МАТЬО ДИНЕВ
ВАЛЕНТИНА КУКЕНСКА

ИНФОРМАЦИОННИ СИСТЕМИ

Ръководство за лабораторни упражнения

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
ГАБРОВО
2024

МАТЬО ДИНЕВ
ВАЛЕНТИНА КУКЕНСКА

ИНФОРМАЦИОННИ СИСТЕМИ

Ръководство за
лабораторни упражнения

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
"ВАСИЛ АПРИЛОВ"
ГАБРОВО, 2024

ИНФОРМАЦИОННИ СИСТЕМИ

Ръководство за лабораторни упражнения

Университетско издателство “Васил Априлов” – Габрово, 2024

ISBN: 978-954-683-718-9

СЪДЪРЖАНИЕ

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 1. Основни елементи в PHP. Синтаксис на езика

ЦЕЛ

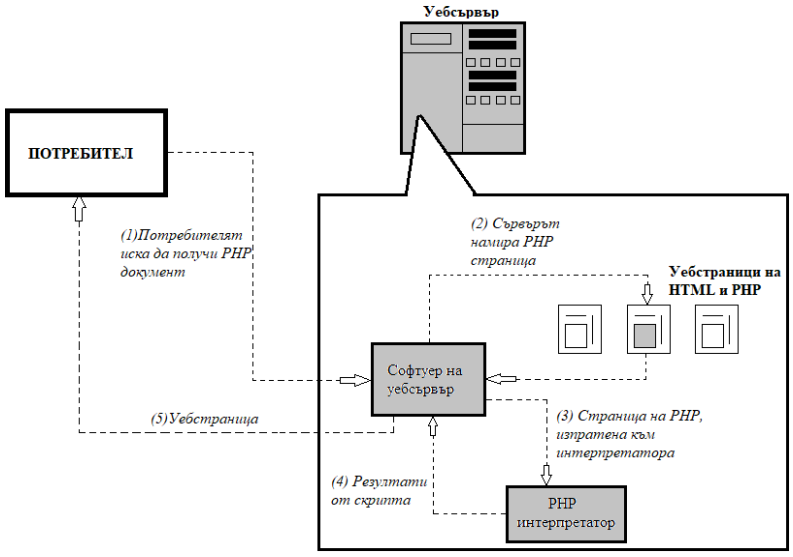
Запознаване на студентите с основните елементи на езика за програмиране PHP. Придобиване на практически опит за дефиниране и използване на променливи и константи, както и за представяне на изрази чрез PHP скриптове.

I. ТЕОРЕТИЧНА ЧАСТ

PHP е популярен скриптов език, който може да бъде вграждан в текст на HTML. Той е широко приложим и особено подходящ за създаването на динамични уебразработки.

PHP в комбинация с (X)HTML позволява създаването на динамични уебсистеми. Стандартният (X)HTML документ се съхранява на сървъра. При подаването на заявка от клиент съдържанието му директно се визуализира в уеббраузъра. Самият документ не се променя. Той може да съдържа код на JavaScript или приставка, които осигуряват взаимодействие с потребителя. JavaScript е език за програмиране предимно от страна на клиента и не е подходящ да извършва промени в съдържанието на документа или да осъществя взаимодействие с базата от данни.

PHP е сървърен скриптов език за обработката на хипертекстови документи преди те да бъдат подадени към браузъра. Потребителят заявява PHP документ. Уебсървърът го изпраща към PHP интерпретатор, който анализира текста на документа. След откриването и изпълнението на командата, съответният документ се изпраща към браузъра на потребителя.



Фиг. 1.1 Комуникация между потребител и уебсървър

За създаването на PHP скрипт се използва текстов или скриптов редактор. Написаният код се записва във файл с разширение „.php“. За изпълнението му е необходимо той да се постави в директорията на уебсървъра. За WAMP пакета това е:

C:\wamp\www

В тази директория могат да се създават отделни поддиректории в които да се съхраняват файловете от отделни проекти. За изпълнението на файла е необходимо в адресната лента на браузъра да се въведе:

localhost/(поддиректория на съответния проект)/(име на файла)

В HTML кода могат да се вграждат много PHP скриптове. За целта се използват специални тагове които маркират началото и края на скрипта.

<php

....

