СОДЕРЖАНИЕ

1. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

1.1. Цель и задачи выполнения практических занятий

В соответствии с учебным планом подготовки студентов по направлению 230100 «Информатика и вычислительная техника» (бакалавриат) и рабочей программой по дисциплине «Информатика» и изложенными в них требованиями к уровню подготовки инженеров для работы в организациях ГА студенты должны обладать практическими навыками и компетенциями в решении задач, связанных с использованием различных сценарных языков и технологий.

Особенностью данного пособия является его прикладная направленность, что способствует формированию у студентов компетенций:

- ОК-1 -владеет культурой мышления, способен к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения;
- ОК-11- осознает сущность и значение информации в развитии современного общества; владеет основными методами, способами и средствами получения, хранения и переработки информации;
- ОК-12 -имеет навыки работы с компьютером как средством управления информацией;

-ПК-2 - осваивать методики использования программных средств для решения практических задач;

- ПК-5- разрабатывать компоненты программных комплексов и баз данных, использовать современные инструментальные средства и технологии программирования.

Целью данного пособия является закрепления студентами теоретического курса дисциплины и приобретение навыков программирования с использованием сценарных языков Bourne shell , PHP.

1.2. Основные вопросы, подлежащие изучению

1. Логика работы ОС семейства UNIX.
2. Освоение приемов и методов работы командного интерпретатора shell.
3. Изучение технологии написания сценариев на языке Bourne shell.
4. Создание и публикация web-документов.
5. Освоение приемов и методов работы с работы с PHP-машиной.
6. Проектирование элементов пользовательского интерфейса.

2. ПЕРЕЧЕНЬ ТЕМ ПРАКТИЧЕСКИХ ЗАНЯТИЙ (12 часов)

I-й семестр (4 часа)

ПЗ -1 Основные команды ОС UNIX. Ввод и вывод данных. Перенаправление. Создание учетной записи пользователя (2 часа)

ПЗ -2 Командный интерпретатор shell. Язык Bourne shell: команды, функции программы (2 часа)

2 семестр (8 часов)

ПЗ -3 Создание простого и составного HTML-документа (2 часа)

ПЗ -4 Проектирование элементов пользовательского интерфейса (2 часа)

ПЗ -5, 6 Практические аспекты создания Web-приложений (4часа)

3. СОДЕРЖАНИЕ ЗАНЯТИЙ

В 1-м семестре учебным планом предусмотрено два практических занятия продолжительностью 2 академических часа каждое.

3.1. Занятие 1. Основные команды ОС UNIX. Ввод и вывод данных. Перенаправление. Создание учетной записи пользователя (2 часа)

3.1.1. Цель занятия

- ознакомление с форматом основных команд семейства ОС UNIX;
- изучение особенностей ввода-вывода и создание каналов;
- установление прав доступа к файлу;
- создание учетной записи пользователя, просмотр конфигурационных файлов.

3.1.2. Методические указания по теме

Для выполнения заданий по данной теме необходимо заранее изучить разделы 1,2 [1].

3.1.3. Примеры заданий

Задание 1.

Укажите команду, позволяющую определить текущую директорию.

Решение:

$ pwd

Задание 2.

Создайте файл 1txt, введите в него пример шаблона имени файла: второй символ а, четвертый символ цифра 7, оканчивается файл текущим годом.

Решение:

$ cat>1txt

(далее нажать клавишу Enter)

*?a?7*2013*

Для сохранения файла необходимо ввести комбинацию клавиш Ctr+D, для установления прав на выполнение, например:

$ chmod 777 1txt

или

$ chmod u+x 1txt

Задание 3

Измените права файла 1txt: пользователю - владельцу файла установить только право на исполнение, группе, являющейся владельцем файла, и остальным – только чтение:

Решение:

$ chmod u=x, go=r 1txt

Задание 4

Дописать в файл 1txt строчку Hello,evm!

Решение:

$ cat>>1txt

(далее нажать клавишу Enter)

Hello,evm!

Для сохранения файла необходимо ввести комбинацию клавиш Ctr+D, для установления прав на выполнение, например:

$ chmod 777 1txt

Задание 5

Вывести на экран имена файлов текущей директории:

Решение:

$ ls

Задание 6

Вывести на экран имена файлов текущей директории в длинном формате:

Решение:

$ ls -l

Задание 7

Вывести на экран имена файлов текущей директории, включая скрытые, в длинном формате:

$ ls -al

Задание 8

Вывести на экран информацию о процессах в коротком формате:

Решение:

$ ps

Задание 9

Вывести на экран информацию о процессах в полном формате:

Решение:

$ ps -f

Задание 10

Скопировать из домашней директории все файлы, имена которых соответствуют шаблону: первый символ лежит в диапазоне от a до z, второй символ может включать цифры от 4 до 8, последний символ не должен содержать цифры 0 и 1 в директорию tmp.

