КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 2

ТЕМА: Управляващи конструкции в PHP. Организиране на програмни цикли. Оператори за включване

ЦЕЛ:

Целта на упражнението е студентите да се запознаят със синтаксиса на операторите, които може да се използват за да се повлияе на хода на изпълнението на скриптовете на PHP. След упражнението студентите би следвало да могат да боравят с операторите за управляващи конструкции, както и с операторите за организиране на програмни цикли.

!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. Оператор if.

Структурата на този оператор има 3 варианта:

- Вариант 1:

if*(израз) блок_за_изпълнение*

- Вариант 2:

if*(израз) блок_за_изпълнение*
else *блок_за_изпълнение1*

- Вариант 3:

if*(израз) блок_за_изпълнение*
elseif*(израз1) блок_за_изпълнение1*
……..
else *блок_за_изпълнениеN*

Израза е всеки правилен PHP израз. В процеса на обработка на скрипта, изразът се преобразува в логически тип. Ако в резултат на преобразуване стойността на израза е истина (true), то се изпълнява блок_за_изпълнение. В противен случай този блок е игнорира. Ако блока за изпълнение съдържа няколко команди или оператори, то той трябва да бъде ограден във фигурни скоби {}.

Конструкция „else“ в оператор „if“ има същото предназначение като другите програмни езици, а именно блок_за_изпълнение1 ще се изпълни когато твърдението в израза е грешно, т.е. има стойност false. Особеното в езика PHP е че позволява използването на конструкция „elseif“, която представлява комбинация между „else“ и „if“. Тя прилича на конструкцията „else“, допълвайки я с конструкцията “if” с отделен израз, който да бъде изпълнен, в случай че твърдението за израза от първата конструкция „if“ не е вярно.

Правила за преобразуване на израза в логически тип: За FALSE се считат:

- Логическа стойност False;
- Целочислена нула (0);
- Реална нула (0);
- Празен стринг и стринг“0“;
- Масив без елементи;
- Тип NULL.

Всички останали значения се преобразуват в стройност True.

2. Оператор switch

Конструкцията „switch“ наподобява поредица от конструкции „if“, управлявани от един общ израз. Тя се използва когато искаме да сравним стойността на дадена променлива спрямо други стойности и по този начин да бъде изпълнен различен програмен код. Синтаксисът на конструкцията „switch“ е следният:

switch *(израз){*
case *значение1**:** блок_от_оператори1 break;*
case *значение2**:** блок_от_оператори2 break;*
…….
default: *блок_от_оператори_по_подразбиране*
}

За разлика от “if” тук израза не се преобразува към логически тип а се сравнява с поредица от константни изрази след конструкции „case“ (значение1, значение2, и т.н.). Ако стойността на израза съвпада с някой вариант то се изпълнява съответстващия блок от оператори – от двоеточието до края на switch или до първото срещане на оператор break, ако такъв се намери. Ако израза не съвпада с нито един от вариантите, то се изпълнява блок_от_оператори_по_подразбиране намиращ се след конструкция „default“.

Логиката на конструкцията „switch“ би могла да бъде реализирана и чрез поредица от изрази „if“, но тогава кодът би станал много сложен и труден за разбиране.

3. Оператор за цикъл While

„While“ е най-простият вид цикъл в PHP, но това съвсем не означава, че не е мощен. Базовият синтаксис на цикъла е следният:

while *(израз) { блок_за_изпълнение }*

Логиката на конструкцията е следната – командите в блок_за_изпълнение се изпълняват докато резултата от израза е истина (true). И тук както в конструкция „if“, израза се привежда в логически тип. В началото на всяка итерация се проверя дали стойността на израза е истина. Ако стойността е true се изпълнява блок_за_изпълнение. Ако стойността е false, цикъла се прекратява и управлението се предава на следващия оператор след „while“.

4. Оператор за цикъл do-while

Цикълът „do-while“ е много подобен на „while“ с тази разлика, че истинността на израза се проверява след края на всяка итерация. Благодарение на това блок_за_изпълнение ще се изпълни поне веднъж. Синтаксиса е следния:

do {блок_за_изпълнение} **while** (израз) ;

5. Оператор за цикъл for

Това е един от най – често използваните оператори и на пръв поглед е малко по сложен от предходните оператори за цикли. Синтаксисът му е следният:

for*(израз1;израз2;израз3) { блок_за_изпълнение }*

Първият израз служи да инициализира цикъла. Той се изпълнява само веднъж – в самото начало. Вторият израз съдържа условие. Той контролира колко пъти да се изпълни цикъла. Третият израз обикновено се използва за да се инкрементира, декрементира променливата, от която зависи броя на итерациите в цикъла. Действието е следното Първо се изпълнява израз1, след което се проверява условието в израз2. Ако то е истина се изпълнява тялото на цикъла (блок_за_изпълнение), след което се изпълнява израз3 и връщане към израз2. Т.е. в началото на всяка итерация се изпълнява условието в израз2. Ако то е грешно се прекратява действието на цикъла и управлението се предава към следващия оператор след цикъл for.

Всеки от изразите може да бъде празен (т.е. да ги няма в структурата на for). Ако изрази 1 и 3 са празни то те трябва да се поставят на съответните места така че да не се променя логическия смисъл на решаваната задача. Ако израз 2 е празен, то това значи че цикъла ще се изпълнява неопределено време (в този случай PHP счита този израз за истинен винаги. Това не е така безполезно, понеже цикъла може да бъде спрян с оператор break.

