возможности конкретной ОС или конкретной версии PHP.

PHP. Урок 6. Логические операторы

На этом уроке мы рассмотрим логические операторы языка PHP. Логические операторы применимы к логическому типу, который является подтипом целого типа. Переменные логического типа могут принимать два значения - TRUE (истина) и FALSE (ложь). При этом, как и в других языках программирования, ноль интерпретируется как ложь, а не ноль - как истина.

А вот и сами логические операторы:

Оператор	Пример	Описание
&&	c$ = $a && $b;	Возвращет истину, только если оба операнда равны истине
And	d$ = $a And $b And c$;	Тоже самое, что и &&
||	c$=(k==3)||(k==5);	Возвращет ложь, только если оба операнда равны лжи
Or	c$ = $a Or $b;	Тоже самое, что и ||
Xor	c$ = $a Xor $b;	Исключающее или. Возвращает истину только тогда, когда ровно один из операндов (левый или правый) равен истине
!	$s=!$d;	Отрицание. Истину превращает в ложь, а ложь - в истину
==	if($n==10)...;	Логическое равно. Возвращет истину, если левая часть равна правой
!=	if($n!=10)...;	Логическое не равно. Возвращет истину только если левая часть не равна правой

Обратите внимание, что логическое равно - это два знака равно (==), а присваивание - только один (=). Не путайте эти две вещи.

PHP. Урок 7. Массивы

Сразу приведем пример работы с одномерным массивом:

```
<?
$a[0]="Igor";
$a[1]="Osco";
$a[2]="Otto";
echo "$a[1]";
?>
```

Указанный фрагмент покажет на экране, естественно, слово Osco. Обратите внимание на то, что у нас нет объявления массива как такового - как и в случае с обычными переменными, мы создаем переменную в момент присваивания ей некоторого значения. Отличие от обычных переменных (не массивов) в том, что после имени переменной мы пишем индекс (номер) элемента в нашем массиве.

При добавлении в массив новых элементов индекс нового элемента можно в принципе не писать - новому элементу автоматически присвоится очередной номер:

```
<?
$a[]="Igor";
$a[]="Osco";
$a[]="Otto";
?>
```

В указанном фрагменте значение "Igor" запишется в $a[0], "Osco" запишется в $a[1] и "Otto" запишется в $a[2]. Как видно из приведенного примера, если мы не будем указывать значения для индекса, то для него будет браться предыдущий оспользованный индекс, увеличенный на 1. Так как у нас сначала вообще ни одного индекса не было, то первый получает значение 0, следующий - увеличивается на 1 и т. д. Вот еще поясняющий пример:

```
<?
$a[0]="Igor";
$a[2222]="Osco";
$a[]="Otto";
echo "$a[2223]";
?>
```

Здесь элемент со значением "Otto" получает номер 2223 (2222+1). Указанный форагмент выведет в браузер слово "Otto".

Для массивов в качестве индексов вместо чисел могут использоваться и строки. Часто это оказывается удобным. Такие массивы называются ассоциативными. Вот пример для дней недели:

```
<?
$a["mon"]=1;
$a["thu"]=2;
$a["wed"]=3;
$a["thu"]=4;
$a["fri"]=5;
$a["sat"]=6;
$a["sun"]=7;
echo $a["fri"];
?>
```

Приведенный фрагмент покажет в браузере 6 (т. е. значение для индекса "fri").

Для создания массивов можно применять функцию array. Вот два примера (для массива с числовым индексом м для ассоциативного массива):

```
<?
$b=array("Igor", "Osco", "Otto");
?>
<?
$c=array("mon"=>1, "thu"=>2, "wed"=>3);
?>
```

И, наконец, поговорим о многомерных массивах. Здесь все аналогично массивам одномерным. Вот так, например, мы можем задать несколько элементов двумерного массива:

```
$n[1][1]=12;
$n[1][2]=-4;
$n[2][1]=7;
$n[2][2]=0;
```

Для задания многомерных массивов мы можем воспользоваться и функцией array. Вот пример:

```
$k=array(array(1, -41), array(3, 10));
```

Здесь создается двумерный массив с элементами от $k[0][0] до $k[1][1] соответственно. Из последнего примера особенно ясно видно, что многомерные массивы интерпретируются как массивы массивов.

PHP. Урок 8. Операторы ветвления

Основной оператор ветвления - это if. С ним все более-менее ясно - никаких принципиальных отличий от других языков нет. Вот сразу пример:

```
if ($k==12){
     echo "Данные равны 12<br>";
}
else{
     echo "Данные не равны 12<br>";
}
```

Если k равно 12, то покажеться одна надпись, а если не равно, то другая. Веточку else можно не писать, если в ней нет необходимости. Условие (проверку), как и в других С-подобных языках, надо писать в круглых скобках. Слово else, естественно, не пишем. Еще раз, кстати, обратите внимание, что для логического равно (т. е. для ответа на вопрос, равно ли что-то чему-то) используется *два* знака равно "==".

Если идет несколько проверок подряд, то можно использовать конструкцию с elseif:

```
if ($k>0){
     echo "Число положительное<br>";
}
elseif($k<0){
     echo "Число отрицательное<br>";
}
else{
     echo "Число равно нулю<br>";
}
```

Если количество возможных ветвлений велико (программа должна идти или сюда, или сюда, или туда...) и возможные значения для проверяемого выражения представляют из себя конкретные значения (а не условия типа меньше или больше), то лучше использовать switch. Вот пример:

```
switch($note){
case 1:
case 2:
     echo "Двоечник<br>";
     break;
case 3:
     echo "Троечник<br>";
     break;
case 4:
     echo "Четверочник<br>";
     break;
case 5:
     echo "Отличник<br>";
     break;
default:
     echo "Ошибка!<br>";
}
```

Обратите внимание, что в конце группы операторов, как правило, надо ставить break, иначе операторы, написанные в последующем case, тоже будут выполняться (у нас это видно для значений 1 и 2 - в обоих этих случаях в браузер выведется "Двоечник"). Если ни одна из проверок не увенчается успехом, то выполнится часть break.

PHP. Урок 9. Операторы циклов

Из циклов самый известный - это for. Он выполняется определеное число раз (как правило, известное заранее). Вот пример:

```
<?
$fact=1;
for($i=1; $i<=10; $i++){
     $fact=$fact*$i;
}
echo "$fact<br>";
?>
```

Данный пример считает произведение чисел от 1 до 10 (т. н. факториал) и выводит результат в браузер.

По задаче часто нужны циклы, про которые заранее не известно, сколько раз они дожны выполнится. В этом случае используем циклы while. У них две разновидности - do-while и while. Сначала рассмотрим цикл do-while. Вот пример его использования для той же задачи нахождения факториала числа:

```
$j=1;
$fact=1;
do{
     $fact=$fact*$j;
     $j++;
}while($j<=5);
echo "$fact<br>";
```

А вот и пример испольования оператора while:

```
$b=array(6, -5, 22);
$i=0;
while($b[$i]>=0){
      $i++;
}
echo "$i-й элемент (равный $b[$i]) - отрицательный
";
```

В этом примере мы ищем первое отрицетельное число в массиве и выводим само число и его номер (начиная нумерацию с нуля).