Решение:

$ cp [a-z][4-8][!01]*

или

$ cp [a-z][4-8][^01]*

Задание 11

Вывести информацию из файла 1txt на экран.

Решение:

$ cat 1txt

Или

$ cat<1txt

Задание 12

Перенаправить вывод команды ls –l в файл 2txt.

Решение:

$ ls –l>2txt

Задание 13

Ввести команду для смены пользователя, например, root.

Решение:

$ su root

Задание 14

Необходимо найти строки в файле 3txt, в которых первым символом является произвольная цифра, второй символ выбирается из диапазона от 0 до 5, а третий символ принадлежит диапазону от 0 до 6.

Решение:

$ grep ' [0-9][0-5][0-6] ' 3txt

Задание 15

Создать нового пользователя в системе с именем user123, паролем 1234567. Выполнять проверку просмотром файла /etc/passwd и /etc/shadow.

Решение:

 # adduser –d /home/user123 user123

 # cat /etc/passwd

 # cat /etc/shadow

 # passwd user123

 # cat /etc/passwd

 # cat /etc/shadow
или
 # adduser –d /home/user123 user123
 # who

Задание 16(контрольное).

Создать файл *data.f* Выполнить в файле *data.f* поиск строк, не начинающихся с заданных символов, например, цифры 4 или 8. Далее скопировать данный файл в созданную директорию test, ранее для которой были установлены права 344. Если операция копирования не может быть выполнена, изменить соответствующие права.

Контрольные вопросы

1. Объясните синтаксис команд.
2. Как можно группировать ключи?
3. Что понимается под термином «перенаправление»?
4. Как добавить нового пользователя в систему, кто может выполнить данную операцию?
4. Как выполнить редактирование конфигурационных файлов?

3.2. Занятие 2. Командный интерпретатор shell. Язык Bourne shell: команды, функции программы (2 часа)

3.2.1. Цель занятий

- изучение структуры файла сценария;
- получение навыков программирования на языке Bourne shell.

3.2.2. Методические указания по теме

Для выполнения заданий по данной теме необходимо заранее изучить разделы 3.2-3.5[1].

3.2.3. Задания для самостоятельного решения

Задание 1.

Написать файл-сценарий, содержащий комментарии к командам: создание директории; установление прав на чтение и исполнение для пользователя, являющегося владельцем файла; ввод трех переменных с клавиатуры; вывод на экран суммы введенных переменных.

Задание 2.

Написать файл-сценарий, содержащий следующие команды: ввод двух переменных с клавиатуры, третья переменная - позиционный параметр. Если позиционный параметр при запуске отсутствует - вывести сообщение на экран, если параметр задан- посчитать произведение трех переменных.

Задание 3.

Написать файл-сценарий, содержащий команду test, проверяющую выполнение условия равенства двух заданных переменных. В противном случае - выполнить операцию деления первой переменной на вторую.

Задание 4.

Написать файл-сценарий, позволяющий в цикле подсчитать произведение 10 введенных чисел.

Задание 5.

Написать файл-сценарий, выводящий на экран:

- значение, возвращенное последней командой;
- номер процесса;
- номер фонового процесса;
- число позиционных параметров, передаваемых в shell.

Задание 6.

Создать файл функции, включающий одну функцию. Эта функция будет загружена интерпретатором команд, протестирована, изменена, а затем повторно загружена.

Задание 7.

Используя команду set выполнить проверку загруженности интерпретатором команд функции, доступных для сценария, созданного в задании 6.

Контрольные вопросы

1. Поясните общую структуру файла-сценария.
2. В каком виде хранятся переменные в программах командного интерпретатора?
3. Что такое параметры?
4. Для каких целей они используются?
5. Какое число параметров может быть передано процедуре?
6. Перечислите переменные (параметры), которым интерпретатор shell автоматически присваивает значения.
7. Как осуществляется ветвление вычислительного процесса процедуры?
8. Какого типа циклы в процедурах могут быть построены средствами языка shell?
9. Поясните процедуру создания файла функций.

