УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
«16» 10 2018 г.

БАКАЛАВРИАТ

Кафедра «Компьютерные информационные системы и управление»

Учебно-методический комплекс дисциплины

Разработка клиент серверных приложений

Направление: **710100 «Информатика и вычислительная техника»**

Профиль: **Компьютерные информационные системы в бизнесе**

Академическая степень **- бакалавр**

Форма обучения **(очная)**

График проведения модулей 7-семестр

неделя	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
лекц. зан.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
прак./лаб. зан.	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

«РАССМОТРЕНО»
Протокол заседания кафедры
«КИСиУ»
№ 2 от 16.10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.

«СОГЛАСОВАНО»
Проректор по академ. вопросам
проф. Мадалиев М.М.

Составитель Маджинов А.Р.

Директор Научной библиотеки Асанова Ж.Ш.

БИШКЕК 2018

Оглавление

АННОТАЦИЯ

Специальный курс «Разработка клиент-серверных приложений» изучается студентами 4 курса, обучающихся по направлению 710100 «Информатика и ВТ», специализаций «Компьютерные информационные системы для бизнеса» и «Международные компьютерные сети и офисные системы».

Курс предусматривает лекционные занятия, лабораторный практикум, а также самостоятельную работу студентов. По итогам изучения дисциплины студенты сдают зачет. Текущий контроль и самоконтроль усвоения курса осуществляется посредством выполнения студентами лабораторных работ и сдачи модулей.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)
1. Пояснительная записка

1.1. Миссия и стратегия

Миссия НОУ УНПК "МУК" – подготовка международно - признанных, свободно мыслящих специалистов, открытых для перемен и способных трансформировать знания в ценности на благо развития общества. Видение НОУ УНПК «МУК»- создание динамичного и креативного университета с инновационными научно-образовательными программами и с современной инфраструктурой, способствующие достижению академических и профессиональных целей.

Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

1.2. Цель и задачи дисциплины (модулей)

Цель дисциплины «PHP» позволяет освоить будущим специалистам инженерам теоретические знания и сформировать у них практические навыки в применении баз данных для создания, обработки и хранения больших объемов информации при решении различных прикладных задач.

Задачи дисциплины:

- создание у студентов упорядоченной системы знаний по проектированию баз данных, управлению и администрированию базами данных, основам структурированного языка, о методах сжатия больших информационных массивов, о реальных возможностях PHP;
- ознакомление студентов с практикой создания информационной модели данных для конкретной предметной области и применения СУБД для создания приложений баз данных.

Дисциплина «Базы данных» относится к циклу ОПД.Ф.02 Цикл общепрофессиональных дисциплин. Федеральный компонент

1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине (модулю) (знания, умения владения), сформулированные в компетентностном формате.

Дисциплина (модуль) направлена на формирование следующих компетенций:

- общенаучными (ОК-4):
  - способен понимать и применять традиционные и инновационные идеи, находить подходы к их реализации и участвовать в работе над проектами, используя базовые методы исследовательской деятельности (ОК-4);

- инструментальными (ИК):
  - владение основными методами, способами и средствами получения, хранения и переработки информации, навыками работы с компьютером, как средством управления информацией, в том числе в глобальных компьютерных сетях и корпоративных информационных системах (ИК-5);
- профессиональными (ПК):
  - способен разрабатывать бизнес-планы и технические задания на оснащение отделов, лабораторий, офисов компьютерным и сетевым оборудованием (ПК-1);

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1. Знать: ИК-5, ПК-1, ОК-4.

2. Уметь: ИК-5, ПК-1, ОК-4.
3. Владеть: ПК-1, ОК-4.

1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина (модуль) «Разработка клиент-серверных приложений» является частью цикла (блока) дисциплин учебного плана по направлению подготовки 710100 «Информатика и вычислительная техника», специальности Компьютерные информационные системы в бизнесе. Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: Основы программирования, Математическая логика и теория алгоритмов.

2. Структура дисциплины (модулей)

Структура дисциплины (модулей) для очной формы обучения

Общая трудоемкость дисциплины составляет 6 кредита, 180 ч., в том числе аудиторная работа обучающихся с преподавателем 90 ч., самостоятельная работа обучающихся 90 ч.

№ п/п	*Раздел, Темы Дисциплины*	*Семестр*	*Неделя семестра*	*Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)*				*Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)*
				лекции	*Сем. Заня/лаб. заня*	*СРС*	*СРСиП*	
	Раздел 1.							
1	Архитектура сети: одно ранговая и клиент/сервер Протоколы Интернета Сетевое взаимодействие. Семи уровневая модель взаимодействия открытых систем OSI	7	1	2	4	4	3	опрос, проверка задания, посещаемость
2	Установка, настройка программного обеспечения Основы языка PHP. Простые и составные операторы. Комментарии, переменные, Типы данных. Операторы вывода информации Логические операторы. Операторы сравнения	7	2	2	4	4	3	опрос, проверка задания, посещаемость
3	Однострочные поля ввода	7	3	2	4	4	3	опрос, проверка

	Поля ввода пароля Независимые переключатели							задания, посещаемость
4	Установка, настройка программного обеспечения Основы языка PHP. Простые и составные операторы. Комментарии, переменные, Типы данных. Операторы вывода информации	7	4	2	4	4	3	опрос, проверка задания, посещаемость
5	Логические операторы. Операторы сравнения Задачи на разветвления Работа с циклами .	7	5	2	4	4	3	Модуль 1
	Раздел 2.							
6	Язык HTML Формы в HTML документах.	7	6	2	4	4	3	опрос, проверка задания, посещаемость
7	Однострочные поля ввода Поля ввода пароля Независимые переключатели	7	7	2	4	4	3	опрос, проверка задания, посещаемость
8	Зависимые переключатели Списки выбора. С единственным выбором. Списки с множественным выбором. Ввод многострочного текста	7	8	2	4	4	3	Модуль 2
9	Списки с множественным выбором. Формы в HTML документах. Математические функции языка PHP Операторы циклов Операторы разветвления	7	9	2	4	4	2	опрос, проверка задания, посещаемость
10	Операции над массивами. Удаление массивов. Слияние массивов. Перебор массивов. Сортировка массивов. Получение части массива. Добавление и удаление элементов массива Работа с многомерным массивом	7	10	2	4	4	2	опрос, проверка задания, посещаемость
	Раздел 3.							
11	Понятие массива и списка. Ассоциативные массивы Операции над массивами. Удаление массивов. Слияние массивов. Перебор массивов. Сортировка массивов. Получение части массива. Добавление и удаление элементов массива	7	11	2	4	4	2	Модуль 3

12	Строковые переменные. Функции поиска в тексте Функции форматного вывода Функции замены в строке Изменение регистров	7	12	2	4	3	1	опрос, проверка задания, посещаемость
13	Распределенные БД. 1. Модели "клиент-сервер" в технологии баз данных 2. Двухуровневые модели 3. Модель сервера приложений 4. Модели серверов баз данных	7	13	2	4	3	1	опрос, проверка задания, посещаемость
14	Преобразование символов Функции удаления пробелов Сравнение строк.	7	14	2	4	3	1	Модуль 4
15	Понятие массива и списка. Ассоциативные массивы	7	15	2	4	3	1	консультация

3. Содержание дисциплины (модулей)

№	Наименование раздела, темы дисциплины	Краткое содержание
1	Основные понятия	«Введение в PHP», «Установка и настройка необходимого ПО»; «Основной синтаксис»; «Переменные, константы и операторы»; «Типы данных. «Управляющие конструкции в PHP» «Условные операторы»; «Циклы»; «Операторы передачи управления»; «Операторы включения»; тест и задание по пройденным темам. «Работа со строками в PHP» «Строки»; «Выделение и работа с подстроками»; «Разделение и соединение строки»; «Строки, содержащие html-код» «Работа с массивами в PHP» «Массивы и операции с ними»
2	Модели данных	«Сортировка массивов»; «Применение функции ко всем элементам, выделение подмассива, сумма элементов массива»; «Функции в PHP» «Функции, определяемые пользователем»; «Аргументы функции» «Использование переменных внутри функции. Возвращаемые значения»; «Переменные функции. Встроенные функции» «Работа с файлами в PHP». «Создание файла. Закрытие соединения с файлом»; «Запись и чтение данных из файла»; «Проверка существования и удаление файла»; «Загрузка файла на сервер»; теста и задания. Седьмой блок «Объектно-ориентированное программирование в PHP»
3	Язык SQL	«Базы данных: основные понятия»; «СУБД MySQL»; «Язык SQL»; «Отображение интерфейса для построения информации»; «Отображение данных, хранящихся в MySQL»; тест и задание. «Регулярные выражения в PHP» «Понятие регулярного выражения»; «Синтаксис регулярных выражений»; «Модификаторы PCRE». «Авторизация доступа в PHP с помощью сессий»

4. Конспект лекций

Лекция №1 содержит историю языка, описание его возможностей, областей применения, способов использования. Рассматривается процесс установки и настройки программного обеспечения, необходимого для работы с PHP.

Лекция №2 посвящена изучению основ синтаксиса PHP. Рассматриваются способы разделения инструкций, создания комментариев, переменные, константы и типы данных, операторы.

Лекция №3 рассматривает условные операторы, организацию с циклов, использование возможностей компоновки проекта.

Лекция №4 посвящена изучению способов отправки данных на сервер и их обработки. Рассматриваются основные понятия клиент-серверных технологий, HTML формы и отправка данных с их помощью, механизм получения данных из HTML форм и их обработка средствами PHP.

Лекция №5 рассматривает понятия функции, функции, определяемых пользователем, аргументы функций, передача аргументов по значению и по ссылке, значения аргументов по умолчанию и значения, возвращаемые функцией.

Лекция №6 посвящена ООП. Рассматриваются понятия классов и объектов, определение и использование классов, возможности расширения классов.

Лекция №7 посвящена более подробному изучению массивов и функций, встроенных в PHP для работы с ними.

Рассматриваются встроенные функции для работы с массивами, а также использование функций, определяемых пользователем для работы с элементами массива, разбивка массива на вложенные массивы и многое другое.

Лекция №8 подробно обсуждает вопросы обработки строк. Изучаются функции, полезные для решения разнообразных прикладных задач. Рассматриваются различные способы вывода строк, разбивка и соединение строк, определение длины строки, выделения подстрок.

Лекция №9 обсуждает вопросы, связанные с созданием файлов, чтением данных из файла, удаление файла, а также проверка наличия файла на сервере.

Лекция №10 рассматривает понятия базы данных и СУБД, основы языка запросов SQL: операции выбора, добавления, изменения и удаления строки, а также операции создания, изменения и удаления таблицы.

Лекция №11 предназначена для знакомства со способами взаимодействия PHP и СУБД MySQL. Основное внимание уделяется установке соединения с базой данных, функциям отправки запросов и обработки ответов.

Лекция №12 посвящена изучению вопросов обеспечения безопасности в сети и использования для этих целей механизма сессий. Рассматриваются следующие вопросы: инициализация сессий, передача идентификатора пользователя, регистрация переменных сессии, уничтожение сессии.

Лекция №13 рассматривает понятие регулярного выражения, реализацию механизма регулярных выражений в языке PHP, их синтаксис и семантику.
Лекция №14 знакомит читателя с понятием объектной модели XML документа и ее использованием в PHP, обработки элементов XML документа с помощью функций PHP (получение значений узлов, атрибутов и многое другое).
Лекция №15 рассматривает понятие шаблона и его использование в языке программирования PHP, а также классы шаблонов FastTemplate и Smarty.

