Пользовательские функции.
Вывод таблиц.

Лабораторная работа № А-4.

ЦЕЛЬ РАБОТЫ

Изучение преимуществ и особенностей использования пользовательских функций в языке PHP, закрепление навыков построения простейших алгоритмов обработки информации.

ПРОДОЛЖИТЕЛЬНОСТЬ

2 академических часа (1 занятие)

РЕЗУЛЬТАТ РАБОТЫ

Размещенный на Веб-сервере и доступный по протоколу http документ (страница сайта) с несколькими таблицами.

ДОПОЛНИТЕЛЬНЫЕ ТРАБОВАНИЯ К РАБОТЕ

В начале программы задается массив с не менее 10 разными элементами, содержащие строки формата: "*C1*C2*C3#C4*C5*C6# … Cn*", где C*x* – произвольный текст, символ "*" – разделитель колонок таблицы, символ "#" – разделитель строк. На странице необходимо вывести соответствующие заданным в переменных структурам таблицы. При этом необходимо разработать и использовать функции для:

- вывода HTML-кода таблицы исходя из ее структуры указанного выше формата;
- формирования содержимого отдельной строки таблицы

Кроме того, при выводе таблиц необходимо учитывать следующее.

- Перед выводом каждой новой таблицы необходимо добавить заголовок второго уровня (тег `<h2>`) формата "Таблица №*x*", где *x* – номер выводимой таблицы.
- В начале программы необходимо инициализировать переменную, которая бы содержала число колонок таблиц. Таблицы всегда должны выводится с требуемым числом колонок (возможно с пустыми ячейками в них), вне зависимости от заданной структуры. Если задано нулевое число колонок – таблицы не выводятся, выводится надпись: "Неправильное число колонок".
- Если в строке нет ячеек – ее HTML-код не выводится.
- Если в структуре таблицы нет строк с ячейками – таблица не выводится, выводится надпись: "В таблице нет строк с ячейками".
- Если в структуре таблицы нет строк – таблица не выводится, выводится надпись: "В таблице нет строк".

РЕКОМЕНДАЦИИ К СТРУКТУРЕ ПРОГРАММЫ

Допустим у нас есть только одна переменная со структурой таблицы. Тогда для ее вывода в форме HTML-кода можно использовать следующий укрупненный алгоритм.

- Разбить структуру на элементы, соответствующие строкам таблицы.
- Если строк одна или более, то:
    - начать вывод таблицы;
    - вывести все строки таблицы по одной;
    - закончить вывод таблицы.

Видно, что часто повторяющееся действие – вывод строки таблицы. Это признак того, что его можно оформить в виде отдельной функции, что значительно улучшит наглядность кода, позволит

более просто искать ошибки и изменять его. Напишем такую функцию и назовем ее `getTR()` – в качестве аргумента в нее передается строка вида: "*C1*C2*C3*".

Листинг А-4. 1

```
function getTR( $data )                          // объявление функции
{
     $arr = explode( '*', $data );               // разбиваем строку в массив
     $ret = '<tr>';                         // начинаем тег строки таблицы

     for($i=0; $i<count($arr); $i++)             // цикл по всем ячейкам таблицы
          $ret .= '<td>'.$arr[$i].'</td>';           // добавляем ячейкам тег

     return $ret.'</tr>';                    // возвращаем строку таблицы
}
```

В качестве аргумента функции выступает переменная `$data` – при обращении к ней в теле функции, мы будем получать его значение. Важно помнить, что нет никаких правил для имен переменных аргументов функций – мы вполне могли назвать такую переменную к примеру `$val` или по-другому.

В теле функции переданная строка со структурой разбивается на соответствующие ячейкам таблицы элементы (разделитель элементов – символ "*"), которые сохраняются в массив $arr. Затем последовательно в цикле эти элементы "оборачиваются" тегом `<td>` и добавляются к переменной, содержащей результат работы функции *`$ret`*. Эта переменная инициализируется в самом начале работы функции и содержит открывающийся тег `<tr>`, затем к ней добавляется необходимое количество тегов `<td>`. Закрывающийся тег `</tr>` в переменную не добавляется – он комбинируется с ней непосредственно при возврате результата работы функции.

Теперь, можно реализовать описанный выше алгоритм, используя построенную функцию как его элемент.

Листинг А-4. 2

```
$structure = 'C1*C2*C3#C4*C5*C6';          // структура таблицы

$strings = explode( '#', $structire );          // разбиваем структуру на строки

$datas='';                              // итоговый HTML-код строк
for($i=0; $i<count($strings); $i++ )       // цикл для всех строк из структуры
     $datas .= getTR( $strings[$i] );           // добавляем код строки в итоговый

if( $datas )                                    // если код строк определен
     echo '<table>'.$datas.'</table>';     // выводим таблицу
else                                    // иначе
     echo 'В таблице нет строк ';          // выводим предупреждение
```

Структура определяется в переменной *`$structure`*. Затем, с помощью стандартной функции `explode()`, структура разбивается на отдельные элементы, соответствующие строкам таблицы (признаком окончания такого элемента является символ "#"). В цикле последовательно для каждого из них вызывается функция `getTR()` , которая возвращает соответствующий строке HTML-код. Код накапливается в переменной `$datas` , а затем, если в итоге обработки всех строк, переменная содержит что-нибудь (хотя бы одна строка успешно распознана и преобразована) – выводится тег таблицы <table> и HTML-код строки. Если переменная пуста – выводится соответствующее сообщение.