II-й семестр (8 часов)

3.3. Занятие 3. Создание простого и составного HTML-документа (2 часа)

3.3.1. Цель занятий

- ознакомление с технологией создания и публикацией простого и составного HTML-документа.

3.3.2. Методические указания по теме

Одна из причин популярности Web и разнообразия ее содержания как раз и заключается в том, что благодаря простоте HTML многие смогли изучить этот язык и создать свои Web -страницы. Название языка HTML является сокращением от HyperText Markup Language, т.е. "язык гипертекстовой разметки". Обычно любой текст - длинная строка символов, которая читается в одном направлении. Гипертекстовая технология заключается в том, что текст представляется как многомерный, т.е. с иерархической структурой типа сети. Под *гипертекстом* понимают систему информационных объектов (статей), объединенных между собой направленными связями, образующими сеть. Гипертекст обладает нелинейной сетевой формой организации материала, разделенного на фрагменты, для каждого из которых указан переход к другим фрагментам по определенным типам связей. При установлении связей можно опираться на разные основания (ключи), но в любом случае речь идет о смысловой, семантической близости связываемых фрагментов.

Гипертекст содержит не только информацию, но и аппарат ее эффективного поиска. По глубине формализации информации гипертекстовая технология занимает промежуточное положение между документальными и фактографическими информационными системами. Гипертексты, составленные вручную, используются давно - это справочники, энциклопедии, а также словари, снабженные развитой системой ссылок.

Язык HTML был задуман в первую очередь как универсальный язык для функциональной классификации различных частей документа. *Разметкой* называется вставка в текст дополнительных служебных символов (например, пометки корректора тоже можно считать своего рода разметкой). Каждый служебный символ в HTML представляет собой команду, которая указывает браузеру, как следует отображать текст. Это позволяет отображать документы на самых разных платформах. Разметка может быть как очень простой, так и чрезвычайно сложной. Все что требуется разработчику простого и составного Web-документа - изучить команды HTML, называемые *тегами (tags).*

Теги HTML

Тегами (tags) называется единица разметки - особый набор символов, имеющий в HTML особое значение. Теги начинаются со знака <, за которым следует ключевое слово, и заканчиваются знаком >. Например, теги могут выглядеть так:

<Strong>
<TITLE>
</B>
<html>

Каждый тег в HTML имеет определенный смысл, в котором обычно нет ничего сложного. Например, тег <B> означает переключение на полужирный шрифт, а тег <HR>- вставляет в документ горизонтальную линию.

Регистр в тегах не учитывается, поэтому теги <Title>,<title>, <TITLE>, <titLE> считаются одинаковыми.

Начальные и конечные теги

Теги делятся на две категории. *Начальный тег* открывает действие некоторого эффекта, а конечный тег - отменяет его. *Конечный тег* всегда выглядит как ключевой слово, перед которым стоит символ / (косая черта или слэш).

Например, текст, отображаемый полужирным шрифтом, заключается между начальным тегом <B> и конечным тегом </B>:

Люблю тебя <B>Петра</B> творенья

В результате слово **Петра** выделяется полужирным шрифтом.

Теги можно вкладывать внутрь других тегов. Например,

Быть или не быть , <B><I>вот</I></B> в чем вопрос.

Слово ***вот*** будет выводиться одновременно и полужирным, и курсивным шрифтом. Обратите внимание, что начальный и конечный тег курсивного начертания полностью содержатся внутри пары тегов для полужирного начертания.

Атрибуты тегов

Многие начальные теги обладают атрибутами, влияющими на поведение данного тега. *Атрибуты* представляют собой ключевые слова, находящиеся в угловых скобках и отделенные от имени тега пробелом - например, <HR NOSHADE> (данный тег рисует горизонтальную линию без теневого выделения). Для некоторых значений атрибутов необходимо указать значение, перед которым стоит знак = (например, тег <HR WIDTH="200"> рисует горизонтальную линию шириной в 200 пикселов).

Следует помнить, что в браузере отражаются не сами теги, а их эффекты. Неизвестные теги и атрибуты игнорируются.

Гипертекстовые ссылки

Ссылка на другой документ описывается следующим образом:

<A HREF="*имя файла*"> Текст, который будет служить как обращение к другому документу</A>.

Приведем пример такой гипертекстовой ссылки:

<A HREF="minihtml.html">Пример HTML-текста</A>

Здесь ключевые слова «Пример HTML-текста» являются гиперссылкой на файл minihtml.html, который лежит в той же директории, что и текущий документ. Можно ссылаться на документ, лежащий в любой директории, описав к нему полный путь. Так, например, ссылку на файл a.html, лежащий в поддиректории Test можно описать как:

<A HREF="Test/a.html">Здесь ссылка </A>

Это так называемые *относительные ссылки*. Можно использовать абсолютное имя файла (полный путь). В общем случае использование ссылки по абсолютному имени файла более предпочтительно.