PHP. Урок 10. Сокращенные операторы

В PHP, как и в других родственных языках, применяются следующие сокращенные операторы:

Оператор	Пример	Эквивалентное выражение
+=	$b+=3;	$b=$b+3;

-=	$a-=4;	$a=$a-4;
*=	$n*=2;	$n=$n*2;
/=	$k/=$z;	$k=$k/$z;
%=	$x%=$y;	$x=$x%$y;

Сокращенные операторы представляют из себя единые знаки и разделять их в тексте программы пробелами нельзя.

К сокращенным операторам относятся также операторы ++ и --. Они соответственно увеличивают или уменьшают свой аргумент на единицу. При этом для каждого из них возможны два варианта написания:

```
$k++;
++$k;
```

Если оператор ++ (или --) является отдельным оператором, то разницы между этими двумя способами нет. Если же этот оператор является частью другого выражения, то возможны нюансы:

```
<?php
$n=10;
echo $n++;
echo "<br>";
echo $n;
?>
```

Этот фрагмент выведет в окно браузера два числа - 10 и 11. Обратите внимание, что в строчке

```
...
echo $n++;
...
```

у нас сначала выведется старое значение $n (равное 10) и только потом произойдет увеличение $n на единицу.

Вот еще пример:

```
<?php
$n=10;
$m=$n++;
echo "n=$n, m=$m";
?>
```

Здесь в окно браузера выведется "n=11, m=10". Если мы запишем так:

```
<?php
$n=10;
$m=++$n;
echo "n=$n, m=$m";
?>
```

то результатом будет "n=11, m=11".

PHP. Урок 11. Передаем данные из формы

На этом занятии мы посмотрим, как данные из браузера пользователя передаются в программу на PHP (т. е. на сервер). Программа наша будет предасавлять из себя некоторое подобие калькулятора - на форме в своем браузере пользователь введет два числа, и выберет с помощью группы радиокнопок одно из арифметических действий - сложение, вычитание и т. д. После нажатия на кнопку "Подсчитать" (Submit), WEB-сервер вернет HTML-страницу с результатом вычислений.

Сначала создадим форму. Для этого создайте на вашем локальном WEB-сервере HTML-файл calc.htm следующего содержания:

```
<html>
<head>
      <title>Калькулятор</title>
</head>
<body>
<form method="post" action="calc.php">
<p>n1=<input type="Text" name="n1"></p>
<p>n2=<input type="Text" name="n2"></p>
<p><input type="Radio" name="calc" value="add"> +</p>
<p><input type="Radio" name="calc" value="sub"> -</p>
<p><input type="Radio" name="calc" value="mul"> *</p>
<p><input type="Radio" name="calc" value="div"> /</p>
<p><input type="Submit" value="Результат"</p>
</form>
</body>
</html>
```

Выглядеть ваша форма будет приблизительно так:

---

n1=

n2=

+

```
-

*

/
```

Особо обсуждать эту форму мы не будем - подробности можно посмотреть в разделе по HTML (урок 23 и далее).

Теперь займемся файлом calc.php. Вот его текст:

```
<?
if($calc=="add"){
    $res=$n1+$n2;
}
if($calc=="sub"){
    $res=$n1-$n2;
}
if($calc=="mul"){
    $res=$n1*$n2;
}
if($calc=="div"){
    $res=$n1/$n2;
}
?>
<html>
<head>
  <title>Калькулятор</title>
</head>
<body>

<p>Ответ: <? echo "$res"; ?></p>

</body>
</html>
```

Разместите его в той же папке на локальном WEB-сервере, где и файл calc.htm.

Небольшое обсуждение кода этой php-странички. У нас есть 4 однотипных if:

```
...
if($calc=="add"){
```

```
    $res=$n1+$n2;
}
...
```

в которых мы проверяем значение переменной $calc. Откуда эта переменная берется? Обратите внимание на строчки

```
...
<p><input type="Radio" name="calc" value="add"> +</p>
<p><input type="Radio" name="calc" value="sub"> -</p>
<p><input type="Radio" name="calc" value="mul"> *</p>
<p><input type="Radio" name="calc" value="div"> /</p>
...
```

в нашем файле calc.htm. А именно, все наши радиокнопки называются calc, и в зависимости от того, какую из них выберет посетитель HTML-странички, значение переменной calc будет тем или другим ("add","sub", "mul" или "div"). И в коде нашего php-файла calc.php мы и анализируем значение этой переменной и затем в переменную $res записываем тот или иной результат действия с переменными $n1 и $n2, которые тоже берутся с нашей странички calc.htm (это имена двух полей для ввода чисел). Далее в строчке

```
...
<p>Ответ: <? echo "$res"; ?></p>
...
```

мы просто выводим значение переменной $res в браузер.

Вот, собственно, и все. Теперь если вы в адресной строке вашего браузера наберете "http://localhost/calc.htm" (если вы разместили эти две странички прямо в корне WEB-сервера), заполните n1 и n2 и нажмете на конопку "Результат", то файл calc.php подсчитает результат и выдаст его вам.

PHP. Урок 12. Функции в PHP

Что такое функция? Очень приближенно, это некоторый кусок кода, который имеет имя и который возвращает некоторое значение (результат). Раз он имеет имя, то мы его можем использовать многократно - просто вызывая по имени. То, что функция возвращает результат, тоже чаще всего используется. Функции делятся на те, которые вы сами написали, и на встроенные функции PHP. Встроенных функций очень много, и в реальности вы будете использовать небольшое их количество. Такие функции вместе с примерами употребления мы постепенно рассмотрим на наших уроках. Сейчас же мы обсудим пользовательские функции.

Итак, давайте в качестве примера сами напишем функцию, которая просто подсчитает произведение всех чисел от 1 до n (так называемый факториал, обозначается n!). Число n передастся в нашу функцию в качестве параметра. Вот текст:

```
<?php
function fact($n){
    $res=1;
    for($i=1;$i<=$n;$i++){
        $res*=$i;
    }
    return $res;
}
//Используем функцию
echo fact(5);
?>
```

Указанный фрагмент выведет в окно браузера 120 (=1*2*3*4*5). Как вы видите, объявление функции начинается с слова function, за которым следуют параметры в круглых скобках. Если параметров несколько, то они разделяются запятыми. Если параметров нет, то круглые скобки все равно обязательно надо ставить (писать в них в этом случае, естественно, ничего не надо). Возвращаемое функцией значение пишется после слова return. На слове return происходит выход из тела функции и следующие далее операторы не выполняются.

Функция может вызывать внутри себя другие функции. В частности, функция может вызывать сама себя (это так называемые рекуррентные функции). Вот как можно переписать нашу функцию для нахождения факториала, так, чтобы она использовала сама себя:

```
function fact2($n){
    if($n!=1){
        return $n*fact2($n-1);
    }
    else{
        return 1;
    }
}
```

Как вы видите, здесь функция fact2 вызывает внутри саму себя. Используем мы тот факт, что n!=n*(n-1)!.