Код успешно работает, но что делать если надо вывести 5 таких таблиц? Скопировать его и вставить несколько раз, изменив значение переменной *`$structure`*? Да, это сработает – но если

таблиц 500? Чтобы избежать такой проблемы, одинаковые фрагменты кода также оформляют в виде отдельных функций. Сделаем это и мы.

Листинг А-4. 3

```
function outTable($structure)                    // объявление функции
{
      $strings = explode( '#', $structure );        // разбиваем структуру на строки

      $datas='';                                 // итоговый HTML-код строк
      for($i=0; $i<count($strings); $i++ )          // цикл для всех строк
          $datas .= getTR( $strings[$i] );          // добавляем код строки в итоговый

      if( $datas )                               // если код строк определен
          echo '<table>'.$datas.'</table>';      // выводим таблицу
      else                                       // иначе
          echo 'В таблице нет строк ';           // выводим предупреждение
}

outTable('C1*C2*C3#C4*C5*C6');

$structure='C7*C8*C9#C10*C11*C12';
outTable($structure);

$data='C13*C14*C15#C16*C17*C18';
outTable($data);
```

Обратите внимание: эта функция, в отличии от `getTR()`, не возвращает значения, она сама выводит результат. Переменную со структурой таблицы мы перенесли в аргументы функции, что позволило минимизировать правку кода программы. Тогда, для вывода нескольких таблиц не надо копировать весь код – достаточно только скопировать вызовы функции. Кроме того, в примере показана равнозначность передачи аргументов в функцию: можно передавать сразу значение, можно использовать промежуточную переменную, как совпадающую с именем в объявлении функции, так и нет.

Теперь еще больше минимизируем код, сохранив структуры разных таблиц в массиве и вызывая функцию в цикле.

Листинг А-4. 4

```
$structure=array(    'C1*C2*C3#C4*C5*C6', 'C7*C8*C9#C10*C11*C12',
                'C13*C14*C15#C16*C17*C18' );    // массив со структурами таблиц

for($i=0; $i<count($structure); $i++)      // для всех элементов массива
      outTable($structure[$i]);            // выводим соответствующую структуре таблицу
```

Таким образом, использование функций значительно упрощает программирование, уменьшает размер кода, повышает его наглядность.

Самостоятельно доработайте программу так, чтобы она полностью удовлетворяла требованиям лабораторной работы.

СПРАВОЧНАЯ ИНФОРМАЦИЯ

Создание массива	`array();`
	Создает массив из указанных элементов. Например:
	`$list_1 = array('A'); // список из одного элемента`
	`$list_2 = array(1, 2, 3); // список из 3-х элементов`
	`$list_3 = array(); // пустой массив`
	`// ассоциированный массив из двух элементов`
	`$assoc_arr = array('A'=>12, 'D'=>'ff');`

Функция без аргументов	`function f() { ... }  // объявление функции` `f();  // вызов функции`
Функция с одним аргументом	`function f($a) { ... }` `f(10);`
Функция с двумя обычными аргументами	`function f($a, $b) { ... }` `f(10, 'ADC');`

Функция с двумя обычными и двумя аргументами со значением по умолчанию	`function f($a, $b, $c=10, $d=true) { ... }`
	`f(10, 'ADC');  // варианты вызова функции`
	`f(10, 'ADC', 17);`
	`f(10, 'ADC', 17, false);`

Возврат значения функции	`function f($a) { return $a+2; }` `echo f(10); // выводит "12"`

Синтез возврата значения и вывода строки в функции	`function f($a) { echo 'Summa'; return $a+2; }`
	`// выводит "Start Summa 12 End"`
	`echo 'Start '.f(10).' End';`

Разбиение строки на элементы массива с использованием символа-разделителя	`// в массиве элементы [a], [b], [d]`
	`$arr = explode( 's', 'asbsd' );`
	`// в массиве элементы [a], [b], [d], []`
	`$arr = explode( 's', 'asbsds' );`
	`// в массиве элемент [abd]`
	`$arr = explode( 's', 'abd' );`
	`// в массиве элемент [], [ab], [d]`
	`$arr = explode( 's', 'sabsd' );`

КОНТРОЛЬНЫЕ ВОПРОСЫ К ЛАБОРАТОРНОЙ РАБОТЕ

Для успешной защиты работы помимо соответствующего требованиям результата необходимо уверенно отвечать на нижеперечисленные и другие вопросы, а также на контрольные вопросы всех предыдущих лабораторных работ.

1. Что такое пользовательская функция?
2. Как функция возвращает значение?
3. Можно ли вызвать из функции другую функцию?
4. Можно ли вызвать из функции эту же функцию?
5. Продолжит ли функция свою работу после выполнения инструкции `return`?
6. Сколько раз инструкция `return` может быть использована в теле функции?
7. Сколько аргументов может быть передано функции?
8. Что-такое аргументы "по умолчанию"?
9. Может ли функция вообще не иметь аргументов?
10. Должны ли совпадать имена переменных-аргументов при объявлении и при вызове функции?
11. Если в теле функции объявлена переменная $P – изменится ли эта переменная с таким же именем в основной программе? В другой функции? В этой же функции, но вызванной другой раз?
12. В каких случаях имеет смысл использовать пользовательские функции?
13. Может ли функция выводить HTML-код?
14. Может ли функция вызываться непосредственно в операторе `echo`?
15. Может ли функция одновременно выводить текст в HTML-код браузера и возвращать значение?
16. В каком порядке формируется HTML-код, если функция не только вызывается непосредственно из оператора echo, но и выводит что-либо внутри себя?