<A Href="http://www.mstucal.ru/test/a.hmml">Здесь ссылка</A>

Гиперссылки могут также использоваться для соединения с определенными разделами документов. Предположим, мы хотим соединить документ А с первой главой документа В, для чего нам необходимо создать *именованную гиперссылку* в документе В:

Здесь вы можете увидеть <A NAME = "Глава 1">Главу 1 Текст первой главы.

Теперь, описывая ссылку в документе А, надо включить не только имя файла "documentB.html"но также и имя гиперссылки, отделяемое символом (#):

Здесь вы можете увидеть текст<A HREF = "documentB.html#Глава1"> Главы 1 </A> документа В.

Таблицы

Таблицы удобны для представления больших объемов данных, а также для точного размещения элементов Web-страниц. Таблица в языке HTML задается при помощи парного тега <TABLE>. Она может содержать *заголовок таблицы,* определяемый парным тегом <CAPTION>, и *строки таблицы,* задаваемые при помощи парных тегов <TR>. Закрывающие теги можно опускать.

Каждая строка таблицы содержит *ячейки таблицы,* которые могут относиться к двум разным типам. Ячейки в заголовках столбцов и строк задают парным тегом <TH>, а обычные ячейки - парным тегом <TD>. Закрывающие теги </TH> и </ TD> можно опускать. Например, «пустая» таблица с двумя строками и двумя столбцами может быть задана следующим образом:

```
<TABLE>
<CAPTION>Пустая таблица</CAPTION>
<TR><TD><TD>
<TR><TD><TD>
</TABLE>
```

Каждая ячейка может содержать произвольный текст, а также любые теги HTML, допустимые в теле документа. В частности, ячейка может содержать вложенную таблицу или изображение.

При отображении таблицы на экране компьютера происходит ее автоматическое форматирование с подбором ячеек в соответствии с объемом размещаемой информации и заданными атрибутами. Атрибуты элементов позволяют оформить таблицу в разных стилях. В таблице приведена краткая характеристика допустимых атрибутов.

Таблица

Краткая характеристика допустимых атрибутов

Атрибут	Элемент	Назначение
ALIGN=	Таблица, заголовок, строка, ячейка	Выравнивание таблицы по горизонтали; заголовок, выравнивание данных по горизонтали; размещение заголовка над или под таблицей
VALIGN=	строка, ячейка	Выравнивание по вертикали
WIDTH=	таблица, ячейка	Ширина
HEIGHT=	ячейка	Высота
COLSPAN=	ячейка	Протяженность в несколько столбцов
ROWSPAN=	ячейка	Протяженность в несколько строк
BGCOLOR=	таблица, ячейка	Цвет фона
CELLSPACING=	таблица	Зазор между ячейками
CELLPADDING=	таблица	Зазор между содержимым ячейки и ее границей
BORDER=	таблица	Отображение границ ячеек и внешней рамки таблицы

Фреймы

Технология frames (рамки), разработанная фирмой Netscape и ставшей стандартом, продолжает широко использоваться в сетях. Использование фреймов можно рассматривать как способ вывода на экран монитора сразу несколько html-документов, не открывая для каждого собственное окно браузера.

Технология фреймов позволяет разбить окно браузера на несколько подразделов, и загрузить в каждый из них нужный документ. Фреймы могут существовать независимо, а могут и влиять друг на друга, используя систему ссылок. В этом еще одно их преимущество: возможность создать удобную систему навигации по сайту, определив один фрейм как навигационную панель, другой - как панель для рекламной информации и т.д.

Использование фреймов имеет недостаток - чтобы использовать фреймы, придется создать, как минимум, три файла: один - невидимый для пользователя - установочный, описывающий раскладку фреймов в окне браузера и назначающий исходные html- документы для каждого из окон, и два (или более) собственно исходных, которые будут помещены в назначенные для них разделы. Установочный файл - служебный, не содержит никакой информации, кроме той, которая определяет для браузера структуру фреймов. Исходные же файлы вполне стандартны, они могут нести в себе текст, графику, звук и прочее, используемое при создании Web-страницы.

