КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 3

ТЕМА: Функции и масиви в PHP. Оператор foreach.

ЦЕЛ:

Целта на упражнението е студентите да се запознаят със синтаксиса на създаване на функции и масиви чрез PHP. След упражнението студентите би следвало да могат да създават PHP скриптове в които да декларират и използват функции и масиви.

!ВАЖНО: За да можете да тествате упражнението ви е необходим следния софтуер. Web server, Php интерпретатор, MYSQL база данни. Всеки един от тези софтуери ги има в пакетите WAMP или XAMP. Настоящото упражнение е тествано върху WAMP Version 3.2.0.

I. ТЕОРЕТИЧНА ЧАСТ

1. Функции в PHP.

Функциите в езика PHP се декларират по следния начин:

```
function fname ( $arg1, $arg2…)
{
//оператор1; оператор2...
[return израз;]
}
```

Където fname е име на функцията, $arg1, $arg2, … - списък с параметри на функцията, [return израз] – незадължителен оператор, чрез който функцията връща стойността на израза и се прекратява нейното изпълнение.

В PHP е възможно рекурсивно извикване на функцията, т.е. тя да извиква сама себи си. Желателно е името и да съответства на нейното предназначение. Функциите имат следните характеристики:

- Започва винаги с ключова дума function;
- Имената на функциите не са чувствителни към регистъра, но следват същите правила като другите имена в PHP: започват с буква или символа „_“ следвани от цифри, букви или „_“;
- Тялото и е между фигурни скоби „{“ и „}“
- Параметрите се поставят след името в скоби „()“ и се спазват правилата за PHP променливи
- Ако една функция е декларирана вътре в друга, то тя ще бъде достъпна, само ако външната се изпълни поне веднъж.

В резултат от своята работа функцията е възможно да връща адрес на някаква променлива. За целта е нужно, при декларирането на функцията, пред името ѝ да поставим знак амперсанд „&“. Същия знак се поставя и при нейното извикване. Обикновено такива функции връщат:

- адрес на глобална променлива;

- адрес на елемент от глобален масив;
- адрес на статична променлива;
- адрес на един аргумент, ако той се предаде по адрес.

При декларирането на една функция може да се зададат подразбиращи се стойности на формалните параметри. Правилото е ако такива има то те да са последни в списъка с параметри. Подразбиращата се стойност трябва задължително да е константен израз. Задаването и става чрез параметър $sign, за който се задава съответния константен израз.

Пример:

```
function Message($sign="The Rector of the TU-Gabrovo.")
{echo "Dear students, welcome to the Technical University of Gabrovo, Computer Sustem and Technology Department!";
echo $sign ."<br>";
}
....
```

Ако при извикването на функцията не се зададе параметър то „$sign“ ще приеме стойността по подразбиране, а ако се зададе то ще приеме зададената стойност.

Възможно е задаването на статични променливи във функциите. Това са променливи които запазват стойността си след изпълнението на ѝ. Достъпни единствено за самата функция. Декларират се чрез ключова дума static. Инициализират само при първото ѝ изпълнение. Пример:

```
function fn(){
    static $counter = 0;
    $counter++;
}
fn(); fn(); fn();
```

Инициализирането на променливата ще се извърши само първия път и при всяко нейно извикване стойността ѝ ще се увеличава с единица.

2. Масиви в PHP.

Масивът в PHP представлява подредена схема, която асоциира индекси със стойности. Тъй като PHP съхранява масивите под формата на схеми, има възможност те да бъдат използвани за изграждане на най – различни структури от данни, например списъци, дървета, стекове и др.

Масив се изгражда чрез конструкция array и би могъл да наподобява структурата и в останалите програмни езици:

```
$масив_име = array(елемент1,елемент2,....);
```

Това е структурата на един прост масив в PHP чиито елементи могат да бъдат достъпни, като добавим индекс към името му. Както останалите програмни езици и в този случай индексирането започва от 0.

```
$масив_име[0]
```
- достъпва първия елемент на масива

За разлика от другите програмни езици обаче в PHP можем да създадем масив и заедно с това да посочим наш собствен индекс наричани още ключове. Той се задава посредством оператора „=>“. Ключът може да бъде или цяло число или низ, като низовете индекси задължително се ограждат в апострофи или кавички. Структурата на масив със задаването на ключове е следната:

```
$масив_име = array(ключ1=>стойност1, ключ2=>стойност2,.....);
```

Ако в създаването на масив, някой ключ бъде пропуснат, то той ще приеме най – голямата стойност от целочислените индекси и ще я увеличи с единица.

Основни характеристики за ключовете на масив:

- Ако ключ е представен като обикновено цяло число то той ще бъде интерпретиран като такова (т.е. „12“ ще се интерпретира като 12);
- Ако се окаже ключ за който вече има присвоена стойност, то тази стойност ще бъде презаписана.
- Плаващите числа в ключовете се съкращават до цели, ако има следната асоциация 13.3=>“Тони“, то тя ще бъде интерпретирана като 13=>“Тони“.
- Използването на true като ключ ще се инициализира като целочислена 1, а false – като целочислена 0.

3. Оператор за цикъл foreach

Този оператор е специално предвиден за обхождане на масиви. Той разполага с два варианта на синтаксис. Първият е:

***foreach** ($масив_име **as** $променлива_стойност) {блок_оператори}*

Цикълът „foreach“ обхожда масива, който е поставен на мястото на „$масив_име“. Стойността на текущия елемент на масива ще бъде присвоена за „$променлива_стойност“. Накрая индекса се инкрементира с единица, така че следващата итерация на цикъла да обхване следващия елемент на масива. Пример за такъв цикъл е представен в задача 3 от практическата част.