5. Информационные и образовательные технологии

Информационные и образовательные технологии

№ п/п	Наименование раздела	Виды учебной работы	Формируемые компетенции (указывается код компетенции)	Информационные и образовательные технологии
1	Раздел №1.	Лекция	ОК-4, ИК-5, ПК-1	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ОК-4, ИК-5, ПК-1	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ОК-4, ИК-5, ПК-1	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты
2	Раздел №2.	Лекция	ОК-4, ИК-5, ПК-1	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ОК-4, ИК-5, ПК-1	Дискуссия, Консультирование с разбором

				абстрактных ситуаций
		Самостоятельная работа	ОК-4, ИК-5, ПК-1	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты
3	Раздел №3.	Лекция	ОК-4, ИК-5, ПК-1	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ОК-4, ИК-5, ПК-1	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ПК-4, ПК-5, ПК-11	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты

6. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

№ п/п	Контролируемые разделы дисциплины (модулей)	Код контролируемой компетенции (компетенций)	Наименование оценочного средства
1	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
2	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
3	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
4	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
5	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение

			лабораторных работ
6	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
7	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
8	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ
9	Разделы №1, №2, №3	ОК-4, ИК-5, ПК-1	опрос, выполнение лабораторных работ

6.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	1, 2, 3, 4, 5 недели	8 баллов	До 40 баллов
- выполнение лабораторных работ	1, 2, 3, 4, 5 недели	6 баллов	До 30 баллов
- посещаемость	1, 2, 3, 4, 5 недели	0,2	10 баллов
Рубежный контроль: (сдача модуля)	5 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	6, 7, 8, 9, 10 недели	8 баллов	До 40 баллов
- выполнение лабораторных работ	6, 7, 8, 9, 10 недели	6 баллов	До 30 баллов
- посещаемость	6, 7, 8, 9, 10 недели	0,2	10 баллов
Рубежный контроль: (сдача модуля)	10 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	11, 12, 13, 14, 15 недели	8 баллов	До 40 баллов

- выполнение лабораторных работ - посещаемость	11, 12, 13, 14, 15 недели 11, 12, 13, 14, 15 недели	6 баллов 0,2	До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	15 неделя	100%×0,2=20 баллов	
Итого за III модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	**Оценка экзамена**
От 0 до 54	неудовлетворительно
от 55 до 69 включительно	удовлетворительно
от 70 до 84 включительно	хорошо
от 85 до 100	отлично

6.3. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Текущий контроль (0 - 80 баллов)

При оценивании посещаемости, опроса и выполнении лабораторных работ учитываются:

- посещаемость (10 баллов)
- степень раскрытия содержания материала (25 баллов);
- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (20 баллов);
- знание теории изученных вопросов, сформированность и устойчивость используемых при ответе умений и навыков (25 баллов).

Рубежный контроль (0 – 20 баллов)

При оценивании контрольной работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 10 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 5 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность – 5 баллов.

Итоговый контроль (экзаменационная сессия) – ИК = Бср × 0,8 + Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (0 баллов);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (10 баллов);

- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (20 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (30 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (0-9 баллов);
- ответ содержит 21-89 % правильного решения (10-39 баллов);
- ответ содержит 90% и более правильного решения (40 баллов).

6.4. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности..

Перечень вопросов:

1. Эволюция концепций обработки данных и развитие технологий обработки данных.
2. Гипертекстовые базы данных.
3. Мультимедийные базы данных.
4. Распределенная обработка данных.
5. Доступ к данным с помощью ADO.
6. Доступ к данным с использованием ODBC.
7. Интерфейс к базам данных на платформе Java.
8. Корпоративные серверы приложений. Corba-технология.
9. Коммерческие БД.
10. Объектно-ориентированные БД.
11. XML-серверы.
12. Публикация БД с использованием XML.
13. Базы данных и Интернет.
14. Примеры организации данных фактографических БД.
15. Примеры организации данных документальных БД.
16. Персональные (настольные) СУБД.
17. Промышленные СУБД.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1.Список источников и литературы

- Литература:
  - Основная:

- Балдин К. В., Уткин В. Б. Информационные системы в экономике. Учебник / К.В. Балдин. - М.: Дашков и Ко, 2012. – 395с.
- Илюшечкин, В.М. Основы использования и проектирования баз данных: Учебное пособие / В.М. Илюшечкин. - М.: Юрайт, 2011 - 213с.
- Советов Б. Я. , Цехановский В. В. , Чертовской В. Д. Базы данных: теория и практика: учебник для бакалавров /Б.Я. Советов. - М.: ЮРАЙТ, 2011. - 459 с.
- Маркин А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин.- М.: Диалог-МИФИ, 2008. – 318 с.
- Мишенин А. И. Теория экономических информационных систем: Учебник 4-е изд., доп. и перераб. / А.И. Мишенин. - М.: Финансы и статистика, 2008. - 240 с.

  - Дополнительная:

- Баженова, И.Ю. Основы проектирования приложений баз данных: Учебное пособие / И.Ю. Баженова. - М.: Интернет-Университет Информационных

Технологий, 2009 - 325с.
- Бекаревич, Ю.Б. Microsoft Access за 21 занятие студента / Ю.Б. Бекаревич, Н.В. Пушкина. - СПб.: БХВ - Петербург, 2005 - 544с.
- Бумфрей Ф., Диренцо О., Даккетт Й. XML. Новые перспективы WWW. – Издательство «ДМКПресс», 2006 – 688 с.
- Гайдамакин, Н.А. Автоматизированные информационные системы, базы и банки данных: Вводный курс / Н.А. Гайдамакин. – М.: Гелиос АРВ, 2002 – 368с.
- Голицына, О.Л. Системы управления базами данных: Учебное пособие / О.Л. Голицына. - М.: ФОРУМ-ИНФРА-М, 2006 - 432с.
- Гринвальд, Р. ORACLE: Справочник / Р. Гринвальд, Д. Крейнс. - СПб: Символ-Плюс, 2005 - 976с.
- Дейт, К.Дж. Введение в системы баз данных. : Пер.с англ. / К.Дж. Дейт. - Киев; М.; СПб.: Вильямс, 1999 - 848с.
- Керман, М.К. Программирование и откладка в Delphi: Учебный курс / М.К. Керман. – Киев; М.; СПб.: Вильямс, 2002 – 672с.
- Кириллов, В.В. Введение в реляционные базы данных / В.В. Кириллов, Г.Ю. Громов. - СПб: БХВ - Петербург, 2009 - 464с.
- Мезенцев, К.Н. Автоматизированные информационные системы: учебник / К.Н. Мезенцев. – М.: Академия, 2010 – 176 с.
- Мусина, Т.В. Visual FoxPro 9.0:Учебный курс / Т.В. Мусина. - Киев: Век+;СПб:Корона-Век, 2011 - 736с.
- Смирнов, С.Н. Обработка документов средствами Oracle: Практикум по XML и JDBC /С.Н. Смирнов. – М.: Гелиос АРВ, 2004 – 192 с.
  ◦

7.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

1. Википедия - http://ru.wikipedia.org
2. Интернет-портал образовательных ресурсов по ИТ - http://www.intuit.ru
3. Портал математических интернет-ресурсов - http://www.math.ru/
4. Портал математических интернет-ресурсов - http://www.allmath.com/
5. Портал ресурсов по математике, алгоритмике и ИТ - http://algolist.manual.ru/
6. http://www.iprbookshop.ru/
7. http://kyrlibnet.kg/ru/
8. http://biblioteka.kg/

8. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся.

8.1. Методические указания для обучающихся по освоению дисциплины (модулей)

Вид работы	Содержание (перечень вопросов)	Трудоемкость самостоятельной работы (в часах)	Рекомендации
Модуль №1.			
Конспектирование материала на лекционных занятиях	Стратегии разработки программных средств и систем и	5	[1, 2]

Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	реализующие их модели жизненного цикла		
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Выбор модели жизненного цикла для конкретного проекта	5	[1, 2]
Итого		10	
Модуль №2.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Классические методологии разработки программных средств	8	[1, 2, 3]
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Case-технологии технологии структурного анализа и проектирования программных средств	8	[1, 2, 4]
Итого		16	
Модуль №3.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ.	Методология объектно-технологии ориентированного анализа и проектирования сложных систем	8	[1, 2, 5]

Работа с учебной литературой. Написание программы			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Инструментальные средства разработки программного обеспечения	8	[1, 2, 5]
Итого		16	
Итого по дисциплине		42	
Вид работы	Содержание (перечень вопросов)	Трудоемкость самостоятельной работы (в часах)	Рекомендации
Раздел №1.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой.		2	

Написание программы			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Итого		10	
Раздел №2.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых		3	

и контрольных работ. Работа с учебной литературой. Написание программы			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Итого		15	
Раздел №3.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях		3	

Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Закрепление пройденного курса		5	
Итого		17	
Итого по дисциплине		42	

8.2. Методические рекомендации по подготовке письменных работ

Разработанное программное обеспечение должно быть предоставлено в скомпилированом виде, а так же в виде текстового файла, содержащего исходный код программы.

8.3. Иные материалы

Не предусмотрено.

9. Материально-техническое обеспечение дисциплины

Для изучения дисциплины, необходимо следующее оборудование: ЭВМ, проектор.

Требования к аудитории: компьютерный класс, имеющий ЭВМ в количестве идентичном количеству обучающихся, ЭВМ для преподавателя с подключенным проектором, наличие доски и средств для отображения/удаления информации на доске (мел/ветошь, маркер/губка).

10. Глоссарий

11. Приложения.

Краткий конспект лекций.

Лекция 1 Архитектура сети: одноранговая и клиент/сервер

Одноранговые сети.

Существует две основные архитектуры сети: одноранговая (peer-to-peer) и клиент/сервер) (client/server), причем вторая практически вытеснила первую. В одноранговой сети все компьютеры равноправны – имеют один ранг. Поэтому любой компьютер может выступать как в роли сервера, то есть предоставлять свои ресурсы (файлы, принтеры) другому компьютеру, так и в роли клиента- использовать предоставленные ему ресурсы.

Одноранговые сети преимущественно распространены в домашних сетях или небольших офисах.

Достоинства одноранговых сетей:
1. Наиболее просты в установке и эксплуатации.
2. Операционные системы DOS и windows обладают всеми необходимыми функциями, позволяющими строить одноранговую сеть.
Недостатки:
В условиях одноранговых сетей затруднено решение вопросов защиты информации. Поэтому такой способ организации сети используется для сетей с небольшим количеством компьютеров и там, где вопрос защиты данных не является принципиальным.

Сети с выделенными серверами и одноранговые сети

Сети клиент/сервер.

В сети клиент/сервер существует один или несколько главных компьютеров-серверов. Все остальные компьютеры сети называются клиентами или рабочими станциями (workstations). Сервер – это специальный компьютер, который предоставляет определенные услуги другим компьютерам. Существуют различные виды серверов (в зависимости от предоставляемых ими услуг): серверы баз данных, файловые серверы, серверы печати, почтовые серверы, web-серверы. Для экономии средств, один сервер сочетает в себе несколько функций, например почтовый сервер, может быть также и WEB-сервером.
Для получения доступа к ресурсу в сети клиент/сервер пользователь должен ввести свой уникальный идентификатор –имя пользователя (login – логин) и пароль (password). Логин является общедоступной информацией. Проверка пользователя называется идентификацией. Подтверждение .(проверка подлинности) имени пользователя паролем-аутентификация. Идентификация_ аутентификация = авторизация.
Лекция 2 Протоколы Интернета
Протокол – это совокупность правил, определяющих взаимодействие абонентов вычислительной системы и описывающий способ выполнения определенного класса функций. Необходимость протоколов обуславливается тем, что в сети могут взаимодействовать компьютеры с самым разным программным обеспечением и самым разным аппаратным устройством. Что бы все компьютеры подключенные к сети могли понимать друг друга, необходимы общие наборы правил. Такими наборами правил являются протоколы.
Протокол TCP/IP (Протокол управления передачей/Интернет протокол) Именно на этом протоколе основана вся сеть Интернет. Он состоит из двух протоколов
TCP - является транспортным протоколом, который обеспечивает гарантированную передачу данных по сети.
IP – является адресным протоколом, который отвечает за адресацию всей сети. Благодаря использованию IP протокола, каждый компьютер в сети имеет свой индивидуальный

адрес (IP-адрес). Широко используемые в Интернете URL-адреса (www.rambler.ru. www.kg, www.ya.ru) являются лишь словесным обозначениями IP-адресов. Сделано это для удобства работы, поскольку человеку легче запоминать символьную информацию нежели числовую. Компьютер же работает с числовой информацией , поэтому при вводе URL-адреса в командную строку браузера , браузер автоматически преобразуется в IP-адрес. За сопоставление словесных URL-адресов и их IP- адресов отвечает специальная служба DNS. (Служба доменных имен).