Правилом хорошего тона в Web-дизайне является включение в установочный файл тега <NOFRAMES> </NOFRAMES>, который позволит сообщить пользователям старых браузеров, что данную страницу необходимо просматривать с помощью программы, поддерживающей технологию фреймов.

Тэг <FRAMESET> </FRAMESET> определяет структуру фреймовой области. Атрибут ROWS="?,?" тэга <FRAMESET> разбивает окно браузера на горизонтальные полосы. Знаки "?" должны быть заменены цифровыми значениями. Например, ROWS= "50,*" определяет верхнюю полосу высотой в 50 пикселов, а остальное пространство отдает нижней полосе. ROWS="20%,*" выделит верхней полосе 20% окна браузера (либо фрейма), остальное - пространство - останется нижней. Значение атрибута ROWS — "*" выделяет данной строке фрейма все оставшееся свободное пространство, еще не зарезервированное точными указаниями процентов или пикселов. Если вы хотите разбить окно на две, три, четыре равные части, то атрибут можно задать таким образом: ROWS="*,*,*" (окно разбито на три части). Атрибут COLS="?,?" разбивает выделенную под фреймы область на колонки. Можно задать ширину, как в пикселях, так и в процентах, либо воспользоваться знаком "*".

Для описания фреймов, входящих в структуру, заданную парным тэгом <FRAMESET></FRAMESET>, используется непарный тэг <FRAME>, не требующий закрывающей части. Тэг <FRAME> создает пустое пространство в окне браузера, а его особенности и формат определяются различными атрибутами. Атрибут MARGINHEIGHT="?" задает расстояние в пикселях между содержимым фрейма и его нижней и верхней границами, где "?" равно количеству пикселов.

Атрибут MARGINWIDTH="?" задает расстояние в пикселях между содержимым фрейма и его левой и правой границами, где "?" также равно количеству пикселов. Для того чтобы фрейм можно было использовать в дальнейшем, необходимо назначить ему уникальное имя. Имя это задается атрибутом NAME="?", этот атрибут имеет особое значение, т.е. он необходим для построения правильной навигации.

С момента разработки первой версии произошло довольно серьезное развитие языка. Почти вдвое увеличилось число элементов разметки, оформление документов все больше приближается к оформлению качественных печатных изданий, развиваются средства описания не текстовых информационных ресурсов и способы взаимодействия с прикладным программным обеспечением. Совершенствуется механизм разработки типовых стилей. Фактически, в настоящее время HTML развивается в сторону создания стандартного языка разработки интерфейсов как локальных, так и распределенных систем.

3.3.3. Задания для самостоятельного решения

Задание 1

Создать HTML-документ, содержащий следующую структуру: название странички; заголовки нескольких уровней, выровненные по центру, левому или

правому полю; параграфы с текстом (с использованием различных стилей); списки (нумерованные, маркированные, вложенные).

Задание 2

Создать HTML-документ с использование таблиц. Установить в таблице фоновый рисунок.

Задание 3

Создать HTML-документ и разместить на нем текстовое поле, командную кнопку и переключатель. Использовать в качестве фона графический файл.

Задание 4

Создать HTML-документ, включающий бегущую строку, фоновый звук, и другие динамические объекты

Задание 5

Использовать две гиперссылки для связи какого-либо слова из первого документа со вторым документом и графического файла из третьего документа с первым документом.

Задание 6

Создать базовую страницу, на которой будут отображаться ранее созданные документы и рамка-фрейм, содержащая графическое изображение. Структура и размеры фреймов задаются в процентах.

Контрольные вопросы

1. В чем заключается гипертекстовая технология?
2. В чем состоит основное отличие текста от гипертекста?
3. Что является разметкой в HTML- документах?
4. Перечислите основные достоинства языка HTML.
5. Назначение и области использования технологии frames.
6. Как описывается ссылка на другой документ?

3.4. Занятие 4. Проектирование элементов пользовательского интерфейса (2 часа)

3.4.1. Цель занятий

- ознакомление с синтаксисом и типом данных PHP;
- приобретение навыков создания PHP-сценариев;
- получение навыков создания сессий в PHP.