Вторият вариант за синтаксиса е следния:

***foreach** ($масив_име **as** $key=>$променлива_стойност) {блок_оператори}*

Цикълът е подобен на предния вариант, но в допълнение на ключа се присвоява стойността на индекса на текущия елемент.

4. Многомерни масиви

PHP подържа и многомерни масиви. Многомерния масив представлява масив от едномерни масиви. Стойността на един едномерен масив може да бъде всякакъв PHP тип, включително и масиви. Следователно многомерните масиви се получават като в масив влагаме друг масив. Структурата на един двумерен масив е следната:

*$масив_име=**array**(ключ1=>**array**(ключ11=>стойност11,ключ12=>стойност12,...),*
*ключ2=> **array**(ключ21=>стойност21,ключ22=>стойност22,...),...);*

Достъпът до елементи на многомерни масиви се извършва както и в другите езици, като се има в предвид че индексите и ключовете имат едно и също значение.

5. Функции за масиви

В езика PHP съществуват доста функции за работа с масиви. Някои по основни от тях са:

- print_r() – разпечатва всички елементи на масива.
- sort() – сортира стойностите на масива, като не запазва ключовете му
- asort – сортира стойностите на масива, като запазва ключовете на масива

Има още доста на брой полезни функции за работа с масиви които не са обект на разглеждане в настоящото упражнение. С тях може да се запознаете на следния адрес: https://www.php.net/manual/en/function.array.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част са показани примери, чрез които са обяснени нагледно работата с функции и масиви в PHP.

ЗАДАЧА1: *Да се създаде функция ref(), която връща адреса на променлива $name, чиято стойност е „Petter“.Да се създаде друга променлива в която се присвоява резултата върнат от функция ref(). Разпечатайте стойността на двете променливи и анализирайте резултата. Предефинирайте променлива $var с име на друг човек и отново разпечатайте стойността на двете променливи.*

КОД:

```
1   <?php
2   function &ref($par){
3       return $GLOBALS['name'];
4   }
5   $name="Petter";
6   $var=&ref($name);
7   echo "<br>name is ".$name;
8   echo "<br>name is ".$var;
9   $var="Mimi";
10  echo "<br>name is ".$name;
11  echo "<br>name is ".$var;
12  ?>
```

ПОЯСНЕНИЕ

На 6ти ред при извикване на функцията „ref()“ тя връща адреса на променливата $name. Така двете променливи сочат до едно и също място от паметта. Затова резултатът за двете променливи от 7 и 8 ред ще бъде „Petter“, а резултатът им от 10 и 11 ред ще е „Mimi“.

ЗАДАЧА2: *Дефинирайте двумерен масив $arr с ключове „Article“ и „Price“ и стойност за всеки един от ключовете – съответно продукти и цени.*

КОД:

```
<?php
$arr=array("Article"=>array(1=>"Kivi",5=>"Apple",3=>"Orange"),
"Price"=>array(1=>2.35,5=>1.35,3=>1.70));
echo $arr['Article'][1]."-".$arr['Price'][1];
echo "<br>".$arr['Article'][5]."-".$arr['Price'][5];
echo "<br>".$arr['Article'][3]."-".$arr['Price'][3];
?>
```

ПОЯСНЕНИЕ

Стойността на ключ „Article“ е масив в който се съдържа отделни артикули, а за ключ “Price” е масив с цените на артикулите. Вижда се че подредбата на целочислените ключове във вътрешните масиви не е последователна. Ръчното индексиране на отделните стойности на масивите в PHP ни позволява да ги задаваме в произволна последователност. Така тази задачата демонстрира използването на многомерни масиви.

Достъпването на елементите става чрез отделните ключове. При изпълнението на задачата ще се получи съответно кой артикул каква цена има.

ЗАДАЧА3: *Създайте PHP скрипт който да обхожда и отпечатва ключовете и стойностите на един масив. Решете задачата чрез оператор "foreach".*

КОД

```
<?php
    $arrColors=array("R"=>"Red","G"=>"Green","B"=>"Blue");
    foreach($arrColors as $strKey=>$strColor)
        echo "<p>$strKey - $strColor";
?>
```

ПОЯСНЕНИЕ

Задачата демонстрира действието на оператор foreach. Тя създава масив със стойности цветовете, червено, зелено и синьо, чиито ключове са първите букви на цвета. Тялото на foreach е командата echo, която разпечатва съответния ключ и стойност. Понеже тялото на цикъла е само от един оператор то не е необходимо да бъде оградено във фигурни скоби.

III. Задача за самостоятелна работа.

1. Тествайте примерните задачи в практическата част.
2. Създайте PHP скрипт съдържащ функцията recArea, която получава два параметъра за дължина $l и ширина $w на правоъгълник и изчислява лицето му $area. Нека функцията извежда и следния текст в браузъра:

   Rectangle Area Function

   Правоъгълник с дължина 2 и ширина 4 има лице 8.

   Изпълнете скрипта като подадете и други стойности на параметрите.
3. Създайте PHP скрипт с едномерен масив, който използва индекси за да представи потребителски имена и елементи, в които се съхраняват паролата за съответното потребителско име. Създайте променливи, които ръчно им посочвате определено потребителско име и парола. Усъвършенствайте скрипта, така че да проверя потребителското име и парола дали съвпадат. Ако не съвпадат съответно да връща съобщение, че не съвпадат.
4. Създайте PHP скрипт с многомерен масив, състоящ се от редове, които съдържат марка автомобил, цвят и брой коли в наличност. За ключове използвайте низ подсказващ предназначението на съответния ред. Достъпвайте елементите на масива като използвате цикъл foreach.