Протокол (ICMP)- протокол межсетевых управляющих сообщений. С помощью этого протокола компьютеры и устройства в сети обмениваются друг с другом управляющей информацией. К примеру: протокол используется для передачи сообщений об ошибках, поверки доступности узла. И т.д.

Протокол FTP (File Transfer Protocol) – протокол передачи файлов. Служит для обмена файлами между компьютерами.

Протокол HTTP- (Hyper Text Transfer protocol) – протокол обмена гипертекстовой информацией. Т.е. документами HTML.

Протоколы POP и SMTP . Протокол POP – протокол почтового отделения. Этот протокол используется для получения электронной почты с почтовых серверов. А для передачи почты служит протокол SMTP (Simple ail Transfer protocol)- протокол передачи сообщений электронной почты.

Протокол IMAP- Служит также для чтения почты. Его отличие от POP, состоит в том, что пользователь читает свои сообщения электронной почты не загружая на свой компьютер. Все сообщения хранятся на сервере и удаляются с сервера.

Протокол SLIP (Serial Line Internet Protocol) – протокол подключения к сети Интернет по последовательной линии. Используется для установления связи с удаленными узлами через низкоскоростные последовательные интерфейсы. В настоящее время вытеснен протоколом PPP и практически не используется.

Протокол PPP (Point-to point Protokol)- обеспечивает управление конфигурацией, обнаружение ошибок, и повышенную безопасность при передаче данных на более высоком уровне, чем протокол SLIP.

Лекция 3 Сетевое взаимодействие. Семи уровневая модель взаимодействия открытых систем OS

Любая сеть связи работает с использованием определенного метода коммутации абонентов. Наиболее распространены три метода коммутации:

- коммутация каналов;
- Коммутация пакетов;
- Коммутация сообщений;

Для использования метода коммутации каналов нужен физический канал для прямой передачи данных между узлами. Коммутация каналов удобна для телефонных сетей. Для передачи компьютерного трафика разработан метод коммутации пакетов. При коммутации пакетов ваши данные разбиваются на части –пакеты. Пакеты передаются как независимые блоки, на другом компьютере переданные вами данные будут «собраны » воедино. Пакетная передача позволяет передача позволяет сбалансировать нагрузку на канал связи и обеспечить наиболее эффективное его использование. Коммутация сообщений в чистом виде уже не существует, поскольку в свое время они послужили прототипом сетей с коммутацией пакетов.

Составные части пакета. Пакет состоит из заголовка и поля данных. Заголовок содержит служебную информацию – адрес отправителя, адрес получателя, порт назначения. Поле данных содержит передаваемые вами данные.

Семиуровневая модель открытых систем OSI (Open System Interconnection)/

1. Физический уровень.
2. Канальный уровень
3. Сетевой уровень

4. Транспортный
5. Сеансовый
6. Представительный
7. Прикладной

Благодаря разбиению на уровни задача сетевого взаимодействия разбивается на ряд более мелких задач.

Пример:

Рассмотрим взаимодействие компьютеров более подробно на примере файловой службы. Допустим вам (компьютер 1) нужно записать какую-нибудь информацию в файл на удаленном (компьютере 2). Взаимодействие между компьютерами обычно осуществляется с помощью каких-либо программных приложений, обладающих специальным набором функций. Эти приложения работают на самом высоком уровне модели взаимодействия – прикладном. Поэтому, когда вы укажите, что хотите записать определенные данные в файл, будет сформировано соответствующее сообщение. В поле данных будет содержаться передаваемая в файл информация. После формирования сообщение будет передано с прикладного уровня на представительский уровень. На этом уровне в заголовок добавляются указания для представительского уровня компьютера адресата. Потом сообщение передается сеансовому уровню, который добавляет свою информацию. Процесс вложения одного протокола в другой называется *инкапсуляцией.* В процессе прохождения исходного блока данных по уровням он разбивается на более мелкие фрагменты для пересылки их по сети. Когда сообщение поступает на компьютер адресат, оно принимается физическим уровнем и передается вверх с уровня на уровень. Каждый уровень анализирует содержание заголовка своего уровня, выполняет содержащиеся в нем указания, затем удаляет относящуюся к нему информацию из заголовка и передает на следующий уровень. Это процесс называется декапсуляцией.

При взаимодействии открытой системы и Интернет модель OSI упрощается, так как некоторые протоколы Интернет включают в себя функции нескольких уровней.

Лекция 4 Адресация в сети Интернет

Любому компьютеру в IP-сети назначен уникальный адрес, который называется IP-адресом. IP-адрес –это 32 разрядное число, которое принято записывать в виде четырех чисел

111.111.213.232

127.0.0.1

При условии, что ваша сеть подключена к Интернету, протокол TCP/IP обеспечивает работу вашей сетевой программы с любым компьютером в мире, как будто он находиться в локальной сети. Уникальность IP – адреса достигается достаточно просто –IP-адреса назначаются централизованно **Сетевым Информационным Центром – (NIC**- Network Information Center**).**

Классы IP сетей. В общем случае IP –сети делятся на классы A,B,C,D, и E.

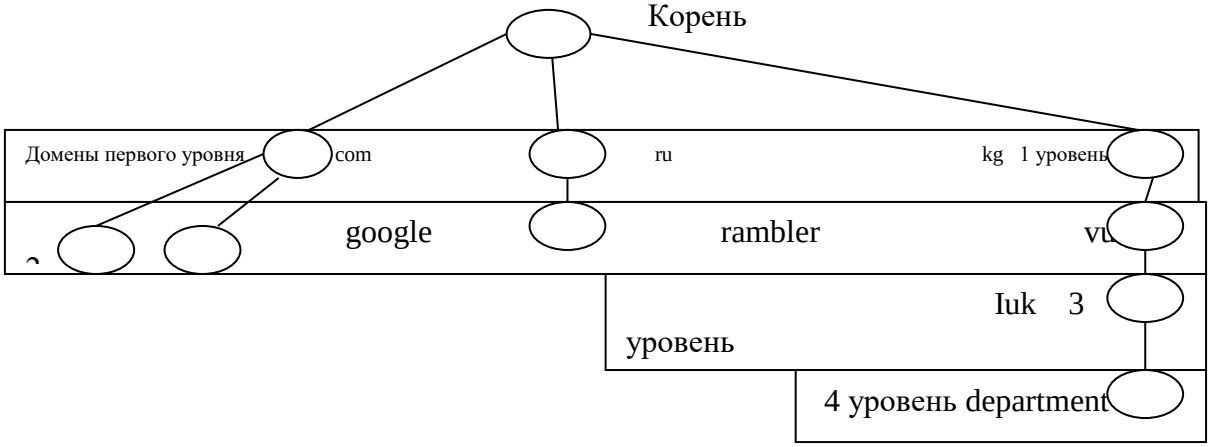
Лекция 5 Служба доменных имен

URL- (Universal Resource Locator)- это универсальный идентификатор ресурса.

Для того чтобы подключиться к какому-нибудь другому компьютеру, например, WEB-серверу, нужно знать его IP-адрес. Это не очень удобно, ведь для человека удобнее запоминать символьную информацию, а не набор цифр. Для преобразования IP- адреса в символьное имя и обратно используется служба доменных имен.- DNS (Domain Name System).

Рассмотрим пример. url –адрес: http:// department.iuk.vuz.kg/ Что происходит в момент когда пользователь вводит в адресную строку данный адрес. Сначала отправляется запрос на разрешение преобразования имени в IP- адрес серверу DNS, который принадлежит провайдеру пользователя. Ели такое имя есть в КЭШе DNS-сервера провайдера он возвращает IP-адрес и браузер устанавливает соединение с этим комьютером. Если такого адреса в КЭШе DNS не оказалось, DNS-сервер провайдера

обращается к серверу, который содержит домен наивысшего уровня, то есть к корню дерева kg. Сервер kg-dns обращается в свою очередь к серверу который делегирует домен vuz, Сервер vuz отправляет отправляется к серверу который администрирует домен iuk. А он уже к серверу который отвечает за домен department. Вот прослеживается своеобразная цепочка.



Как уже отмечалось, в заголовке пакета указывается IP-адрес отправителя и IP-адрес получателя, а также номер порта. Если при отправлении пакета в качестве адресата указывать только IP- адрес получателя, то приложения выполняемые на нем на смогут разобраться кому из них предназначены присланные данные. Что бы решить эту проблему используется механизм портов. Каждый порт имеет свой номер, каждой сетевой программе, которая работает по протоколу TCP/IP, сопоставляется свой номер пота. Например 80- это порт WWW-сервера (обычно это APACHE), 53 – это порт системы доменных имен.

Хостинг и хостинг-провайдеры

Организация, которая занимается созданием Internet-хостов (физических или виртуальных) с цель их продажи, называется хостинг-провайдером или хостером.

Лекция 6 Структура пакетов IP и TCP

TCP - является транспортным протоколом, который обеспечивает гарантированную передачу данных по сети.

IP – является адресным протоколом, который отвечает за адресацию всей сети. Благодаря использованию IP протокола, каждый компьютер в сети имеет свой индивидуальный адрес (IP-адрес). Широко используемые в Интернете URL-адреса (www.rambler.ru. www.kg, www.ya.ru) являются лишь словесным обозначениями IP-адресов. Сделано это для удобства работы, поскольку человеку легче запоминать символьную информацию нежели числовую. Компьютер же работает с числовой информацией , поэтому при вводе URL-адреса в командную строку браузера , браузер автоматически преобразуется в IP-адрес. За сопоставление словесных URL-адресов и их IP- адресов отвечает специальная служба DNS. (Служба доменных имен).

Лабораторные работа 1 Установка, настройка программного обеспечения

Для начала необходимо определиться с операционной системой. По большому счету все равно какую именно операционную систему Вы будите использовать, ведь PHP работает в Linux и Windows. Для того, чтобы работать на наших занятиях нам будет все равно какая операционная система установлена на нашем сервере, если же Вы собираетесь писать профессионально, то эти знания необходимы. Под Windows существует свой аналог PHP, который называется ASP (Active Server Pages). При этом в качестве HTTP-сервера вместо APACHE используется IIS (Internet Information Server)-

специальный продукт компании Microsoft. Для того чтобы изучить PHP нам необходимы следующие программы:
WEB-сервер APACHE- это программа которая превратит ваш домашний компьютер в WEB-сервер. PHP. Сервер баз данных MySQL.