3.4.2. Методические указания по теме

Для PHP [2-4] существует *четыре способа отделения* его от общего кода HTML (имеется ввиду для интерпретации):

1. <? echo ("SGML инструкции\n"); ?>
2. <?php echo("XML документ\n"); ?>
3. <script language="php">
   echo ("специально для FrontPage");

```
</script>
```

4.
```
<% echo ("ASP-стиль"); %>
<%= $variable; # Комментарий "<%echo .." %>
```

Инструкции в PHP отделяются друг от друга точкой с запятой (перед закрывающим тегом (?>) точку с запятой ставит не обязательно).

```
<?php echo "This is a test";
 echo "This is a test also" ?>
```

PHP поддерживает комментарии в стиле 'C', 'C++' и Unix shell. Например:

```
<?php
echo "test"; // Комментарий в стиле C++
/* Это многострочный
комментарий     в стиле C++*/
 echo "test2";
 echo "Test3"; # Это unix-shell комментарий
?>
```

PHP поддерживает следующие типы данных: integer (целочисленные); floating-point numbers или double (числа с плавающей запятой); string (строки, текст); array (массивы); object (обьекты). Преобразование типов происходит следующим образом. Если переменной присваивается строка, то эта переменная становится строковой. Если же с ней совершается одна из многих математических функций или она приравнивается к численной переменной, она становится численной.

Пример:

```
$foo = "0";            // $foo строка (ASCII 48)
$foo++;                // $foo тоже строка "1" (ASCII 49)
$foo += 1;             // $foo теперь integer (2)
$foo = $foo +2.3; // $foo теперь double (2.3)
$foo = 2 + "10 BC"; // $foo теперь снова integer (12)
$foo =2 + "10 BC"; // $foo и по прежнему integer (12)
```

В скобках указано результирующее значение переменной.

Чтобы определить тип переменной, можно воспользоваться функциями gettype(), is_long(), is_double(), is_string(), is_array() и is_object().

Область видимости переменных

Существуют границы определения переменных, чтобы использовать глобальные переменные в функциях необходимо их сначала декларировать как глобальные.

Рассмотрим пример:

```
$a = 1;        /* глобальное определение */
Function OBL () {
$a=2;
echo $a;       /* локальная переменная */
}
```

```
OBL ();
echo $a;
```

Таким образом, в функции используется локальная, собственная переменная. В результате выполнения данной программы будут выведены числа 2 и 1.

Пример использования в функции глобальной переменной (декларирование с помощью оператора global):

```
$a = 1; /* глобальное определение */
Function OBL ()
{
global $a;
$a=2;
echo $a;      /* локальная переменная */
}
OBL ();
echo $a;
```

Результатом выполнения программы будет вывод числа 2 и 2.

Пример предыдущей программы, реализованный через ассоциативный массив $GLOBALS[]:

```
$a = 1;         /* глобальное определение */
Function OBL ()
{
$GLOBALS["a"]=2;
echo $a;       /* локальная переменная */
}
OBL ();
echo $a;
```

Внимание: Переменная в массиве указывается без символа $.

Функции

В языке PHP существует множество встроенных функций [2,3]. Рассмотрим ряд из них:

1) trim() – удаление пробельных символов из начала и конца строки, например:

```
<?php
$str = ‘’   удаление пробелов   ’’;
$new_str = trim($str);
?>
```

2) ereg() – функция может содержать параметры: строка и множество, где осуществляется поиск:

```
<?php
if ( ereg (”home”, “state”));
{
echo ‘’Строка соответствует шаблону‘’
}
```

```
else
{
echo ‘’Строка  не соответствует шаблону‘’
}

?>
```

3) eregi() – эта функция идентична ereg(), за исключением того, что она игнорирует различия в регистре символов алфавита;

Пример задания:

Написать программу, подсчитывающую количество посетителей на сайте. Время нахождения пользователя на сайте – 7 минут, далее считать, что пользователь покинул сайт.

Решение:

```
<?php
session_start();
$id = session_id();
$currentTime = time();
$oldTime = time() - 420; // время на 7 минут раньше
$mas = @file("n.txt");
$k = 0;
for ($i = 0; $i < sizeof($mas); $i++)
{
 $line = explode("/", $mas[$i]);
 if ($line[1] > $oldTime)
        {
         $new_mas[$k] = $mas[$i];
                $k++;
         }
}
if(!isset($new_mas))$new_mas = null;
for ($i = 0; $i<sizeof($new_mas); $i++)
{
$line = explode("/", $new_mas[$i]);
 if ($line[0]==$id)
 {
  $line[1] = trim($currentTime)."\n";
        $is_id = true;
}
$line = implode("/", $line);
 $new_mas[$i] = $line;
}
$fp = @fopen("n.txt", "w");
for ($i = 0; $i<sizeof($new_mas); $i++)
```

```
{
fputs($fp, $new_mas[$i]);
}
fclose($fp);
if(!isset($is_id))$is_id = false;
if (!$is_id)
{
 $fp = @fopen("n.txt", "a");
 $line = $id."/".$currentTime."\n";
fputs($fp, $line);
fclose($fp);
}
$mes = file("n.txt");
echo "Сейчас на сайте посетителей -  ",sizeof($mes);
?>
```

Задание 2.