Функция может не возвращать никакого значения (это аналог процедур в некоторых языках программирования). Вот пример такой функции:

```
function f(){
    echo "Вызов функции f";
}
```

Вызывается такая функция обычным образом:

```
...
    f();
...
```

В таких не возвращающих значения функциях тоже можно писать return для выходя из функции. В таких случаях после return ставить возвращаемое значение смысла нет.

PHP. Урок 13. Функции с переменным числом параметров

Про некоторые функции заранее трудно сказать, сколько параметров у нее будет. Например, функция может подсчитывать сумму некоторого количества чисел, причем сколько чисел передается в такую функцию для суммирования - заранее неизвестно. Вот сейчас мы и посмотрим, что в таких случаях можно сделать. Давайте в качестве примера и напишем функцию для подсчета суммы своих параметров. Вот ее текст:

```
<?php
function f(){
    $sum=0;
    for($i=0; $i<func_num_args();$i++){
         $sum+=func_get_arg($i);
    }
    return $sum;
}
//Использование функции
echo f(2, 3, 1, 5);
?>
```

Этот фрагмент выдаст, разумеется, 11. Что мы в нашей функции используем? Во-первых, мы используем встроенную функцию PHP func_num_args(). Она возвращает количество переданных в функцию параметров. Во-вторых, мы используем встроенную функцию func_get_arg(...). Она возвращает значение параметра функции с номером, который передается в func_get_arg(...) в качестве аргумента. Так, func_get_arg(0) возвратит первый параметр, а func_get_arg(1) - второй. Наряду с этими двумя функциями существует еще функция func_get_args(), возвращающая список всех параметров.

PHP. Урок 14. Функции с параметрами по умолчанию

Очень часто в функциях с несколькими параметрами часть из них имеет стандартное значение. Тогда при объявлении функции следует использовать параметры по умолчанию. Когда мы такую функцию будем вызывать, то параметры по умолчанию можно не писать - вместо них подставятся некоторые заданные нами стандартные значения. В тех же редких случаях, когда в функцию надо передать нестандартное значение, соответствующие параметры у функции пишутся. Вот пример:

```
<?
function func($n, $town="New York"){
  echo $n;
  echo "<br>";
  echo $town;
  echo "<br>";
}
  $k=22;
  //Вызов функции с параметром по умолчанию
  func($k);
  //Вызов функции с заданным параметром
  func($k, "Los Angeles");
?>
```

Указанный фрагмент выведет четыре строки: 22, New York, 22 и Los Angeles.

Параметров по умолчанию у функции может быть несколько. Важно однако, что они должны идти подряд и до конца. Т. е. если некоторый параметр имеет значение по умолчанию, то и все параметры справа от него тоже должны иметь значение по умолчанию:

```
...
//Правильно
function func1($n, $town="New York", $type=2){
  ...
}
//Неправильно
function func2($n, $town="New York", $type){
  ...
}
...
```

PHP. Урок 15. Передаем параметры по ссылке

Параметры в функцию в PHP можно передавать по значению (по умолчанию так и делается) и по ссылке. Если вы передаете параметр по значению, то

создается копия переменной. Функция может внутри себя изменять такой параметр, но при выходе из функции значение переменной, переданной в качестве параметра, не изменится. Т. е. функция внутри себя имеет дело с локальной копией. Вот поясняющий это пример:

```
<?
function func($n){
  $n++;
  echo $n; //Напечатается 9
};
//Вызов функции
$k=8;
  func($k);
  echo $k; //Напечатается 8
?>
```

Иногда же нам необходимо, чтобы функция изменяла передаваемые в нее параметры. Т. е. мы хотим, чтобы после выполнения функции переданные в нее параметры получили другое значение. Делается этто так:

```
<?
function func(&$n){
  $n++;
  echo $n; //Напечатается 9
};
//Вызов функции
$k=8;
  func($k);
  echo $k; //Напечатается 9
?>
```

Обратите внимание на знак амперсанда (&) перед параметром функции:

```
...
function func(&$n){
...
```

Именно благодаря ему наша функция получает не копию переменной, а "оригинал".

Это был синтаксис в стиле C/C++. В PHP существует и другой синтаксис для этой цели:

```
<?
function func($n){
  $n++;
```

```
  echo $n; //Напечатается 9
};
//Вызов функции
$k=8;
  func(&$k);
  echo $k; //Напечатается 9
?>
```

Как видите, здесь амперсанд ставится при вызове функции, а не при объявлении, как в первом случае. Если ваша функция всегда должна изменять значение своего параметра то, разумеется, первый способ предпочтительнее (так как программисту ничего не надо будет помнить - просто вызываем функцию и все). Если же ваша функция иногда должна менять свой параметр, а иногда нет, то используйте второй способ.

PHP. Урок 16. Работаем со строками

На этом занятии мы с вами рассмотрим наиболее употребительные функции по работе со строками.

Сначала операция конкатенации строк. Под этим мудреным названием скрывается простое сложение нескольких строк. В отличие от других языков программирования, в PHP используется оператор "точка" (.). Вот пример:

```
$str1="Война";
$str2=" и ";
$str3="мир";
echo $str1.$str2.$str3;
```

Указанный фрагмент выведет в браузер "Война и мир".

Следующая распространенная задача - это вывести в браузер значения некоторых переменных вместе с некоторым сопровождающим текстом. Вот так это делается:

```
$x=23;
$y=20;
echo "x=$x, y=$y";
```

Приведенный фрагмент покажет в окне браузера строку "x=23, y=20" (без кавычек, разумеется). Обратите внимание, что мы используем тут двойные кавычки. Их использование заставляет интерпретатор PHP заменять переменные типа $x на сооветствующие значения. Если же мы напишем вместо двойных кавычек одинарные:

```
$x=23;
```

```
$y=20;
echo 'x=$x, y=$y';
```

то никакой замены производиться не будет, и в окно браузера выведется строка "x=$x, y=$y".

Наряду с echo можно использовать print:

```
$str1="Испания";
print "Солнечная $str1";
```

Правило здесь такое же как и для echo - в двойных кавычках имена переменных заменяются на соответствующие значения, а в одинарных замены не происходит.

Для превращения символа в его ascii-код используем функцию ord:

```
echo ord("A");
```

Этот фрагмент напечатае 65 (код символа "A").

Для обратного преобразования - получения символа из его ascii-кода - используем функцию chr:

```
echo chr(65); // Напечатается "A"
```

PHP. Урок 17. Продолжаем работать со строками

Продолжаем работать со строками.

Одно из типичных примений строк - это прием информации от посетителя WEB-странички. Так как посетители не отличаются особым вниманием, то довольно-таки часто вместе с собственно данными будет посылаться лишние пробелы. В PHP предусмотрено несколько функций по удалению таких пробелов. Рассмотрим их.

Функция trim(). Отсекает лишниие пробельые символы в начале и в конце строки. Под пробельными символами имеются в виду пробелы, табуляция, переходы на новую строку. Пример использования:

```
$str="   Ушку   ду Кашку \n  ";
$str=trim($str);
```

После выполнения указанного фрагмента в переменной $str откажется строка "Ушку ду Кашку". Обратите внимание, что удаляются только лишние начальные и конечные пробельные символы. Из середины строки ничего не удаляется.