WEB-сервер APACHE разработан и поддерживается организацией Apache project. Первоначально сервер Apache был разновидностью WEB-Сервера NCSA, разработанного в Национальном центре разработок суперкомпьютеров Иллинойского университета. Возникновение APACHE было связано с тем, что в 1994 году ушел из проекта главный разработчик NCSA, оставив многих последователей самостоятельно разбираться в своем сервере. Со временем начали появляться исправления и дополнения к серверу- так называемые «патчи». В апреле 1995 года вышла первая версия сервера APACHE. Позже APACHE стал самостоятельной разработкой. Сейчас сервер APACHE поддерживается группой – программистов-добровольцев APACHE-GROUP.
Сервер APACHE был разработан под ОС LINUX и UNIX, но позже были разработаны версии и для ОС WINDOWS.
Вспомним, что функции WEB-сервера выполняет не компьютер, а программа, установленная на нем, так вот APACHE и есть та самая программа. APACHE- это своего рода стандарт WEB-сервера в Интернете.
Джентльменский набор Web-разработчика («Д.н.w.p», читается «Денвер») — набор дистрибутивов (Apache, PHP, MySQL, Perl и т.д.) и программная оболочка, используемые Web-разработчиками для разработки сайтов на «домашней» (локальной) Windows-машине без необходимости выхода в Интернет. Главная особенность Денвера — удобство при удаленной работе сразу над несколькими независимыми проектами и возможность размещения на Flash-накопителе.
Базовый пакет Денвера включает:
Инсталлятор (поддерживается также инсталляция на flash-накопитель).
Apache.
PHP5 с поддержкой MySQL.
MySQL5 с поддержкой транзакций.
Система управления виртуальными хостами, основанная на шаблонах. Чтобы создать новый хост, вам нужно лишь добавить директорию в каталог /home, править конфигурационные файлы не требуется. По умолчанию уже поддерживаются схемы именования директорий многих популярных хостеров; новые можно без труда добавить.
Система управления запуском и завершением всех компонентов Денвера.
phpMyAdmin — система управления MySQL через Web-интерфейс.
Эмулятор sendmail и SMTP-сервера (отладочная «заглушка» на localhost:25, складывающая приходящие письма в /tmp в формате .eml); поддерживается работа совместно с PHP, Perl, Parser и т.д.

phpMyAdmin

Программа phpMyAdmin является одним из самых распространенных средств для администрирования, управления и каждодневной работы с базами данных MySQL. Практически на каждом хостинге, где установлена поддержка этой СУБД, в качестве панели управления предлагают phpMyAdmin. Что же собой представляет этот инструмент?

phpMyAdmin - это набор скриптов, написанных на PHP, который предоставляет практически все необходимые функции по работе с базами данных MySQL. Все функции доступны прямо из браузера, даже перезагрузка удаленного сервера (если эта возможность разрешена учетной записью пользователя).сновная документация phpMyAdmin: возможности программы:

phpMyAdmin позволяет:
создавать и удалять базы данных
создавать, копировать, удалять, переименовывать и изменять таблицы
осуществлять сопровождение таблиц
удалять, править и добавлять поля
выполнять SQL-запросы, в том числе пакетные SQL-запросы
управлять ключами
загружать текстовые файлы в таблицы
создавать (*) и просматривать дампы таблиц
экспортировать (*) данные в форматах CSV, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, Excel и LATEX
администрирование нескольких серверов
управлять пользователями MySQL и привилегиями
проверять целостность ссылочных данных в таблицах MyISAM
использовать запрос по образцу (Query-by-example - QBE), создавать комплексные запросы, автоматически соединяясь с указанными таблицами
создавать графическую схему базы данных в формате PDF
осуществлять поиск в базе данных или в её разделах
модифицировать хранимые данные в различные форматы, использующиеся в предустановленных функциях, например, отображение BLOB-данных как изображений или как загружаемые ссылки и т.д.
поддерживает InnoDB таблицы и внешние ключи (см. FAQ 3.6)
поддерживает mysqli, улучшенное расширение MySQL (см. FAQ 1.17)

Лабораторные работа 2 Основы языка PHP. Простые и составные операторы. Комментарии, переменные, Типы данных.

Основы PHP

Для того чтобы добавить в программу HTML команды языка PHP, их необходимо заключить в специальные теги, различают четыре вида тегов, мы же будем использовать следующий вид:

```
<?
…..
?>
```

Простые и составные операторы

Конструкция echo(5+5); называется простым или однострочным оператором, каждый оператор заканчивается разделителем «точка с запятой» Простые операторы могут содержать перевод строки Например

```
<?
Echo(5+5); /* эквивалентен */
?>
<? Echo(5
+5); ?>
```

Комментарии

В данном языке программирования предоставляются несколько вариантов написания комментариев к программе:

```
<?
echo "Текст"; //это первый вид комментариев.
echo "<br>Текст на 2 строке"; # это второй тип комментариев.
echo "<br>Текст на3 строчке"; /*это третий тип комментариев */
?>
```

Комментарии действуют внутри тегов кода PHP/

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.

В PHP переменные начинаются со знака $. Имена переменных в языке PHP чувствительны к регистру поэтому $User и $user, это две различные переменные.

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

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

Название	Пример, характеристика
Integer	Этот тип предназначен для вывода целых, реальных, натуральных чисел. Диапазон [..]-3-2-1 0 1 2 3[..] $gold = "589";
Double	Это тип предназначен для вывода дробных чисел. $USD = 5.33; Внимание! В качестве отделителя целой части от дробной выступает точка (.), а не запятая (,)!
String	Этот предназначен для вывода текстовой информации. $name = "Владимир Владимирович";
Boolean	Этот тип возвращает два значение: True - если выражение действительно, и False - если выражение не действительно. $separate = TRUE;

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 - строковый, 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 именная переменных заменяются на значения переменных. Вот пример:

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

В программе можно определить тип переменной :

```
<?
$l=5;
$l1=”5”;
echo(gettype($l));
echo(<br>);
echo(gettype($l1));
```

Существует также функция settype()- которая назначает явно тип переменной:

```
<?
$a=”6”;
echo(gettype($a));
settype($a,integer);
echo(gettype($a));
?>
```

кроме функции settype(), можно прообразовывать типы переменных с помощью операторов преобразования типов:

```
<?
$var=”5”;
$var=int($var);
echo($var gettype($var));
```

?>

Лабораторные работа 3 Операторы вывода информации

Операции над числами

Выполнение операций над числами выполняется с помощью арифметических операторов:

+	Сложение
*	Умножение
-	Вычитание
/	Деление
%	Деление по модулю (Вычисляет остаток от деления Н-р 5%2=1)
++	Увеличение на единице (инкремент)
--	Уменьшение на единицу (декремент)

Различают две формы инкремента: префиксная и постфиксная Пример использования:

```
<?
// префиксная форма
$var=5;
echo(++$var); //результат будет 6
// постфиксная  форма
$var=7;
echo($var++);// результат будет 7
?>
```

Не верно писать следующие:

```
++1;
++($a+$b);
```

Но верно применять данный оператор для символьных переменных Н-Р

```
<?
$var=”aaa”;

++$var;

echo($var); // результат будет aab
?>
```

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 выполняется 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

Лабораторные работа 4 Логические операторы. Операторы сравнения

Операторы сравнения

(>)- больше, (<) меньше, (>=) больше или равно (<=) меньше или равно

(==) равно, (!=) не равно. Все эти операторы сравнивают заданные значения и возвращают TRUE если условие выполняется и FALSE если условие не выполняется

Логические операторы

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($n==10)...;	Логическое равно. Возвращает истину, если левая часть равна правой
!=	if($n!=10)...;	Логическое не равно. Возвращает 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.

Операторы.
Операторы выбора
Условный оператор выбора if
Синтаксис условного оператора :
If (условие) оператор 1 else оператор 2

В качестве аргумента условие оператор if принимает логическую переменную или выражение, возвращающие логическую переменную. Если оно истинно, то выполняется оператор 1 , в противном случае выполняется оператор 2. Допустима сокращенная форма записи условного оператора в которой отсутствуют else и оператор 2.
ПРИМЕР:

```
<?
$flag=true; // Истина
if ($flag)
{
echo “<p>Переменная  flag имеет значение true</p>”;
}
else
{
echo “<p>Переменная  flag имеет значение false</p>”;
}
```

```
?>
```

В условном операторе оператор 1 и оператор 2 в свою очередь могут быть условными, что позволяет организовывать цепочки проверок любой степени вложенности. В этих цепочках условный оператор может быть полным или сокращенным В связи с этим возникает множество ошибок не однозначного составления if и else. Синтаксис языка PHP предполагает, что при вложенных условных операторах каждое else соответствует ближайшему if.

ПРИМЕР

```
<?
$x=1;
$y=2;
if ($x==1) // первый условный оператор
   if ($y==2) echo(“x=1 и y=2”);//второй условный оператор
   else echo(“y не равен 2”);
?>
```

Условная операция

В PHP предоставляется возможность заменять блоки if….else условной операцией. В изображении условной операции присутствует два размещенных не подряд символа “ ? ” и “:” и три операнда выражения:

Выражение_1 ? выражение _2 : выражение _3

Первым вычисляется значение выражения_1 если оно истинно, то вычисляется значение выражения_2, которое и становится результатом, если значение выражения_ 1 ложно, то в качестве результата берется выражение_3.

Классическим примером условной операции является выражение:

x<0 ? –x : x;

ПРИМЕР

```
<?
$y=9;
$x=6;
echo"x=$x y=$y <br>";
$t =($x<$y) ? "x меньше y"  : "y меньше x";
echo "$t <br>"  ;
?>
```

Лабораторные работа 5 Задачи на разветвления

Переключатель SWITCH

Переключатель switch является более удобным средством для организации множественного выбора. Синтаксис переключателя:

```
Switch (expression)  // переключающие выражение
{
case value1:
      statements;//блок операторов
            break;
case value2:
      statements;//блок операторов
            break;
case value3:
            statements;//блок операторов
            break;
```

```
case value4:
      statements;//блок операторов
            break;

default:
statements;//блок операторов
  }
```

Управляющая структура switch передает управление тому оператору из помеченных операторов case, для которого значение константного выражения совпадает со значением переключающего выражения. Сначала анализируется переключающее выражение expression и осуществляется переход к той ветви программы, для которой его значение совпадает с константным выражением. Далее следует выполнение оператора или группы операторов до тех пор. Пока не встретится ключевое слово break, которым обозначается выход из переключателя. Если же значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору помеченному меткой default. В каждом переключателе может быть не более одной метки default. Ключевые слова break и default не являются обязательными и их можно опускать.