Разработать программу и сценарий форума с использованием PHP-кода, встроенного в HTML-код, позволяющую любому пользователю:

3.4.3. Задания для самостоятельного решения

Задание 1.

Написать PHP-код для файла, выполняющего отправку сообщений.

Видимая часть формы включает озаглавленное поле многострочного ввода и две управляющие кнопки для очистки поля ввода и отправки сообщений. Для управления PHP- сценарием ввести в форму скрытое поле с именем yes. Файлы с текущим количеством сообщений и текстом этих сообщений должны создаваться динамически. Использовать функцию fputs().

Задание 2.

Написать PHP-код для файла, выполняющего чтение сообщений и одновременное удаление с сервера. Если файлы-сообщения отсутствуют, пользователь должен быть проинформирован. Если есть – должен создаваться массив строчек файла и построчно выводить на экран.

Использовать функции file_exists() и unlink().

Задание 3.

Написать PHP-код для файла, выполняющего проверку корректности введенного пользователем адреса электронной почты.

Задание 4

Написать PHP-код, удаляющий пробельные символы из начала и конца строки, используя функцию trim().

Задание 5.

Написать PHP-код для подсчета общего количества посетителей сайта.

Контрольные вопросы

1. Что такое PHP, его возможности?
2. Поясните способы отделения PHP-кода от общего кода HTML.
3. Сформулируйте правила для выбора имени переменной в PHP.
4. Поясните способы ввода данных пользователем.
5. Поясните алгоритм проверки открытия файла PHP-сценарием.
6. Поясните работу функции создания сессии.

3.5. Занятия 5-6. Практические аспекты создания Web-приложений (4 часа)

3.5.1. Цель занятий

- ознакомление со средствами и инструментами создания Web-приложений на примере спецификации .Net;
- получение навыков создания управляемых приложений.

3.5.2. Методические указания по теме

По данной теме рабочей программой по дисциплине «Информатика» предусмотрено два 2-х часовых занятия:

1) средства и инструменты создания Web-приложений;
2) создание управляемых приложений.

Серверные элементы управления являются составляющими пользовательского интерфейса Web-формы. Выделяется четыре типа серверных элементов управления: серверные элементы управления HTML, элементы Web, средства подтверждения, пользовательские элементы.

Элементы управления HTML представляют собой обычные элементы формы HTML, такие как текстовые поля ввода и кнопки, но создаются они на сервере, где ими можно управлять. Аналогичны им элементы Web, но они более функциональны и могут формировать сложный пользовательский интерфейс. Средства подтверждения используются для проверки правильности пользовательского ввода данных. Пользовательские элементы специально предназначены для реализации некоторой особой функциональности.

Все элементы управления, размещенные на сервере, имеют свойства, методы и события. Они намного функциональнее, чем традиционные элементы формы HTML, и существенно упрощают процесс формирования пользовательского интерфейса.

При создании серверного элемента не нужно заботиться о написании кода HTML. При поступлении запроса на страницу элемент управления автоматически генерирует HTML, корректно отображаемый в браузере. Например, следующая строка создает сервере элемент управления Button:

<asp:Button text="Submit" runat="server"/>

В момент ответа на запрос клиенту выдается такой код HTML:

<input type="Submit" name="ctrll" value="Submit">

Эти две строки лишь похожи друг на друга. Первая выполняется только на сервере - клиент никогда не получит ее. (Даже если бы и получил, то не знал бы, что ней делать - браузер понимает только код HTML.) Вторая строка - это то, что принимает клиент.

Также .NET знает о возможностях каждого браузера и, следовательно, отсылает соответствующий код каждому из них. Например, если браузер и поддерживает динамический HTML (DHTML), то .NET не будет отправлять ему такой код. Этот подход называется низкоуровневой поддержкой связи с тем, что .NET может сглаживать вывод кода HTML для браузеров, которые не поддерживают современный уровень функциональности. В идеале, низкоуровневая поддержка должна отображать корректно i-ый элемент управления.

Для более детального изучения средств, инструментов и спецификации .NET необходимо ознакомиться с документацией [5].

Пример задания:

Написать программу для создания серверных элементов управления и демонстрации динамического изменения их свойств.