Для удаления только начальных или только конечных символов используются функции ltrim() и chop(). Вот пример их применения:

```
$str1="   Испания  ";
$str1=ltrim($str1);
$str2="  Франция  ";
$str2=chop($str2);
```

После выполнения указанного фрагмента в переменной $str1 будет строка "Испания ", а в переменной $str2 - " Франция"

Вместо функции chop() можно использовать rtrim() - это одно и тоже.

Для нахождения длины строки используем функцию strlen():

```
$str="Испания";
$z=strlen($str);
echo $z;
```

Фрагмент выведет 7 (число символов в слове "Испания").

Для выделения части строки в PHP используется функция substr:

```
$str1="Шмидт Отто";
$str2=substr($str1, 0, 5); //В $str2$ - строка "Шмидт"
$str3=substr($str1, 6); //В $str3 - строка "Отто"
echo "$str2\n$str3";
```

Параметры у этой функции достаточно понятные: первый - строка, из которой выделяем нужные символы, второй - с какой позиции выделяем (нумерация начинается с нуля), и третий - сколько символов выделяем. Третий параметр можно не писать - тогда функция возвратит все символы до конца строки (начиная с символа, определяемого вторым параметром). В качестве второго параметра можно использовать отрицательное число - в этом случае символы берутся с конца строки. Вот пример:

```
$str1="Шмидт Отто";
$str2=substr($str1, -4);
echo $str2;
```

Приведенный фрагмент возвратит "Отто".

Для сравнения строк используем функцию strcmp(). Она возвращает число ноль, если строки равны, отрицательное число, если перрвая строка идет раньше второй в алфавитном порядке и положительное число, если первая строка идет дальше второй в алфавитном порядке. Пример использования:

```
$str1="aaa";
$str2="bbbbb";
$res = strcmp($str1, $str2);
echo $res;
```

При этом обратите внимание, что эта функция определяет именно алфавитный порядок следования строк. Так, указанный фрагмент возвратит -1, несмотря на то, что в первой строке 3 символа, а во второй - пять. Из особенностей этой функции отметьте еще, что она учитывает строчные и прописные буквы. Если же необходимо сравнить две строки без учета регистра, то используйте функцию strcasecmp().

Для нахождения фрагмента в строке используется функция strpos. Она возвращает номер позиции, с которого начинается искомый фрагмент. Пример использования:

```
$str1="Alexeev Igor";
$str2="Igor";
$res = strpos($str1, $str2);
echo $res;
```

Этот фрагмент возвратит 8.

И, наконец, функции по замене символов в данной строке. Сюда можно отнести такие функции как strtr и str_replace. Первая из них производит замены нескольких отдельных символов, вторая - подстрок. Вот примеры их использования:

```
$str1="abba";
$str2=strtr($str1, "abc", "xyz");
echo $str2;
```

Правило замены тут такое - a заменяется на x, b заменяется на y, c заменяется на z. В частности, можно во втором и третьем аргументах писать по одной букве.

```
$str1="Osco родился в ... году в .... Родители Osco
...";
$str2=str_replace("Osco", "Casco", $str1);
echo $str2;
```

Приведенный фрагмент возвратит строку "Casco родился в ... году в .... Родители Casco ..."

PHP. Урок 18. Функции по работе с датами и временем

Функций по работе с числами и датами в PHP есть достаточно много. Вот самые распространенные из них:

Функция time(). Возвращает значение типа int - количество секунд, прошедших с 1 января 1970 года. Именно с этого момента отсчитывается время в UNIX-системах. Кроме своего прямого использования, эта функция может применяться для получения различных случайных чисел.

Функция getdate(int). Возвращат ассоциативный массив, элементы которого содержат отдельные значения для секунд, минут, ..., месяца, года. В качестве параметра функция берет количество секунд с 1 января 1970 года (т. е. возвращаемое функцией time()). Вот пример использования этой функции:

```
$d = getdate(time());
echo $d["hours"]; //Выводит количество часов
echo ":";
echo $d["minutes"]; //Выводит количество минут
echo ":";
echo $d["seconds"]; //Выводит количество секунд
echo "<br>";
echo $d["mon"]; //Выводит номер месяца
echo "<br>";
echo $d["month"]; //Выводит название месяца
echo "<br>";
```

В окне браузера выведется время в формате часы:минуты:секунды, номер текущего месяца и название (английское) месяца. Кроме приведенных значений в качестве индекса возвращаемого ассоциативного массива могут выступать mday (число месяца), wday (день недели в виде числа), weekday (английское название дня недели), year (год), yday (номер дня в году).

Функция date(). Аналогична функции getdate(). Возвращет строку, которая может включать в себя значения для года, месяца, часов, минут и т. п., причем данные могут возвращаться в разных форматах. Формат возвращаемого значения определяется первым параметром функции. Второй (необязательный) параметр опередеяет момент времени (количество секунд с 1 января 1970 года). Если второй параметр не задан, то подразумевается текущее время. Вот парочка примеров использования этой функции:

```
echo date("l, F d, Y");
echo "<br>";
echo date("Сегодня d.m.Y");
echo "<br>";
```

В первом случае в браузер введется что-то вроде "Monday, August 19, 2002", во втором - "Сегодня 19.08.2002". Как вы видите, для первого параметра существуют форматирующие символы. Вот некоторые из них: y и Y - год (2 и 4 цифры соответственно), F - название месяца, m - номер месяца, d и j - номер дня месяца (с начальным нулем для однозначных чисел и без него), h и H - часы (в 12 и 24-часовом форматах), l - название дня недели.

PHP. Урок 19. Случайные числа

В PHP есть две группы функций по работе со случайными числами. Чисто внешне их можно отличить по префиксу mt_ у всех функций одной из групп.

Вот функции первой группы:

Функция rand. Возвращает целое число от нуля до значения RAND_MAX (которое чаще всего равно 32767). Может иметь два необязательных целых параметра - если они указаны, то генерируется случайное число от первого параметра до второго. Обратите внимание, что функция rand всегда возвращает одну и ту же последовательность случайных чисел (очень удобно для отладки ;)).

Функция srand. Задает последовательность случайных чисел, выдаваемую функцией rand. Имеет целый параметр - при разных значених этого параметра rand будет выдавать разные последовательности чисел. Функцию srand достаточно вызвать только один раз перед всеми вызовами функции rand. Пример использования:

```
srand(1288); //Инициализация генератора случайных чисел
for($i=0; $i<5;$i++){
  echo rand();
  echo "<br>";
}
```

Функция getrandmax(). Возвращает значение максимального случайного числа. Параметров не имеет.

Функции второй группы. Как уже было сказано, они имеют префикс mt_. Их названия и действия аналогичны функциям первой группы - mt_rand, mt_srand и mt_getrandmax. Основное отличие - что случайные числа берутся из более широкого диапазона: от 0 до 2147483647 (2 в 15-й степени минус 1). Пример использования:

```
//Инициализация генератора случайных чисел текущим временем
mt_srand(time());
for($i=0; $i<5;$i++){
```

```
  //Выдаем случайное число от 1000 до 2000
  echo mt_rand(1000, 2000);
  echo "<br>";
}
```

Данный фрагмент будет выводить в браузер случайные числа от 1000 до 2000. Выводимые числа будут каждый раз разные - обратите внимание, что в функции mt_srand в качестве параметра мы используем текущее время.