ПРИМЕР

```
<?
$x=3;
switch($x)
{
case 1:
echo"<br>Выполняется пункт  $x";
break;
case 2:
echo "<br> Выполняется пункт $x";
break;
default:
echo "<br>Ошибка в выполнении";
}
?>
```

Функции для работы со степенью

Функция sqrt- возвращает квадратный корень аргумента.

```
Float sqrt(float $arg)
<?
$num=sqrt(4)
?>
```

Функция log- возвращает натуральный логарифм аргумента

```
Float log (float $arg)
<?
$num=log(15) // результат равен 2.7080502011022
?>
```

Функция log10- возвращает десятичный логарифм аргумента

```
Float log 10(float $arg)
<?
$num=log10(15) // результат равен 1.1760912590557
?>
```

Функция pow- возведение в степень
Возвращает число $base в степени $exp
Float log (float $base? Float $exp)
$num=pow(2,8) // результат равен 256
Лабораторные работа 6 Работа с циклами

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<br>";
```

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

Самостоятельная работа студентов с преподавателем
Язык HTML

1. Формы в HTML документах.

Самостоятельная работа студентов без преподавателя
ТЕМЫ работ

1. Однострочные поля ввода

2. Поля ввода пароля
3. Независимые переключатели
4. Зависимые переключатели
5. Списки выбора. С единственным выбором.
6. Списки с множественным выбором.

Задания для выполнения самостоятельных работ

1. Лабораторная работа №1.
    a. «Используя, все теги форм разработать анкету для регистрации пользователя на почтовом сервере».
    b. «Используя, все теги форм разработать анкету для приема сотрудника на работу в учебное заведение»
    c. «Используя, все теги форм разработать анкету для проведения социологического опроса на тему «Интернет “+” и “-”»»
2. Лабораторная работа №2. «Обработка radio кнопок».
3. Лабораторная работа №3. «Работа с CHECKBOX». «Пользователю предлагается выбрать из набора рисунков (12 шт.) все картинки, принадлежащие к одной тематической группе».
    a. Фрукты, овощи, ягоды
    b. Времена года
4. Лабораторная работа №4. «Работа со списком». «Создать список, длиною 20 строк, состоящий из чисел выбранных случайным образом из диапазона от 0 до 50. Пользователь выбирает любое n-количество чисел. Произвести расчет среднего арифметического выбранных чисел».

Вопросы к модулю 1.

1. Однострочные поля ввода
2. Поля ввода пароля
3. Независимые переключатели
4. Зависимые переключатели
5. Списки выбора. С единственным выбором.
6. Списки с множественным выбором.
7. Формы в HTML документах.
8. Математические функции языка PHP
9. Операторы циклов
10. Операторы разветвления
11. Ввод многострочного текста

МОДУЛЬ II Функции для работы с массивами

КОЛИЧЕСТВО НЕДЕЛЬ 9

Модуль 2. Включает в себя:

-Лекционные занятия в объеме 3 часов.

- Лабораторные занятия в объеме 6 часов
- Самостоятельная работа студентов с преподавателем в объеме 3 часов
- Самостоятельная работа студентов без преподавателя в объеме 9 часов
- Сдача модуля организуется виде контрольной работы.

Лекция 7,8,9 Функции пользователя. Создание и использование собственных функций

Функции

Функция представляет собой самостоятельный фрагмент программы, предназначенный для реализации определенных действий. Выполнение этих действий достигается вызовом функции в нужном месте программы. Обычно функция принимает несколько аргументов,

используя эти аргументы функция выполняет некоторые операции и при необходимости выводит полученной значение.

Определения и вызов функции.

Функция определяется при помощи ключевого слова function, после которого в фигурных скобках записываются различные операторы, составляющие тело функции:

```
Function Myfunction()
{
// операторы
}
```

Если функция принимает аргументы, то они записываются как переменные в объявлении функции. Аргумент функции представляет собой переменную, передаваемую в тело функции для дальнейшего использования в операциях. В случае, использования больше одного аргумента, эти переменные разделяются запятыми:

```
Function Myfunction()
{
return $ret; // возвращается значение переменной $ret
}
```

Пример

```
<?
function get_sum()
{
$var=5;
$var1=10;
$sum=$var+$var1;
return $sum;
}
echo(get_sum()); // результат 15
?>
```

Модифицируем функцию, чтобы она не возвращала полученный результат, а выводила его в браузер.

```
function get_sum1()
{
$var=5;
$var1=10;
$sum=$var+$var1;
echo($sum);
}
get_sum1(); // результат 15
?>
```

Переменные $var и $var1 мы можем объявить как аргументы- в этом случае в теле функции их объявлять не надо.

```
function get_sum2($var1,$var)
{
$sum=$var+$var1;
echo($sum);
}
get_sum2(5,11); // результат 16
```

?> Переменная содержащая значение значение, переданное через аргумент, называется параметром функции. Передача аргумента функции по значению:

```
function get_sum3($var)
{
```

```
$var=$var+5;
return $var;
}
$per=3;
echo(get_sum3($per)); // результат 8
```

?> Для того чтобы переменные, переданные функции, сохраняли свое значение при выходи из нее, применяется передача параметров по ссылке, при любом изменении значения параметра происходит изменение переменной аргумента.

```
function get_sum4(&$var)
{
$var=$var+5;
return $var;
}
$per=3;
echo(get_sum4($per)); // результат 8
echo("<br> $per"); // результат 8
```

Лабораторные работа 7 Синтаксис заданий функций 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. Сайчас 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 рекурентные 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.

Лабораторные работа 8 Передача параметров функций (передача значений)
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.

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){
  ...
}
...
```

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.

Лабораторные работа 9 Локальные и глобальные переменные

Переменные в функциях имеют локальную область видимости. Это означает, что если локальная и внешняя переменная имеют одинаковые имена то изменение локальной переменной ни как не отразится на внешней переменной.

```
?
 function get_sum()
 {
$var=5;// локальная переменная
echo($var);
        }
$var=10;// внешняя переменная
get_sum();
echo("<br> $var"); // Выводит 10 внешняя перемення
```

Локальную переменную можно сделать внешней , если перед ее именем указать слово global. Если внешняя переменная объявлена как global, то к ней возможен доступ из любой функции.

```
 echo("<hr> <br>");
 function get_sum1()
 {
 global $var;
$var=5;// локальная переменная
echo("<br> $var");
        }
$var=10;// внешняя переменная
echo($var);
get_sum1();
```

Время жизни переменной называется интервал выполнения программы, в течение которого она существует Поскольку локальные переменные имеют своей областью видимости функцию, то время жизни локальной переменной определяется временем выполнения функции, в которой она объявлена. Это означает, что в различных функциях могут использоваться переменные с одинаковыми именами. Локальная переменная при каждом вызове функции инициализируется заново. Для того чтобы локальная переменная сохраняла свое предыдущие значение при новых вызовах функции , ее можно объявить статистической при помощи ключевого слова ststic

```
Function shet()
{
static $s=0;
return ++$s;
```

} При первом обращении $s=0 при повторном обращении к функции переменная помнит, чему равно ее предыдущее значение. Временем жизни статистических переменных является время работы сценария, если пользователь перегружает страницу, то это приводит к новому выполнению скрипта и $s=0.

Самостоятельная работа студентов с преподавателем

Понятие массива и списка.

Ассоциативные массивы

Самостоятельная работа студентов без преподавателя

ТЕМЫ РАБОТ

1. Операции над массивами. Удаление массивов. Слияние массивов. Перебор массивов. Сортировка массивов.
2. Получение части массива.
3. Добавление и удаление элементов массива
4. Работа с многомерным массивом

Задания для выполнения самостоятельных работ

1. Лабораторная работа №5. «Создать тест, используя, тег-список» Пример «Исключи лишнее слово»
2. Лабораторная работа №6. «Анаграммы»

Вопросы к модулю II

Понятие массива и списка.

1. Ассоциативные массивы
2. Операции над массивами. Удаление массивов. Слияние массивов. Перебор массивов. Сортировка массивов.
3. Получение части массива.
4. Добавление и удаление элементов массива

Модуль принимается в виде контрольной работы.

МОДУЛЬ III Работа с файлами.

КОЛИЧЕСТВО НЕДЕЛЬ 13

Модуль 3. Включает в себя:

-Лекционные занятия в объеме 5 часов.

- Лабораторные занятия в объеме 12 часов
- Самостоятельная работа студентов с преподавателем в объеме 7 часов
- Самостоятельная работа студентов без преподавателя в объеме 15 часов

Сдача модуля организуется виде контрольной работы.

Лекция 10,11,12,13,14. Основные функции для работы переменными

Массивы

Массив представляет собой индексированную совокупность переменных одного типа. Каждая переменная или элемент массива имеет свой индекс, т.е. все элементы массива последовательно проименованы от 1 до n, где N длина массива. Часто длину массива n называют размерностью массива.

Стеки представляют собой объект для хранения данных, работа с которыми осуществляется по принципу LIFO(last-in first-out- “последним зашел первым ушел”). Добавление объектов в стек может происходить в любой момент, но удаляется только тот объект, который добавлен последним. Наглядно стек можно представить как стопу тарелок. Когда добавляется очередная тарелка, она становиться самой верхней в стопке, и при надобности она будет взята первой.

Более близким к информационным технологиям примером стека является механизм отмены действий в текстовых редакторах, с помощью которого можно отменить последние операции редактирования. В Интернет-браузерах адреса посещаемых пользователем WEB-сайтов также хранятся в виде стеков. При посещении нового сайта, его адрес добавляется в стек адресов. Нажав кнопку НАЗАД пользователь может вернутся в открытые ранее сайтам, при этом откроется сайт, который был посещен последним.

Очереди. Эта структура аналогична стеку, с той разницей, что те элементы которые были добавлены в очередь первыми, первыми из нее и извлекаются. Очередь действует по принципу FIFO(first-in first-out- “первым пришел, первым ушел”). Добавление элементов в очередь происходить в любой момент, удаляется же только тот элемент который был добавлен в очередь первым. При этом говорят, что элементы добавляются в очередь с конца, а удаляются с начала. В быту Вы можете видеть очередь например в тех же магазинах. Люди встают в очередь с конца, а покупают товар те кто находятся в начале очереди. По принципу очереди осуществляются практически все операции в

приложениях, связанных с автоматической обработкой клиентских запросов, к примеру, в приложениях для автоматического бронирования номеров в гостиницах через Интернет

Инициализация массивов

Инициализация одномерных массивов.

В PHP существует 2 метода инициализации массивов. Первый из них состоит в простом присвоении значений элементам массива.

```
<?
$car[]=”автобус”;
$car[]=”грузовик”;
Echo ($car[1]);
?>
```

Результатом выполнения будет вывод –грузовик, потому, что инициализация массива в PHP начинается с 0. Индекс массива можно указывать и явно.

```
<?
$car[0]=”автобус”;
$car[1]=”грузовик”;
Echo ($car[1]);
?>
```

Если при объявлении элементов массива смешиваются переменные с явной индексацией и без индексации, то тому элементу, индеек которого не задан, PHP присвоит первый доступный индекс после наибольшего использованного до сих пор индекса. Например, если мы создадим массив с элементами, индексы которого будут равны 10,20,30, а потом создадим элемент, индеек которого явно не укажем, то ему автоматически будет присвоен индекс 31. Пример:

```
<?
$car[10]=”автобус”;
$car[20]=”грузовик”;
$car[30]=”трактор”;
$car[]=”легковой автомобиль ”;
Echo ($car[31]);
?>
```

Результат выполнения программы «легковой автомобиль»

Второй способ определения массивов состоит в использовании конструкции array

```
<?
$car=array(“трактор”,”грузовик”);
Echo($car[1]); // выводит грузовик
?>
```

Для явного указания индексов можно применить оператор =>

```
<?
$car=array(“трактор”,5=>”грузовик”,”мерседес”,”жигули”);
Echo("$car[0] <br>"); // выводит трактор
Echo("$car[5] <br>"); // выводит грузовик
Echo("$car[6] <br>"); // выводит Мерседес
Echo("$car[7] <br>"); // выводит жигули
?>
```

Индексами массива могут быть не только цифры, но и строки:

```
<?
$car=array(“ps”=>“трактор”,”lr”=>”грузовик”);
Echo("$car[ps] <br>"); // выводит трактор
Echo("$car[lr] <br>"); // выводит грузовик
```

```
?>
```

На лекции о Циклах нами было рассмотрено 3 из 4 возможных в PHP.

ЦИКЛ foreach для обхода массивов

Обход массива в цикле можно организовать при помощи цикла Foreach который имеет следующий синтаксис:

```
Foreach (array as [$key=>] $value)
{
        Операторы;
}
```

Принцип работы цикла прост: при проходе каждого элемента массива в переменную $key помещается индекс этого массива, а в переменную $value – его значение. Имена этих переменных могут быть любыми.

ПРИМЕР

```
<?
$car=array(“автобус”,”грузовик”,”трактор”,”легковой автомобиль”,”ока”);
Foreach($car as $index=>$val)
{
Echo”$index->$val <br>”;
}
?>
```

Переменная $key необязательна и может быть пропущена

```
$car=array(“автобус”,”грузовик”,”трактор”,”легковой автомобиль”,”ока”);
Echo “<ul>”;
Foreach($car as $index=>$val)
{
Echo”<li> $val </li> <br>”;
}
Echo “</ul>”;
?>
```

Многомерные массивы

Для инициализации многомерных массивов используются вложенные конструкции array/

Обход многомерных массивов используются вложенных циклов.