6. Оператори за предаване на управление – break и continue

Понякога е необходимо да се излезе от цикъл преди условието за край да е приеме стойност false. В такъв случай се използва оператор break. Той завършва изпълнението на текущия цикъл, било то for, while, do-while или foreach.Последният е наличен само в езика PHP и ще бъде разгледан в следващото упражнение.

Понякога е нужно не напълно да се прекрати работата на текущия цикъл а само да започне неговата нова итерация. Това се извършва чрез оператор continue.

7. Оператори за включване – include и require

Оператор „include“ се използва за включване на код, съдържащ се в указан файл и изпълнението му толкова пъти колкото програмата среща този оператор. Включването може да стане по един от следните начини:

- include ‘име_на_файл’; **Пример:** *include ‘params.php’;*
- include име_променлива; **Пример:** *$file_name=’params.php’; include $file_name;*
- include(“име_на_файл”); **Пример:** *include (“params.php”);*

Оператор „require“ действа по същия начин като оператор „include“.Основната разлика между двата е при отношението на възникване на грешка: при грешка (липсва на файл например) оператор „include“ извежда предупреждение, но работата на скрипта продължава. Докато грешка при оператор „require“ предизвиква фатална грешка и изпълнението на скрипта се прекратява. Освен това условните оператори и цикли не влияят на оператор “require”. Синтаксиса на оператора е следния:

require(“име_на_файл”);

8. Алтернативен синтаксис за някои оператори на PHP

Езика PHP предлага алтернативен синтаксис за някои свои управляващи структури, а именно за операторите „if“, „while“, „for“, „foreach“ и „switch“. За всеки от тях алтернативата се дължи на това че отварящата фигурна скоба „{“ се заменя с двоеточие

„:“, а затварящата съответно с – „endif“, „endwhile“, и т.н. Смисъла и действието на операторите си остава същото. Например алтернативния синтаксис на оператор „for“ е следния:

for**(израз1;израз2;израз3) **:** блок_за_изпълнение **endfor

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част са показани примери чрез които са обяснени нагледно някои от операторите в теоретичната част.

***ЗАДАЧА1:** Да се декларират две променливи от целочислен тип с произволни стойности. Нека едната от променливи целочислената и стойност бъде декларирана като стринг. Да се напише скрипт който да извежда коя от двете стойности е по голяма.*

КОД:

```
<?php
$intNumber=100;
$strNumber="80";
if($intNumber>$strNumber) echo "<p>$intNumber is larger than $strNumber</p>";
elseif ($intNumber==$strNumber) echo "<p>$intNumber is equal to $strNumber</p>";
else echo "<p>$intNumber is small than $strNumber</p>";
?>
```

ПОЯСНЕНИЕ

Забележете че в PHP няма значение че едната променлива е число а другата низ. Проверката се реализира успешно. Резултата от изпълнението ще покаже съобщението „100 is larger than 80“. Може да тествате няколко пъти този пример като смените стойностите на променливите.

***ЗАДАЧА2:** Създайте php скрипт който да извежда името на сезона („пролет“, „лято“, „есен“, „зима“) по зададен номер на месец. Реализирайте задачата като използвате алтернативен синтаксис на съответния оператор.*

КОД:

```
<?php
   $month=10;
   switch ($month):
     case 3: case 4: case 5: echo "Сезона е пролет"; break;
     case 6: case 7: case 8: echo "Сезона е лято"; break;
     case 9: case 10: case 11: echo "Сезона е есен"; break;
     case 12: case 1: case 2: echo "Сезона е зима"; break;
     default: echo "$month е невалиден номер за месец";
   endswitch;
?>
```

ПОЯСНЕНИЕ

Задачата демонстрира използването на оператор switch чрез алтернативния му синтаксис. Резултата от изпълнението на кода ще бъде „Сезона е есен“.

ЗАДАЧА3: *Създайте PHP скрипт който да отпечатва всички четни числа между 1 и 9. Решете задачата чрез операторите "while", "do-while" и "for".*

КОД

```
<?php
//Решение чрез оператор while;
    $i=1;
    while($i<10):
        if($i%2==0) print $i." ";
        $i++;
    endwhile;
//Решение чрез оператор do-while
    $i=1;
    do {
        if($i%2==0) print $i." ";
        $i++;
    } while($i<10);
//Решение чрез оператор for
    for($i=1;$i<10;$i++)
        if($i%2==0) print $i." ";
?>
```

ПОЯСНЕНИЕ

Задачата разпечатва четните числа между 1 и 10, а именно 2 4 6 и 8, като е решена по три начина. Откриването на четно число става чрез деление на две. Ако остатъка е нула числото е четно.

Както се забелязва за отпечатването на числата е използвана командата print. Това е така само за да демонстрира как работи тя. За първия начин е използван алтернативния синтаксис на оператор while. При решението чрез оператор for тялото на цикъла се състои само от един оператор if затова то не е оградено във фигурни скоби

III. Задача за самостоятелна работа.

1. Тествайте примерите задачите в практическата част.
2. Редактирайте решението чрез цикъл for на задача 3 от практическата част, така че в структурата на for да се пропусне израз2.
3. Редактирайте задача 1 от практическата част така че стойността на променливите да се съхранява в друг файл. Реализирайте задачата като използвате в операторите за включване за да включите файла със променливите в основния файл.
4. Създайте скрипт, който използва цикъла „while“, за да изпише на екрана таблицата за умножение по 2.

   1 x 2 = 2

   2 x 2 = 4

   …….

   10 x 2 = 20

   Направете скрипта си достатъчно гъвкав, така че да можете просто да смените стойността на някоя променлива, за да покаже таблицата за умножение по друго число.