Кроме того, функции второй группы работают быстрее.

PHP. Урок 20. Функции по работе с датами и временем (продолжение)

На уроке 18 мы уже смотрели некоторые функции, работающие с датами и временем. На этом уроке мы еще рассмотрим некоторые функции из этого же семейства.

Функция **checkdate**. Предназначена для проверки того, существует ли указанная дата или нет. Пример использования:

```
...
if(checkdate(11, 31, 2002))
      echo "Дата существует<br>";
else
      echo "Дата не существует<br>";
...
```

Первый параметр тут - это месяц, второй - число, третий - год. Указанный фрагмент выведет "Дата не существует", так как в ноябре 30, а не 31 день. Функцию checkdate удобно использовать для автоматического формирования различных календарей.

Функция **mktime**. По заданному времени в часах, минутах, секундах, месяцах, днях, годах (именно в таком порядке) возвращает целое число - количество секунд, прощедших с 1 января 1970 года. Часто используется вместе с функцией date для формирования строки, содержащей дату:

```
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1988));
```

Приведенный пример сформирует и выведет в окно браузера строку "Jan-01-1988".

Функция **gmdate**. Возвращает время по Гринвичу в строковом виде. Аналогична функции date () (см. урок 18). Используется, если посетителям

сайта предоставляется возможность выбора своего часового пояса. Так, строка

```
echo gmdate("M-d-Y", mktime(0, 0, 0, 1, 1, 1988));
```

выведет в окно браузера "Dec-31-1987" (если на WEB-страница расположена в часовом поясе Москвы).

PHP. Урок 21. Форма с проверкой заполнения

На этом уроке мы с вами рассмотрим решение стандартной задачи - а именно создание формы с обязательными полями для заполнения. Если посетитель такой WEB-странички не заполнит обязательные поля и нажмет на кнопку submit, то он останется на той же странице, и для него будет выведена информация об незаполненных полях.

Сначала приведем код, которые затем обсудим:

```
<html>
<head>
  <title>Форма с обязательным заполнением полей</title>
</head>
<?php
  function show_form($l_name, $email) {
  ?>
  <form action="form.php" method="post">
        Ваше имя *<input type="text" name="l_name"
value="<?php print $l_name ?>"><br>
    E-mail*<input type="text" name="email" value="<?php
print $email ?>"><br>
          <input     type="submit"     name="submit"
value="Отправить!"><input type="Reset">
  </form>
  <?
  }
  function check_form($l_name, $email) {
    if(!$l_name || !$email):
      print("Вы не заполнили нужные поля!<br>");
    if(!$l_name) {
      print("Введите Ваше имя.<br>");
    }
    if(!$email) {
      print("Введите ваш e-mail.<br>");
    }
    show_form($l_name, $email);
    else:
```

```
      confirm_form($l_name, $email);
    endif;
  }
  function confirm_form($l_name, $email) {
  ?>
    <h2>Введенная Вами информация:</h2>
  <?
    print("<br>$l_name<br>$email\n");
  }
  if(!$submit):
  ?>
  <p>Введите информацию о себе</p>
  <p>Все поля обязательны для заполнения.<p>
  <?php
    show_form("", "");
  else:
    check_form($l_name, $email);
  endif;
  ?>
</body>
</html>
```

Теперь начинаем обсуждение приведенного кода:

Как видно, у нас тут есть три функции - show_form, check_form и confirm_form. Они предназначены для показа, проверки и подтверждения правильности заполнения формы. Каждая из них принимает по два параметра - $l_name, $email, в которых сохраняются введенные посетителем страницы имя и e-mail. Рассмотрим эти функции более подробно.

Функция show_form:

```
  ...
  function show_form($l_name, $email) {
  ?>
  <form action="form.php" method="post">
        Ваше имя *<input type="text" name="l_name"
value="<?php print $l_name ?>"><br>
    E-mail*<input type="text" name="email" value="<?php
print $email ?>"><br>
          <input type="submit" name="submit"
value="Отправить!"><input type="Reset">
  </form>
  <?
  }
  ...
```

Обратите внимание на то, что тело функции (выводимая форма) фактически находится вне тегов php-скрипта:

```
...
?>
...
<?
}
```

Этот прием можно использовать вместо того, чтобы писать внутри php-функции большое количество операторов echo или print. Просто мы выключаем форму из php-кода, но это не означает, что она не будет выводится при вызове функции show_form - как раз наоборот - форма выводится будет, причем поля формы будут заполнятся значениями переменных $l_name и $email, передаваемых в качестве параметров в функцию show_form. При первом вызове сценария эти переменные содержат пустые строки, так что посетитель увидет пустую страничку, если же посетитель заполнит какое-нибудь из этих полей, то соответствующая переменая примет некоторое значение, которое и выведется в поле формы.

Следующая функция - check_form. Она проверяет заполнение посетителем всех необходимых полей:

```
function check_form($l_name, $email) {
  if(!$l_name || !$email):
    print("Вы не заполнили нужные поля!<br>");
  if(!$l_name) {
    print("Введите Ваше имя.<br>");
  }
  if(!$email) {
    print("Введите ваш e-mail.<br>");
  }
  show_form($l_name, $email);
  else:
    confirm_form($l_name, $email);
  endif;
}
```

Сначала мы проверяем две переменных $l_name и !$email, связанные с соответствующими полями формы. Если какое-нибудь из них не заполнено, то мы пишем в окне браузера "Вы не заполнили нужные поля!", выводим информацию о незаполнении посетителем конкретных полей, после чего вызываем функцию show_form, которая опять выводит нашу форму, причем заполенные посетителем поля передаются в функцию show_form и уже оказываются заполненными. Если же посетитель заполнит все поля (т. е.

условие !$l_name || !$email оказывается ложным), то мы вызываем функцию confirm_form, которую мы описываем в нашем коде дальше.

Функция confirm_form:

```
function confirm_form($l_name, $email) {
?>
  <h2>Введенная Вами информация:</h2>
<?
  print("<br>$l_name<br>$email\n");
}
```

Здесь мы просто выводим введенную посетителем информацию (т. е. значения $l_name и $email). Как и в случае с функцией show_form, часть кода (а именно строчку <h2>Введенная Вами информация:</h2>) мы пишем вне тегов php-скрипта, чтобы не возится с операторами echo или print.

И, наконец, главная часть скрипта:

```
if(!$submit):
?>
<p>Введите информацию о себе</p>
<p>Все поля обязательны для заполнения.<p>
<?php
  show_form("", "");
else:
  check_form($l_name, $email);
endif;
?>
```

Обратите внимание, что она расположена вне всех функций. Это означает, что она и будет сразу выполнятся на WEB-страничке (то же, что мы написали внутри функций, будет выполнятся только при вызове этих функций). Что мы тут делаем? Сначала мы проверяем значение переменной $submit. Если оно чему-то равно (это означает, что посетитель заполнил форму и нажал на кнопку submit), то мы вызываем функцию check_form, которая проверяет заполнение формы, если же посетитель на кнопку submit не нажимал, то вызывается функция show_form с пустыми параметрами, что эквивалентно показу незаполненной формы.