Пример

```
<?
$ship=array(
                “Пассажирские корабли ” => array(“Лайнер”, “Яхта”,”Паром”),
                “Военные корабли”      => Array(“Линкор”,”Крейсер”,”Эсминец”),
                “Грузовые корабли” => array(“Танкер”,”сухогруз”,”контейнеровоз”);
Foreach($ship as $key=>$type)
{
      echo("$key <br>");
      echo("<ul>");
      foreach($type as $ship)
      {
       echo("<li>$ship");
       }
        echo("</ul>");
}
?>
```

Основные функции для работы с массивами

Название функции	**Синтаксис**	**Назначение функции**
COUNT	int count(mixed var)	Эта функция принимает в качестве

		аргумента массив и возвращает количество элементов в нем

ПРИМЕР
<?
$car=array(“автобус”,”грузовик”,”трактор”,”легковой автомобиль”,”ока”);
Echo(count($car)); // выводит 5
?>		

IN_ARRAY	Boolean in_array (mixed needle, array haystack)	Функция поиска элемента в массиве она ищет в массиве haystack значение needle и возвращает true, если оно истинно и false в противном случае.

ПРИМЕР
<?
$car=array(“автобус”,”грузовик”,”трактор”,”легковой автомобиль”,”ока”);
Echo(in_array(“автобус”,$car)); // выводит true или 1
?>		

RESET	mixed reset(array array)	Функция reset устанавливает указатель массива на первый элемент и возвращает значение первого элемента массива

Сортировка массивов

Сортировка массивов является одной из наиболее часто встречающихся задач. Без нее не обходится практически ни одно приложение, в котором выполняется работа с массивами.

SORT	Void sort(array array [, int sort_flags])	Эта функция сортируется массив array по возрастанию. Необязательный аргумент sort_flags указывает, как именно должны сортироваться элементы (задает флаги сортировки). Допустимы следующие значения этого аргумента: **sort_regular**(задает нормальное сравнение элементов, т.е. сравнивает элементы “как есть”) **sort_numeric**(сравнивает элементы как числа) **sort_string**(сравнивает элементы как строки)

ПРИМЕР
<?
$arr=array(“2”,”1”,”3”,”5”,”4”);// исходный массив
echo (“До сортировки массива:
”);
for($i=0; $i<count($arr);$i++)
{
Echo(“$i:$arr[$i] ”);
}
sort($arr); // сортируем массив
echo (“
 после сортировки массива:
”);
for($i=0; $i<count($arr);$i++)
{
Echo(“$i:$arr[$i] ”);
}		

```
?>
РЕЗУЛЬТАТ
До сортировки массива:
0:2 1:1 2:3 3:5 4:4
после сортировки массива:
0:1 1:2 2:3 3:4 4:5
Сортировка строк производится в алфавитном порядке по старшинству первой буквы. Такую сортировку часто называют альфа-бета порядок
<?
$arr=array(“Лена”,”Катя”,”Маша”,”Даша”,”Оля”);// исходный массив
echo (“До сортировки массива: <br>”);
for($i=0; $i<count($arr);$i++)
{
 Echo(“$i:$arr[$i]    ”);
}
sort($arr); // сортируем массив
echo (“<br> после сортировки массива: <br>”);
for($i=0; $i<count($arr);$i++)
{
 Echo(“$i:$arr[$i]    ”);
}
?>
РЕЗУЛЬТАТ
До сортировки массива:
0:Лена 1:Катя 2:Маша 3:Даша 4:Оля
после сортировки массива:
0:Даша 1:Катя 2:Лена 3:Маша 4:Оля
```

Функции для работы с массивами

Название функции	**Синтаксис**	**Назначение функции**
RSORT	Void rsort(array array [, int sort_flags])	Функция сортировки массива по убыванию. Во всем остальном ведет себя аналогично функции sort

<?
$arr=array(“2”,”1”,”3”,”5”,”4”);// исходный массив
echo (“До сортировки массива:
”);
for($i=0; $i<count($arr);$i++)
{
Echo(“$i:$arr[$i] ”);
}
rsort($arr); // сортируем массив
echo (“
 после сортировки массива:
”);
for($i=0; $i<count($arr);$i++)
{
Echo(“$i:$arr[$i] ”);
}
?>		

ASORT	Void assort(array array[i, int sort_flags])	Функция сортировки ассоциативного массива по возрастанию. Функция assort сортирует массив array так чтобы его значения шли в алфавитном порядке(если это строки) и по возрастанию (если это числа). Значения не обязательного параметра sort_flags такие же как для функции sort. При сортировки функцией asort связи между ключами и соответствующими им значениями сохраняются

Лекция 15 Работа с датой 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.

Лабораторные работы 10 Строковые переменные. Функции поиска в тексте

Строковые функции

PHP предоставляется большой выбор функций для работы со строками. Все их можно подразделить на четыре основные группы:

Функции преобразования символов
Функции поиска в тексте
Функции удаления пробельных символов
Функции форматного ввода

Функции поиска в тексте

SUBSTR

String substr(sring string, int start[, int length])

Эта функция возвращает часть строки. Первый аргумент функции – исходная строка, из которой вырезается текст, второй – положение первого символа (отсчет начинается с нуля) в строке, которую нужно вернуть, третий – длина возвращаемой строки в символах. Если третий аргумент не указан, то возвращается вся оставшаяся часть строки.

```
<?
$msg=substr("Вот и наступила осень",0,1);
echo $msg;
?>
```

STRPOS

String strpos(sring haystack, string needle[, int offset])

Эта функция возвращает позицию в строке haystack, с которой начинается переданная ей подстрока needle

```
<?
$var=strpos("123456789","1"); // начиная с 0
echo "$var";
```

```
echo"<br>";
```

?>, необязательный аргумент offset позволяет указать в сроке позицию с которой нужно начать поиск:

STRRPOS

String strpos(sring haystack, string needle)

Эта функция ищет в строке haystack последнюю строку, в которй встречается символ needle

```
$var=strrpos("12345678945","45");// вывод 9
echo "$var";
echo"<br>";
```

STRSTR

String strstr(sring haystack, string needle)

Данная функция возвращает участок строки, заданной в параметре haystack, начиная с первого фрагмента, указанного в параметре needle, и до конца строки.

В случае неудачи возвращается false.Функция чувствительна к регистру

```
$url=("http://www.rambler.ru");
$www=strstr($url,"w");
echo "$www"; // результат www.rambler.ru
echo"<br>";
```

STRISTR

String stristr(sring haystack, string needle)

Данная функция возвращает участок строки, заданной в параметре haystack, начиная с первого фрагмента, указанного в параметре needle, и до конца строки.

В случае неудачи возвращается false.Функция не чувствительна к регистру чувствительна к регистру

```
$url=("http://www.rambler.ru");
$www=strstr($url,"w");
echo "$www"; // результат www.rambler.ru
echo"<br>";
```

STRRCHR

String strrchr(sring haystack, string needle)

Функция strrchr отличается от функции strstr тем, что осуществляется поиск последнего вхождения подстроки. Т.е. эта функция возвращает участок строки, заданной в параметре haystack, начиная с последнего фрагмента, указанного в параметре needle, и до конца строки. В случае неудачи возвращается false. Функция strrchr чувствительна к регистру. В следующем примере рассмотрим скрипт, определяющий каталог, который прописан последним в переменной окружения $PATH.

```
echo($PATH);
echo("<br>");
$dir=substr(strrchr($PATH,";"),1);
echo $dir;
/*РЕЗУЛЬТАТ
C:\\WINDOWS;C:\\WINDOWS\\COMMAND
C:\\WINDOWS\\COMMAND
*/
```

Функция htmlspecialchars- Функции преобразования символов и производит преобразование специальных символов в их HTML-эквиваленты. Функция **htmlspecialchars** выделена в отдельный раздел в силу ее исключительной важности, так она позволяет надежно защитить ваши приложения. Применение этой функции

гарантирует, что любой введенный пользователем код (PHP, JavaScript) будет отображен, но выполнятся не будет. Если же не обрабатывать введенный текст функцией **htmlspecialchars,** то код**,** который может содержаться в этом тексте, выполнится. Таким образов, эту функцию следует применять, если нужно вывести в браузере какой-то код

Всегда применяйте эту функцию при обработке текстовых сообщений из формы, так как в этом случае вы будет надежно защищены от всяких нехороших людей, которым нечем заняться, кроме как писать вредоносные скрипты в ваших гостевых книгах, форумах.

Лабораторные работы 11 Функции форматного вывода

Функции форматного вывода

Функции printf и sprintf

int printf(stringe format [,mixed args])- выводит в браузер

string sprintf(stringe format [,mixed args])- выводит в форматированном виде в строку

Использование спецификаторов вывода

```
*/
$day=2;
$month=3;
$year=2003;
printf("%2d/%02d/%04d",$day, $month,$year);
// результат 02/03/2003

/*
```

Лабораторные работы 12 Функции замены в строке

Сравнение строк

функция strcmp

int strcmp(stringe str1, stringe str2)- функция strcmp сравнивает две строки и возвращает:

0- если сторки полностью совпадают

1- если, сторка str1 лексиграфически больше сторки str2

-1- если, сторка str1 лексиграфически меньше сторки str2

Поскольку сравнение происходит побайтово, регистр символов влияет на сравнение

```
*/
echo("<hr>");
$str1="tttt";
$str2="ttttt";
echo ("<br>результат сравнения строк     ".$str1."      и          ".$str2);
echo(strcmp($str1,$str2)) ;
echo ("<br >результат сравнения строк      ".$str2."      и          ".$str1);
echo(strcmp($str2,$str1)) ;
echo ("<br>результат сравнения строк     ".$str2."      и          ".$str2);
echo(strcmp($str2,$str2)) ;
/*
```

функция similar_text

int similar_text(stringe str_first, stringe str_secon[,double percent)- функция similar_text определяет схожесть двух строк по алгоритму Оливера. Функция возвращает число символов совпавших в строках str_first и b str_second. Третий не обяхательный параметр передается по ссылке и в нем хранятся процент совпадения строк

Замечание: В место стека в псевдо коде Оливера , эта функция использует рекурсивные вызовы, что делает ее достаточно медленной. Скорость вполнения этой функции равна n в третьей степени, где n - длина строки

```
*/
echo("<br>");
```

```
$str1="привет всем";
$str2="привет";
$var=similar_text($str1,$str2,&$proz);
echo("$var");
echo ("<br>");
echo("$proz");

?>
```

Лабораторные работы 13 Изменение регистров

Работа с блоками текста

Функция wordwrap

синтаксис данной функции

string wordwrap(string str[, int width[,stringe break])

Функция wordwrap разбивает исходный текст на строки с определенными завершающими символами. Согласно синтаксису, эта функция разбивает блок текста str на несколько строк, которые завершаются символами break. В каждой из строк, на которые разбивается блок текста, количество букв не превышает значение, указанное в аргументе width. Поскольку разбиение слов происходит по границам слов, текст остается вполне читаемым

```
*/
$str="Здесь были студенты ВШ НИТ";
$w=wordwrap($str,10,"<br>");
echo ("$w <br>");
echo("<hr>");
/*
```

ФУНКЦИЯ str_replace

Синтаксис функции

string str_replace(string trom, string to, string str);

Функция заменяет в исходной строке str одни подстроки на другие. То есть эта функция заменяет в строке str все вхождения подстроки from на to и возвращает результат.

```
*/
$str="Здесь были студенты ВШ НИТ";
$s="ВШ НИТ";
$s1="ВШ ЭиБ";
$w=str_replace($s,$s1,$str);
echo ("$w <br>");
echo("<hr>");
/*
```