Решение.

```
<%@Import Namespace="System.Web.UI.WebControls " %>
<html>
<head>
  <title>Dynamic Table</title>
<script language="VB" runat="server">
 Sub SubmitButton_Click(Source As Object, e As EventArgs)
   Table1.Bgcolor = Select1.Value
   Table1.Border = Select2.Value
   Table1.Cellpadding = Select3.Value
   Table1.Cellspacing = Select4.Value
     If Select5.Value = "1" Then
     Tr1.Bgcolor = Select6.Value
   Else If Select5.Value = "2" Then
     Tr2.Bgcolor = Select6.Value
   Else If Select5.Value = "3" Then
     Tr3.Bgcolor = Select6.Value
   End If
  If radio1.Checked = True Then
    Labet1.Text=Text1.Value
  Else If radio2.Checked = True Then
    Labet1.Text= " "
  End If
   End Sub
</script>
```

```
</head>
  <body>
<form id="Form1" method="post" runat="server">
 <center>
<h3><asp:Label runat="server" id="Labet1">Table Header</asp:label></h3>
<br>
<table runat="server" id="table1" cellspacing=3 cellpadding=3 border=1>
   <tr runat="server" id="Tr1">
     <td>One</td>
     <td>NO_One</td>
   </tr>
   <tr runat="server" id="Tr2">
     <td>Two</td>
     <td>NO_Two</td>
   </tr>
   <tr runat="server" id="Tr3">
     <td>Three</td>
     <td>NO_Three</td>
   </tr>
 </table>
<p>
</center>
<br>
<br>
<br>
Table Header:
<Input type="text" runat="server" id="Text1" value="Table Header"></Input>
<br>
Table Background Color:
<select id="Select1" runat="server">
 <option value="#ffffff">Белый</option>
 <option value="#ff0000">Красный</option>
 <option value="#C1C1C1">Серый</option>
</select>
<br>
Table Border:
<select id="Select2" runat="server">
 <option value="0">0</option>
 <option value="1">1</option>
 <option value="2">2</option>
 <option value="3">3</option>
 <option value="4">4</option>
 <option value="5">5</option>
```

```
</select>
<br>
Table Cellpadding:
<select id="Select3" runat="server">
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
</select>
<br>
Table Cellspacing:
<select id="Select4" runat="server">
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
</select>
<br>
<p>
Select a Row:
<select id="Select5" runat="server">
  <option>1</option>
  <option>2</option>
  <option>3</option>
</select>
<br>
Row Color:
<select id="Select6" runat="server">
  <option>White</option>
  <option>Yellow</option>
  <option>Red</option>
  <option>Blue</option>
</select>
<br>
Show Table Header:
<input type ="radio" runat="server" id="radio1" name="radio1"
checked="true">Yes
<input type ="radio" runat="server" id="radio2" name="radio1">No
<br>
```

```
<center>
<input type="submit" runat="server" id="submit1"
value="Submit" onserverclick="SubmitButton_Click">
</center>
</form>
</body>
</html>
```

3.5.3. Задания для самостоятельного решения

Задание 1.

Написать программу для создания серверных элементов управления в форме средств подтверждения.

Задание 2.

Написать программу для создания серверных элементов управления в форме пользовательских элементов.

Задание 3.

Имеется файл в корневом каталоге Web- сервера с установленным атрибутом – только для чтения. Написать PHP-код, выполняющий проверку на запись и чтение с помощью функций is_writable() и is_readtable() соответственно.

Задание 4.

Написать фрагмент программы, импортирующей два пространства имен, содержащих компоненты работы с данными. Предусмотреть возможность работы с базой данных, работающей, например, под управлением СУБД Microsoft SQL Server.

Контрольные вопросы

1. Что понимается под Web- приложением?
2. Перечислите основные компоненты базовой технологии.
3. Перечислите основные типы серверных элементов управления
4. Чем отличается процесс компиляции в среде .NET Framework?
5. Перечислите основные свойства серверных элементов управления.

Литература

1. Романчева Н.И. ОС Linux- принципы, технологии, инструменты: учебное пособие. - М.: МГТУ ГА, 2011.- 80 с.

2. Мазуркевич А., Еловой Д. PHP: настольная книга программиста. - Мн.: Новое знание, 2003. - 480 с.

3. Котеров Д.В. Самоучитель PHP 4.- СПб.: БХВ-Петербург, 2001. - 576 с.

4) http://microsoft.com/php

5) http:// www.microsoft.com/net