С этим уроком все!

PHP. Урок 22. Функция isset()

На предыдущем уроке мы проверяли, установлена ли переменная $submit (т. е. нажал ли пользователь кнопку submit) таким образом:

```
...
if(!$submit)
   ...
```

При запросе страницы с такой строчкой выдавалось предупреждение:

Notice: Undefined variable: submit in D:\Program Files\PHP Expert Editor\php34.tmp on line 35

("Неопределенная переменная submit в файле D:\Program Files\PHP Expert Editor\php34.tmp на строке 35")

Вообще же это правильнее делать через функцию isset(), которая принимает в качестве параметра имя переменной. Естественно, если для переменной задано некоторое значение, то функция isset() возвратит истину, если не задано, то ложь.

Таким образом, указанное место нашего сценария лучше переписать так:

```
...
if(!isset($submit))
   ...
```

Предупреждение в браузер уже выдаваться не будет.

PHP. Урок 23. Два варианта для функции для вывода HTML-кода

Часто php-код используется для того, чтобы выводить в браузер некоторую большую часть HTML-странички. Чаще всего для этого мы пишем отдельную функцию, для которой существует два способа написания.

Способ первый:

```
<?
function show_html(){
  print "<h1>Глава 1</h1>";
  print "<p>В этой главе мы рассмотрим....</p>";
}
?>
```

Как вы видите, мы тут используем оператор print и всю функцию пишем внутри тегов <?...?> (т. е. такая пара тегов у нас только одна).

Второй способ:

```
<?
function show_html(){
?>
  <h1>Глава 1</h1>
  <p>В этой главе мы рассмотрим....</p>
<?
}
?>
```

Здесь мы уже заключаем в теги <?...?> только начало и конец функции:

```
<?
function show_html(){
?>
  ...
<?
}
?>
```

Все же, что функция должна показать в браузере пользователя, мы пишем обычным HTML-кодом - без операторов print, без кавычек и без завершающих точек с запятой.

Разумеется если для второго способа внутри функции встречаются некоторые php-операторы (например, if), то мы должны и их заключить в теги <?...?>.

И в первом, и во втором случаях мы вызываем эту функцию совершенно одинаково:

```
...
<?
  show_html();
?>
...
```

Результат будет тоже одинаков - вывод HTML-кода с соответствующим форматированием.

PHP. Урок 24. Модульное строение страницы

Страницы сайтов обычно строятся по модульному принципу - например, наверху каждой страницы находится название сайта с логотипом, внизу - знак охраны авторских прав, слева - навигационная панель, посредине -

собственно содержание WEB-страницы (статья, рецензия и т. п.). При этом на всех страницах все части, кроме содержания, одинаковы (т. е. одинаковая шапка, панель навигации т. п.).

Делать такие страницы отдельно - это плохой тон. Представьте себе, что вы добавили на сайт еще один раздел и вам понадобилось изменить панель навигации на каждой странице сайта. Даже страшно подумать, если их несколько сотен. Поэтому мы сейчас и посмотрим, как нужно строить страницу по модульному принципу - т. е. каждая часть страницы будет хранится в отдельном файле - верх страницы с логотипом в одном, навигация в другом и т. п. Теперь, если вам, например, надо изменить навигацию, то это достаточно будет сделать только в одном файле.

Каждая наша страница будет состоять из четырех частей: верхняя часть с названием сайта, нижняя часть с информацией об авторских правах, левая часть с навигацией и информационная правая часть.

Каждая страница будет "собираться" из 5 файлов - по одному для каждой части и плюс еще файл, объединяющий все четыре части. Вот перечисление всех необходимых файлов:

- xxxxx.php - файл с информацией. Именно этот файл и будет набирать посетитель в строке браузера. Включает в себя файл main.inc. Имя этого файла будет свое для каждой страницы сайта. В нашем примере таких страниц будет три: 1.php, 2.php и 3.php.
- main.inc - файл, задающий общее расположение элементов на WEB-странице. Включает в себя все остальные файлы, кроме xxxxx.php.
- header.inc - файл верхней частью страницы (с названием сайта).
- footer.inc - файл нижней частью страницы (с авторскими правами).
- nav.inc - файл с навигацией по сайту (располагается в левой части каждой страницы).

Приведем код для каждого файла.

Файл 1.php (это один из информационных файлов xxxxx.php):

```
<?
$content="
<h2>Мебель</h2>
<p>Добро пожаловать на наш сайт!...</p>
";
include ("main.inc");
?>
```

Как вы видите, тут мы заводим переменную $content, в которую записываем информафионное содержание нашей страницы. В эту переменную можно

записывать в том числе и теги (что мы, собственно говоря, и делаем). Кроме того в эту страницу включается посредством оператора include файл main.inc. include мы делаем после объявления переменной $content, так как в файле main.inc эта переменная будет использоваться. Обратите также внимание на то, что все содержимое файла 1.php мы заключаем в теги php.

Файлы 2.php и 3.php мы делаем абсолютно аналогично:

```
<?
$content="
<h2>Столы</h2>
<p>Наша фирма рада предложить Вам следующие столы
...</p>
";
include ("main.inc");
?>
<?
$content="
<h2>Шкафы</h2>
<p>Мы рады предложить Вам следующие модели шкафов
...</p>
";
include ("main.inc");
?>
```

Файл main.inc:

```
<html>
<head>
  <title></title>
</head>

<body>
<table cellspacing="2" cellpadding="2" border="0"
width=100%>
<tr>
    <td colspan="2">
    <?include ("header.inc");?>
    </td>
<tr>
<tr>
    <td width="20%"></td>
    <td></td>
</tr>
<tr>
    <td colspan="2" align=right>
```

```
    <?include ("footer.inc");?>
    </td>
</tr>
<table>
</body>
</html>
```

Тут мы, по сути дела, задаем макет для нашей HTML-странички - задаем теги <head>, <body> и другие, задаем расположение элементов на странице посредством таблицы. В первую ячейку таблицы мы вставляем файл заголовка с названием сайта:

```
    ...
    <td colspan="2">
    <?include ("header.inc");?>
    </td>
    ...
```

в последнюю - файл с авторскими правами:

```
    ...
    <td colspan="2" align=right>
    <?include ("footer.inc");?>
    </td>
    ...
```

Средний ряд таблицы состоит из двух ячеек, в которые мы вставляем файл с навигацией и значение переменной $content:

```
<tr>
    <td width="20%"><?include ("nav.inc");?></td>
    <td><?print $content?></td>
</tr>
```

Переходим к файлам header.inc, footer.inc и nav.inc.

Файл header.inc:

```
<h1>Мебельный сайт</h1>
```

Файл footer.inc:

```
(c) Copywrite Alexeev Igor, 2002-2003
```

Файл nav.inc:

```
<a href="1.php">Главная</a><br>
```

```
<a href="2.php">Столы</a><br>
<href="3.php">Шкафы</a>
```

Эти файлы одинаковы для всех страниц сайта. В них мы описываем верхнюю и нижнюю части для каждой WEB-страницы сайта, а также панель навигации.