ФУНКЦИЯ substr_replace

Синтаксис функции

string substr_replace(string str, string replacement, int start [, int length]);

Функция заменяет в исходной строке str одни подстроки на другие. Она заменяет строку str, в которой часть от символа с позицией start и длиной length заменяется строкой replacement. Если аргумент длины Length не задан, замена производится до конца строки. Если значение аргумента start положительно отсчет производится от начала строки, если отрицательно - от конца строки.

```
*/
$str="Здесь были студенты ВШ НИТ";
$s1="был я ";
$w=substr_replace($str,$s1,6);
echo ("$w <br>");
echo("<hr>");
/*
```

```
Функция удаления обратных слешей  stripslashes
stringe stripslashes (string str)
Заменяет обратные слеши на их кодовые эквиваленты
*/
$str="Здесь \были \студенты \ВШ НИТ";
$w=stripslashes($str);
echo ("$w <br>");
echo("<hr>");
/*
Функция   strrev
strrev (string str)
Функция производит реверс строки
*/
$str="Здесь были студенты ВШ НИТ";
$w=strrev($str);
echo ("$w <br>");
echo("<hr>");
?>
<?
/*
Функции преобразования регистра
Функция strtolower
Синтаксис функции sring  strtolower(string str)
Производит преобразование символов строки str в нижний регистр
*/
$m="QWERTYUI";
$a=strtolower($m);
echo("$a <br>");
echo("<hr>");
/*
Функции преобразования регистра
Функция strtoupper
Синтаксис функции sring  strtoupper(string str)
Производит преобразование символов строки str в  верхний регистр
*/
$m="asdfgh";
$a=strtoupper($m);
echo("$a <br>");
echo("<hr>");
/*
Функция ucfirst
Синтаксис функции sring  ucfirst (string str)
Производит преобразование первого символа строки str в верхний регистр
*/
$m="asdfgh";
$a=ucfirst($m);
echo("$a <br>");
echo("<hr>");
/*
Функция ucwords
Синтаксис функции sring  ucwords (string str)
```

```
Производит преобразование  первого символа каждого слова в строке str в верхний регистр. Под словом понимается участок строки, которому предшествует любой пробельный символ.
*/
$m="международный университет кыргызстана";
$a=ucwords($m);
echo("$a <br>");
echo("<hr>");
?>
```

Лабораторные работы 14 Преобразование символов
Строковые функции

```
<?
/*функция substr_count
int substr_count(stringe hatstack, stringe needle)
находит количество вхождений фрагмента в строку и возвращает число фрагментов
*/
$str="Мы рады Вам!";
$str_count=substr_count($str,"Мы");
echo $str_count; //Результат 1
echo("<br> <hr>");
/*функция strspn
int strspn(stringe str1, stringe str2)
определяет присутствие символа в строке.
*/
$str="as2as1as1";
$rez=strspn($str,"as1");
echo($rez);
echo("<br> <hr>"); // результат 3

/*функция strspn
int strcspn(stringe str1, stringe str2)
определяет отсутствие символа в строке.
*/
$str="as12as1";
$rez=strcspn($str,"2");
echo($rez);
echo("<br> <hr>"); // результат 3
/*функция strlen
возвращает длину строки, которую принимает в качестве аргумента
*/
$str="jhksjfhd dsfhkjsfh sdfhksjfh";
$dlin=strlen($str);
echo($dlin);
echo("<br> <hr>"); // результат 28
/*функция chr
функция принимает в качестве аргумента ASCII-код символа и возвращает соответствующий этому коду фактический символ.
*/
$str=chr(36);
echo($str); // результат $
echo("<br> <hr>");
```

```
/*функция ord
функция выполняет обратное действие chr функции возвращает  ASCII-код символа
*/
$str=ord("$");
echo($str); // результат 36
echo("<br> <hr>");
/*
Функции форматного вывода
Функции printf и sprintf
int printf(stringe format [,mixed args])- выводит в браузер
string sprintf(stringe format [,mixed args])- выводит в форматированном виде в строку
Использование спецификаторов вывода
*/
$day=2;
$month=3;
$year=2003;
printf("%2d/%02d/%04d",$day, $month,$year);
// результат 02/03/2003
```

Лабораторные работы 15 Функции удаления пробелов

```
/* TRIM функции функции удаления пробельных символов
trim- принимает в качестве аргумента с троку и удаляет из нее пробельные символы
*/
$str="  ВШНИТМУК";
echo("Длигна исходной строки   ".strlen($str));
$udalim=trim($str);
echo (" <br> Длина полученной строки    ". strlen($udalim));
echo("<br> <hr>");
/*
Функция ltrim-  удаление начальных пробелов, rtrim, chop- удаление конечных пробелов
*/

?>
?
```

Самостоятельная работа студентов с преподавателем

Работа с файлами. Открытие файлов.
Запись в файл.
Чтение файлов

Самостоятельная работа студентов без преподавателя

ТЕМЫ РАБОТ

Функции для работы с файлами.
Временные файлы и их использование
Функции для работы с именами файлов.
Функции определения типа и параметров файла
Чтение файлов в строку
Чтение CSV-файлов
Задания для выполнения самостоятельных работ
Лабораторная работа №7. «Транслейт текста»
Лабораторная работа №8. «Создать тест, используя, любые теги форм ». Пример «Логика или интуиция».

Лабораторная работа №9. «Работа с рисунками и генератором случайных чисел» (Примеры сложения и вычитания двух чисел)
Лабораторная работа №10. «Функции для работы с файлами»
Лабораторная работа №11. «Система регистрации и авторизации пользователя для получения доступа к выполненным лабораторным работам»

Вопросы к модулю III

Работа с файлами. Открытие файлов.
Запись в файл.
Чтение файлов
Функции для работы с файлами.
Временные файлы и их использование
Функции для работы с именами файлов.
Функции определения типа и параметров файла
Чтение файлов в строку
Чтение CSV-файлов
Модуль принимается в виде контрольной работы.

Раздел 5. Методические указания для лабораторных занятий:
ПРАКТИЧЕСКИЕ ЗАДАНИЯ
Формы в HTML документах
Некоторые WWW browser позволяют пользователю, заполнив специальную форму, возвращающую полученное значение, выполнять некоторые действия на вашем WWW - сервере. Когда форма интерпретируется WEB - браузером, создается специальные экранные элементы GUI, такие, как поля ввода, checkboxes, radiobuttons, выпадающие меню, скроллируемые списки, кнопки и т.д. Когда пользователь заполняет форму и нажимает кнопку "Подтверждение" (SUBMIT - специальный тип кнопки, который задается при описании документа), информация, введенная пользователем в форму, посылается HTTP-серверу для обработки и передаче другим программам, работающим под сервером, в соответствии с CGI (Common Gateway Interface) интерфейсом.
Когда вы описываете форму, каждый элемент ввода данных имеет тэг <INPUT>. Когда пользователь помещает данные в элемент формы, информация размещается в разделе VALUE данного элемента.

Синтаксис
Все формы начинаются тэгом <FORM> и завершаются тэгом </FORM>
<FORM METHOD="get|post" ACTION="URL"> Элементы _формы_и_другие _элементы _HTML </FORM>
METHOD
Метод посылки сообщения с данными из формы. В зависимости от используемого метода вы можете посылать результаты ввода данных в форму двумя путями:
GET: Информация из формы добавляется в конец URL, который был указан в описании заголовка формы. Ваша CGI-программа (CGI-скрипт) получает данные из формы в виде параметра переменной среды QUERY_STRING. Использование метода GET не рекомендуется.
POST: Данный метод передает всю информацию о форме немедленно после обращения к указанному URL. Ваша CGI-программа получает данные из формы в стандартный поток ввода. Сервер не будет пересылать вам сообщение об окончании пересылки данных в стандартный поток ввода; вместо этого используется переменная среды CONTENT_LENGTH для определения, какое количество данных вам необходимо считать из стандартного потока ввода. Данный метод рекомендуется к использованию.
ACTION
ACTION описывает URL, который будет вызываться для обработки формы. Данный URL почти всегда указывает на CGI-программу, обрабатывающую данную форму.

ТЭГИ ФОРМЫ

TEXTAREA
Тэг <TEXTAREA> используется для того, чтобы позволить пользователю вводить более одной строки информации (свободный текст). Вот пример использования тэга <TEXTAREA>:
<TEXTAREA NAME="address" ROWS=10 COLS=50>Москва, Дмитровкое шоссе,д.9Б, офис 448 </TEXTAREA>
Атрибуты, используемые внутри тэга <TEXTAREA> описывают внешний вид и имя вводимого значения. Тэг </TEXTAREA> необходим даже тогда, когда поле ввода изначально пустое. Описание атрибутов:
NAME - имя поля ввода
ROWS - высота поля ввода в символах

COLS - ширина поля ввода в символах
Если вы хотите, чтобы в поле ввода по умолчанию выдавался какой-либо текст, то необходимо вставить его внутри тэгов <TEXTAREA> и </TEXTAREA>.

INPUT
Тэг <INPUT> используется для ввода одной строки текста или одного слова. Атрибуты тэга:
CHECKED - означает, что CHECKBOX или RADIOBUTTON будет выбран.
MAXLENGTH - определяет количество символов, которое пользователи могут ввести в поле ввода. При превышении количества допустимых символов броузер реагирует на попытку ввода нового символа звуковым сигналом и не дает его ввести. Не путать с атрибутом SIZE. Если MAXLENGTH больше чем SIZE, то в поле осуществляется скроллинг. По умолчанию значение MAXLENGTH равно бесконечности.
NAME - имя поля ввода. Данное имя используется как уникальный идентификатор поля, по которому, впоследствии, вы сможете получить данные, помещенные пользователем в это поле.
SIZE - определяет визуальный размер поля ввода на экране в символах.
SRC - URL,. указывающий на картинку (используется совместно с атрибутом IMAGE).
VALUE - присваивает полю значение по умолчанию или значение, которое будет выбрано при использовании типа RADIO (для типа RADIO данный атрибут обязателен)
TYPE - определяет тип поля ввода. По умолчанию это простое поле ввода для одной строки текста. Остальные типы должны быть явно,их полный список приведен ниже:

CHECKBOX
Используется для простых логических (BOOLEAN) значений. Значение, ассоциированное с именем данного поля, которое будет передаваться в вызываемую CGI-программу, может принимать значение ON или OFF.

HIDDEN
Поля данного типа не отображаются браузером и не дают пользователю изменять присвоенные данному полю по умолчанию значение. Это поле используется для передачи в CGI-программу статической информации, как то ID пользователя, пароля или другой информации.

IMAGE
Данный тип поля ввода позволяет вам связывать графический рисунок с именем поля. При нажатии мышью на какую-либо часть рисунка будет немедленно вызвана ассоциированная форме CGI-программа. Значения, присвоенные переменной NAME будут выглядеть так - создается две новых переменных: первая имеет имя, обозначенное в поле NAME с добавлением .x в конце имени. В эту переменную будет помещена X-координата точки в пикселях (считая началом координат левый верхний угол рисунка), на которую указывал курсор мыши в момент нажатия, а переменная с именем, содержащимся в NAME и добавленным .y, будет содержать Y-координату. Все значения атрибута VALUE игнорируются. Само описание картинки осуществляется через атрибут SRC и по синтаксису совпадает с тэгом <IMG>.

PASSWORD
То же самое, что и атрибут TEXT, но вводимое пользователем значение не отображается браузером на экране.

RADIO

Данный атрибут позволяет вводить одно значение из нескольких альтернатив. Для создания набора альтернатив вам необходимо создать несколько полей ввода с атрибутом TYPE="RADIO" с разными значениями атрибута VALUE, но с одинаковыми значениями атрибута NAME. В CGI-программу будет передано значение типа NAME=VALUE, причем VALUE примет значение атрибута VALUE того поля ввода, которое в данный момент будет выбрано (будет активным). При выборе одного из полей ввода типа RADIO все остальные поля данного типа с тем же именем (атрибут NAME) автоматически станут невыбранными на экране.

RESET
Данный тип обозначает кнопку, при нажатии которой все поля формы примут значения, описанные для них по умолчанию.

SUBMIT
Данный тип обозначает кнопку, при нажатии которой будет вызвана CGI-программа (или URL), описанная в заголовке формы. Атрибут VALUE может содержать строку, которая будет высвечена на кнопке.

TEXT
Данный тип поля ввода описывает однострочное поле ввода. Используйте атрибуты MAXLENGTH и SIZE для определения максимальной длинны вводимого значения в символах и размера отображаемого поля ввода на экране (по умолчанию принимается 20 символов).

МЕНЮ ВЫБОРА В ФОРМАХ
Под меню выбора в формах понимают такой элемент интерфейса, как LISTBOX.
Существует три типа тэгов меню выбора для форм:
Select - пользователь выбирает одно значение из фиксированного списка значений, представленных тэгами OPTION. Данн вид представляется как выпадающий LISTBOX.
Select single - то же самое, что и Select, но на экране пользователь видит одновременно три элемента выбора. Если их больше, то предоставляется автоматический вертикальный скроллинг.
Select multiple - позволяет выбрать несколько элементов из LISTBOX.

SELECT
Тэг SELECT позволяет пользователю выбрать значение из фиксированного списка значений. Обычно это представлено выпадающим меню.
Тэг SELECT имеет один или более параметр пежду стартовым тэгом <SELECT> и завершающим </SELECT>.По умолчанию, первый элемент отображается в строке выбора.
Вот пример тэга <SELECT>:

```
<FORM>
<SELECT NAME=group>
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
</SELECT>
</FORM>
```

SELECT SINGLE
Тэг SELECT SINGLE - это то же самое, что и Select, но на экране пользователь видит одновременно несколько элементов выбора (три по умолчанию). Если их больше, то

предоставляется автоматический вертикальный скроллинг. Количество одновременно отображаемых элементов определяется атрибутом SIZE. Пример:

```
<FORM>
<SELECT SINGLE NAME=group SIZE=4>
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
<OPTIONS> Pentium PRO
</SELECT>
</FORM>
```

SELECT MULTIPLE

Тэг SELECT MULTIPLE похож на тэг SELECT SINGLE, но пользователь может одновременно выбрать более чем один элемент списка. Атрибут SIZE определяет количество одновременно видимых на экране элементов, атрибут MULTIPLE - максимальное количество одновременно выбранных элементов. Пример:

```
<FORM>
<SELECT SINGLE NAME=group SIZE=4 MULTIPLE=2>
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
<OPTIONS> Pentium PRO
</SELECT>
</FORM>
```

Если выбрано одновременно несколько значений, то серверу передаютс соответствующее выбранному количество параметров NAME=VALUE с одинаковыми значениями NAME, но разными VALUE.

ОТПРАВЛЕНИЕ ФАЙЛОВ ПРИ ПОМОЩИ ФОРМ

Формы можно использовать для отправки не только небольших информационных сообщений ввода параметров, а также и для отправки файлов.

Внимание! Поскольку данная возможность требует поддержки получения файлов WEB - сервером, то, соответственно, необходимо, чтобы сервер поддерживал получение файлов!

Например:

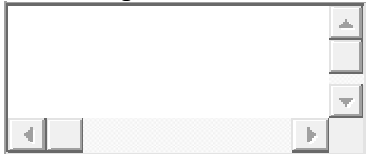
```
<FORM ENCTYPE="multipart/form-data" ACTION="url" METHOD=POST>
Отправить данный файл: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Отправить файл">
</FORM>
<FORM action="http://pandemonium.cs.nstu.ru/~gun/docs/sites.htm" encType="multipart/form-data" method="post">
Отправить данный файл: <INPUT name="userfile" type="file">
<INPUT type="submit" value="Отправить файл">
</FORM>
```

ПРИМЕР ФОРМ В HTML ДОКУМЕНТАХ

Начало формы

Поле редактирования

Многострочное поле



Пароль

Флажок

- (1) | - (2)

Переключатель

-(1) -(2) -(3)

Список

Выберите год: ----------

Послать данные | Сброс

Рис.1 Пример формы в HTML документах

Пример (код программы, отображенной на рисунке 1.)

```
<h1>Пример форм в HTML документах</h1>

<form method="post" action="action.php">
<p>Поле редактирования
<br /><input type="text" name="name">
<p>Многострочное поле
<br /><textarea name="text" cols="20" rows="4"></textarea>
<p>Пароль
<br /><input type="password" name="pass">
<p>Флажок
<br /><input type="checkbox" name="c1" value="1-й флажок"> - (1) | <input type="checkbox"
name="c2" value="2-й флажок"> - (2)
<p>Переключатель
<br /><input type="radio" name="r" value="1">-(1) <input type="radio" name="r" value="2">-
(2)  <input type="radio" name="r" value="3">-(3)
<p>Список
<br />Выбирите год:
<select name="sel">
<option value="0">----------</option>
<option value="2000">2000</option>
<option value="2001">2001</option>
<option value="2002">2002</option>
<option value="2003">2003</option>
<option value="2004">2004</option>
<option value="2005">2005</option>
<option value="2006">2006</option>
<option value="2007">2007</option>
<option value="2008">2008</option>
<option value="2009">2009</option>
<option value="2010">2010</option>
</select>
<p><input type="submit" name="submit" value="Послать данные"> | <input type="reset"
name="reset" value="Сброс">
</form>
```

Пример (код программы, обработчика формы рисунка 1. “action.php”)

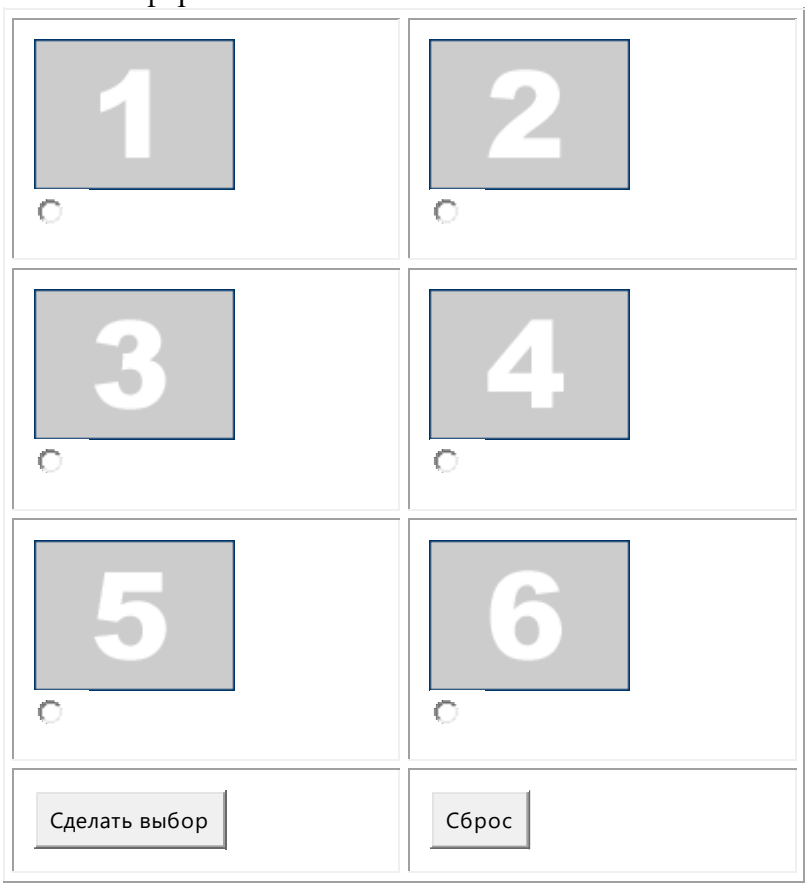
```
<h1>Результат обработки формы</h1>
<?
```

```
echo "<p>Поле редактирования - ".$_POST['name'];
echo "<p>Многострочное поле<br />".$_POST['text'];
echo "<p>Пароль - ".$_POST['pass'];
echo "<p>Флажок";
if ($_POST['c1'] == "" && $_POST['c2'] == "") {
        echo " - Ни один флажок не выбран";
} else {
        if ($_POST['c1'] != "") {
                echo "<p>".$_POST['c1'];
        }
        if ($_POST['c2'] != "") {
                echo "<p>".$_POST['c2'];
        }
}
echo "<p>Значение переключатель ".$_POST['r'];
echo "<p>Список, был выбран - ".$_POST['sel']." год";
?>
```

ПРИМЕР РАБОТЫ С КАРТИНКАМИ И RADIO-КНОПКАМИ

Выберите рисунок

Начало формы



Конец формы

Рис.2. Пример работы с картинками и radio-кнопками

Пример (код программы, отображенной на рисунке 2.)

```
<head>
        <META http-equiv=Content-Type content="text/html; charset=windows-1251">
        <title>Выбор рисунка</title>
</head>
```

```
<body>
<h1>Выбирите рисунок</h1>
<form method="post" action="action.php">
<table border='1' width='400' cellspacing='4' cellpadding='10'>
	<tr>
		<td align='center'>
			<img src='i/1.gif' width='100' height='75' alt='Рисунок #1' border='0' />
			<br /><input type="radio" name="r1" value="1">
		</td>
		<td align='center'>
			<img src='i/2.gif' width='100' height='75' alt='Рисунок #2' border='0' />
			<br /><input type="radio" name="r1" value="2">
		</td>
	</tr>
	<tr>
		<td align='center'>
			<img src='i/3.gif' width='100' height='75' alt='Рисунок #3' border='0' />
			<br /><input type="radio" name="r2" value="3">
		</td>
		<td align='center'>
			<img src='i/4.gif' width='100' height='75' alt='Рисунок #4' border='0' />
			<br /><input type="radio" name="r2" value="4">
		</td>
	</tr>
	<tr>
		<td align='center'>
			<img src='i/5.gif' width='100' height='75' alt='Рисунок #5' border='0' />
			<br /><input type="radio" name="r3" value="5">
		</td>
		<td align='center'>
			<img src='i/6.gif' width='100' height='75' alt='Рисунок #6' border='0' />
			<br /><input type="radio" name="r3" value="6">
		</td>
	</tr>
	<tr>
		<td align='center'><input type="submit" name="submit" value="Сделать
выбор"></td>
		<td align='center'><input type="reset" name="reset" value="Сброс"></td>
	</tr>
</table>
</form>
</body>
</html>
```

Пример (код программы, обработчика формы рисунка 2. “action.php”)

```
<h1>Ваш выбор</h1>
<table border='1' width='200' cellspacing='4' cellpadding='10'>
<?
for ($i=1; $i<4; $i++) {
	$k = "r".$i;
	if ($_POST[$k] == "") {
		$res = "Нет выбора";
	} else {
```

```
$res = "<img src='i/".$_POST[$k].".gif' width='100' height='75' alt='Рисунок #".$_POST[$k]."'
border='0' />";
        }
        echo "<tr><td align='center'>".$res."</td></tr>";
}
?>
</table>
<a href="index.html"> &larr; Назад</a>
```

Раздел 6. Методические рекомендации по СРС:

По дисциплине «Технологии интернет программирования» разработано методическое пособие. «Основы WEB- технологий» авторами Савченко Е.Ю., Мусакулова Ж.А. В данном пособии отражены методические рекомендации к самостоятельной работе студентов изучающих дисциплины «Технологии интернет программирования» и «WEB-ориентированное программирование».