Все, стараницы созданы! Разместите их в одном каталоге на вашем WEB-сервере, и через браузер просмотрите странички 1.php, 2.php и 3.php. Слева на каждой страничке будет панель навигации, справа - содержательная часть страницы, наверху и внизу - название сайта и информация об авторских правах. Если вы просмотрите через браузер HTML-код для запрашиваемых страниц 1.php, 2.php и 3.php, то никакого кода php, естественно, не увидите - тем будет только сгенерированный WEB-сервером HTML-код.

PHP. Урок 25. Модульное строение страницы (с параметрами)

На прошлом уроке мы рассмотрели создание HTML-страниц через шаблоны. Один из недостатков у генерируемых страниц в нашем варианте страниц был тот, что у панели навигации все ссылки были активны, в том числе и ссылка на страницу, на которой мы находились. Это не слишком хорошо - по идее страница, на которой мы находимся, должна отображаться не в виде гиперссылки, а в виде просто текста (названия). Исправим это. Для этого внесем изменения во все информационные файлы (т. е. в файлы 1.php, 2.php и 3.php) и в файл nav.inc. Все остальные файлы нам менять не надо.

Смысл изменений в файлы 1.php, 2.php и 3.php такой - в каждый из них мы добавим переменную с номером раздела нашего сайта, в файле же nav.inc мы будем эту переменную анализировать в выдавать в браузер или гиперссылку, или просто текст.

Вот конкретные изменения.

Файл 1.php:

```
<?
$div=1;
$content="
<h2>Мебель</h2>
<p>Добро пожаловать на наш сайт!...</p>
";
include ("main.inc");
?>
```

Файл 2.php:

```
<?
$div=2;
$content="
<h2>Столы</h2>
<p>Наша фирма рада предложить Вам следующие столы
...</p>
";
include ("main.inc");
?>
```

Файл 3.php:

```
<?
$div=3;
$content="
<h2>Шкафы</h2>
<p>Мы рады предложить Вам следующие модели шкафов
...</p>
";
include ("main.inc");
?>
```

Как видно, мы в каждый из информационных файлов добавили переменную $div со значениями 1, 2 или 3. Именно через эту переменную мы и будем узнавать, в каком разделе сайта мы находимся.

Изменения в файле nav.inc:

```
<?
if($div==1){
?>
Главная<br>
<?
}
else{
?>
<a href="1.php">Главная</a><br>
<?
}
?>

<?
if($div==2){
?>
Столы<br>
<?
```

```
}
else{
?>
<a href="2.php">Столы</a><br>
<?
}
?>

<?
if($div==3){
?>
Шкафы<br>
<?
}
else{
?>
<a href="3.php">Шкафы</a><br>
<?
}
?>
```

Тут изменения более существенные, хотя в них тоже ничего сложного нет. Мы анализируем значение переменной $div и выводим в браузер или гиперссылку или (если находимся в соответствующем разделе) просто текст.

Все! Теперь название раздела, в котором мы находимся, будет показываться в виде простого текста, а не в виде гиперссылки.

PHP. Урок 26. Сессии

Сначала пару слов о том, что такое сессии и с чем их едят. Представьте себе, что вам надо хранить информацию о посетителе вашего WEB-узла, когда он переходит от страницы к странице. Например, на первой странице сайта посетитель может зарегистрироваться, и надо, чтобы последующие страницы узнавали нашего посетителя (скажем, на некоторые из них доступ ему может быть разрешен, а на некоторые - запрещен). Короче говоря, нам нужен некоторый идентификатор посетителя, который будет всегда доступен, пока посетитель не покинул данный WEB-узел. Так вот именно для этой цели и служат сессии.

Теперь практический пример. У нас будет некоторая переменная, которая будет задана и выведена в окно браузера на странице 1.php. При переходе на вторую страницу эта переменная сохранится - мы увидим это, так как просто покажем ее в окне браузера.

Текст файла 1.php:

```
<?
  session_start();
  $a = "Некоторое значение";
  session_register("a");
?>
<html>
<head>
  <title>Сессии</title>
</head>
<body>
<h1>Сессии</h1>
<?
  echo $a;
?><br>
<p>Переход на страницу <a href="2.php">2.php</a></p>
</body>
</html>
```

Итак, мы тут объявляем сессию путем вызова функции session_start();. Далее мы заводим некоторую переменную $a, которую затем регистрируем путем вызова функции session_register("a"); (имя переменой мы пишем при этом без знака $). Обратите также внимание, что это все мы делаем **до** открывающего тега <html> - это важно. На самой же WEB-страничке мы просто показываем значение переменной $a.

Теперь код для странички 2.php:

```
<?php
  session_start();
?><html>
<head>
  <title>Сессии</title>
</head>
<body>
<h1>Сессии</h1>
<?
  echo $a;
?>
</body>
</html>
```

Как вы видите, код тут аналогичный. Опять мы тут вызываем функцию session_start(); и делаем мы это **до** тега <html>. Затем мы на страничке просто выводим значение нашей переменной $a, которрую мы задали на

предыдущей странице. Естественно, в окно браузера выведется надпись "Некоторое значение".

Таким обрахоп переменая $a будет доступна на всех страницах сайта, на которых мы запустили сессии.

Вот еще парочка функций, связанных с сессиями:

session_unregister("a");. Эта функция позволяет отменить регистрацию переменной ($a в данном случае).

session_destroy(); - завершеие сессии. Эта функция может пригодится, например, когда посетитель сначала зарегистрировался на сайте (скажем для просмотра своей электронной почты, а потом хочет выйти из режима просмотра своей переписки, но при этом остаться на сайте).

V. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВЫХ ПРОЕКТОВ (РАБОТ)

Находятся в пункте 2.6 рабочей программы.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ (ПРАКТИКУМОВ)

Форма проведения лабораторного занятия: а) приветствие студентов, 1 мин.; б) определение личного состава студенческой группы, 4 мин.; в) объявление тематики и вопросов лабораторного занятия, 1 мин.; г) выполнение заданий, 70 мин.; д) подведение итогов лабораторного занятия, 13 мин.; е) прощание со студентами, 1 мин.

Список рекомендуемой литературы (основной и дополнительной) приведен в рабочей программе.

VII. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ (СЕМИНАРСКИМ) ЗАНЯТИЯМ

Практические и семинарские занятия не предусмотрены.

VIII. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ДОМАШНИХ ЗАДАНИЙ И КОНТРОЛЬНЫХ РАБОТ

При выполнении домашних работ необходимо использовать конспекты лекций, любую дополнительную литературу.

Перед контрольной работой студентам необходимо повторить теоретические основы методов, указанных преподавателем. При выполнении контрольной работы конспект лекций и другую дополнительную литературу не использовать.

IX. ПЕРЕЧЕНЬ ПРОГРАММНЫХ ПРОДУКТОВ, РЕАЛЬНО ИСПОЛЬЗУЕМЫХ В ПРАКТИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ ВЫПУСКНИКОВ И СООТВЕТСТВУЮЩЕЕ УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ, РАСКРЫВАЮЩЕЕ ОСОБЕННОСТИ И ПЕРСПЕКТИВЫ ИСПОЛЬЗОВАНИЯ ДАННЫХ ПРОГРАММНЫХ ПРОДУКТОВ

Для обеспечения учебного процесса необходимы следующие программные средства: операционная система Windows 98 (NT,2000,ME,XP), офисный пакет MS Office 97(2000, XP), Internet Explorer 5.5 (Opera, Netscape Navigator), FTP клиент - Far менеджер (или CuteFTP).

X. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРИМЕНЕНИЮ СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ДЛЯ ПРЕПОДАВАНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ (В Т. Ч. РАЗРАБОТАННЫЕ ВЕДУЩИМИ ПРЕПОДАВАТЕЛЯМИ ФИЛИАЛА)

Данные методические указания отсутствуют.

XI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОМУ СОСТАВУ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО И ЭКЗАМЕНАЦИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ (МАТЕРИАЛЫ ПО КОНТРОЛЮ КАЧЕСТВА ОБРАЗОВАНИЯ)

Преподаватель готовит контролирующие материалы в виде тестов, задач и в другой форме. Тематика контролирующих материалов должна соответствовать тематике материалов, прочитанных студентам на лекционных занятиях к моменту контроля знаний, и тематике задач, разобранных на практических занятиях к моменту контроля знаний. Преподаватель самостоятельно выбирает форму теста, правила работы с контролирующими материалами, время на его выполнение. Во время проведения контроля знаний студентов преподаватель объясняет студентам правила работы с контролирующими материалами и выдаёт эти материалы студентам. После истечения установленного времени контролирующие материалы собираются и обрабатываются. Критерии оценки знаний преподаватель устанавливает самостоятельно. Студентам, не сдавшим тест или не присутствующим на нем по каким-либо причинам, предоставляется дополнительная возможность пройти тест.

XII. КОМПЛЕКТЫ ЗАДАНИЙ ДЛЯ КОНТРОЛЬНЫХ РАБОТ, ДОМАШНИХ ЗАДАНИЙ

Задания для контрольных работ и домашних заданий берутся из книг, указанных в рабочей программе.

XIII. ФОНД ТЕСТОВЫХ И КОНТРОЛЬНЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ ПО ДИСЦИПЛИНЕ

Фонд тестовых и контрольных заданий для оценки качества знаний по дисциплине приведен в приложении А.

XIV. КОМПЛЕКТЫ ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ ДЛЯ КАЖДОГО ИЗ ПРЕДУСМОТРЕННЫХ ЭКЗАМЕНОВ ПО ДИСЦИПЛИНЕ И КОНТРОЛЬНЫЕ ВОПРОСЫ К ЗАЧЕТУ

Комплекты экзаменационных билетов составляются на основе экзаменационных вопросов, представленных в п. 2.7 рабочей программы данной дисциплины, по следующей форме:

ГОУВПО «Амурский государственный университет»

Утверждено на заседании кафедры
«___» _____ 200 г.
Заведующий кафедрой –
Труфанова Т.В.
Утверждаю: _____

Кафедра *математического анализа и моделирования*
Факультет *математики и информатики*
Курс 5
Дисциплина "*Современные информационные технологии*"

Экзаменационный билет 1

1. Понятие информационных технологий. Приведите примеры ИТ.
2. Характеристика символа в документе.
3. Какой тег следует использовать для включения в html – документ изображения image.bmp?

XV. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Дисциплину в полном объёме ведёт: Подопригора Сергей Алексеевич, старший преподаватель кафедры математического анализа и моделирования.

ПРИЛОЖЕНИЕ А

ФОНД КОНТРОЛЬНЫХ РАБОТ

Практическое задание
для выполнения в табличном процессоре Excel

Обработка данных метеостанции

Задание: Имеется таблица, содержащая количество осадков в миллиметрах, построенная на основе наблюдений метеостанции г. Екатеринбурга. Наберите эту таблицу в тех же ячейках как на рисунке:

Таблица 1

	A	B	C	D
1	**Количество осадков (мм)**			
2	Таблица построена на основании наблюдений			
3	Метеостанции г. Екатеринбурга			
4		**1992**	**1993**	**1994**
5	Январь	37,2	34,5	8
6	Февраль	11,4	51,3	1,2
7	Март	16,5	20,5	3,8
8	Апрель	19,5	26,9	11,9
9	Май	11,7	45,5	66,3
10	Июнь	129,1	71,5	60
11	Июль	57,1	152,9	50,6
12	Август	43,8	96,6	145,2
13	Сентябрь	85,7	74,8	79,9
14	Октябрь	86	14,5	74,9
15	Ноябрь	12,5	21	56,6
16	Декабрь	21,2	22,3	9,4

Определить *для всей таблицы в целом:*

1) минимальное количество осадков, выпавшее за 3 года;
2) суммарное количество осадков, выпавшее за 3 года;
3) среднемесячное количество осадков по итогам 3-летних наблюдений;
4) максимальное количество осадков, выпавшее за 1 месяц, по итогам 3-летних наблюдений;
5) количество засушливых месяцев за все 3 года, в которые выпало меньше 10 мм осадков.

Данные оформить в виде отдельной таблицы:

Таблица 2

	E	F	G
1			
2			
3		Данные за 1992-1994 годы	
4		Макс. кол-во осадков за 3 года (мм)	
5		Мин. кол-во осадков за 3 года (мм)	
6		Суммарное кол-во осадков за 3 года (мм)	
7		Среднемесячное кол-во осадков за 3 года (мм)	
8		Кол-во засушливых месяцев за 3 года	

Практическое задание для выполнения в текстовом редакторе Word

Задание: Создать документ с таблицей с заданными условиями.

Название таблицы – 16 п, полужирный курсив, шапка и основной текст – 14.
Цветовое оформление : ячейки с адресами магазинов – заливка серым, столбцы Компьютеры IKS –заливка оранжевым, Компьютеры Zet – заливка оранжевым. Перед адресами магазинов вставьте символ конверта, перед телефоном – символ

телефона.
® - вставка, специальные символы

Фирма «Панорама»

Парковая пл. 13, 890-45-56 Веселова 8, 544-67-38 Футбольный 13,780-56-34 Чемпионский 65, 567-35-56		
www.iks.ru Курс	Нал.=31.09 Курс	Б/нал=32.89
Системный блок Корпус, дисковод, видео мат.плата, процессор, память	Компьютеры IKS Стоимость программного обеспечения не входит в цену	Компьютеры Zet В комплекте Windows 2099 Гарантия 2 года
Intel® Pentium ®41500	408	509
Intel® Pentium ®42200	715	818
Intel® Pentium ®41700	435	536

Оформить колонтитулы следующим образом:
Верхний колонтитул должен содержать полное имя файла и имя автора, нижний –дату и номер страницы.

Задание: Создайте HTML-документ

Создайте список "Устройства вывода информации"

A. *Мониторы*
- ЭЛТ- мониторы
  - Samsung SyncMaster
  - Nec MultiSync
- ЖК-мониторы
  - RowerScan
  - LG Flatron
  - ViewSonic

B. *Принтеры*
- Матричные
- Струйные
- Лазерные
  - HP LaserJet
  - Canon
  - Xerox

C. *Плоттеры*
- Струйные

- Лазерные
- Электростатические

Монитор

СОДЕРЖАНИЕ