?>

В PHP скрипта може да се вграждат HTML тагове като се използват апострофи и кавички. Това е показано в задача 1 от практическата част.

1. Команди echo и print

Командите **print** и **echo** се използват за да връщат стойност. За техния синтаксис са важни следните условия:

- Когато връщаната стойност е символен низ, то той се поставя в кавички или апостроф. Ако се използват кавичките, то може да се включат и имена на променливи в съответния низ. Ако низа е ограден с апостроф и се включат имената на променливите, то резултата ще разпечата съответното име на променливата;
- Когато връщаната стойност е от числов формат кавички и апострофи не се поставят;
- Точката (.) в операторите echo и print е оператор за конкатениране.

В символните низове може да се използват и скоби заедно с кавичките и апострофите.

Примери:

echo “Hello World”; *// ще изведе съобщението “Hello World” на екрана*

echo (“10-7=”).(10-7); *// ще изведе 10-7=3*

2. Операции в PHP

Поддържаните операции и техните означения в PHP са:

- аритметични:
  - ✓ събиране (+);
  - ✓ изваждане (-);
  - ✓ умножение (*);
  - ✓ деление (/);
  - ✓ деление по модул (%);
- логически:
  - ✓ логическо И – (&&) или and;

- ✓ логическо ИЛИ – (||) или or;
- ✓ логическо отрицание – (!);

- побитови:
  - ✓ побитово И – (&);
  - ✓ побитово ИЛИ – (|);
  - ✓ изключващо или – (^);
  - ✓ побитово отрицание – (~);
  - ✓ побитово изместване на ляво – (<<);
  - ✓ побитово изместване на дясно – (>>);
- за сравнение – ==, != или <>, >, <, >=, <=;
- за присвояване – =, +=, -=, *=, /=, %=;

3. Константи в PHP

Константите в PHP се декларират с функция **define()**. Тя има следната структура: ***define****("Име","Стойност"),* където:

- Име - низ отговарящ за името на съответната константа. Името трябва да започва с буква или долна черта и да съдържа произволен брой букви, цифри или долни черти;
- Стойност – валиден PHP израз, отговарящ за стойността на съответната константа;

ЗАБЕЛЕЖКА! При изписване на имената на константите има значение употребата на главни и малки букви. Стандартно те се изписват изцяло с главни букви.

За проверка дали дадена константа е дефинирана се използва функцията ***defined****(„Име")*. Тя има само един аргумент отговарящ на името на константата. Функцията връща булева стойност ***true*** или ***false*** в зависимост от това дали константата е дефинирана или не.

Примери:

```
define("PASSWORD","querty");
echo "PASSWORD=".PASSWORD;       //ще изведе PASSWORD=querty
echo "PASSWORD=".password   // ще изведе грешка понеже няма такава константа
```

4. Променливи в PHP

Променливите в PHP не се декларират явно. Типа им зависи от присвоената стойност. Пред променливите в PHP се използва знака $ и могат да съдържат долна черта, букви или цифри.

ЗАБЕЛЕЖКА! При изписване на имената на променливите (също като при константите) има значение употребата на главни и малки букви. Например променливата „$name е различна от променливата $Name.

Инициализацията на променливите не е задължителна. Стойността на неинициализираните променливи зависи от контекста в който се използват. За

променливите от логически тип подразбиращата се стойност е ***false***, за числов тип (*integer*, *float*) – 0, за тип string – празен низ, за масив (тип array) – празен масив.

Инициализиране на променливите може да се прави:

- със стойност ($y=5;)
- с адрес($y=&$x)
- индиректно – съдържанието на променливата се използва като име на променлива ($x=”Peter”; $$x=”John”; - така се създават две променливи първата е с име x и стойност Peter, а втората е с име Peter и стойност John)

Индиректно инициализиране на променливите не се среща в другите програмни езици и е едно от големите предимства на PHP.

Типа на променливата може да се провери чрез функцията **gettype**(„*име“*). Като резултат от нейното изпълнение се получава типа на променливата зададена като аргумент.

Пример:

$my_var=”Georgi”; Echo gettype(my_var); //ще изведе string
$my_var=88.9; Echo gettype(my_var); //ще изведе double
$my_var=8; Echo gettype(my_var); //ще изведе integer
$my_var=null; Echo gettype(my_var); //ще изведе NULL

За управление на променливите в PHP се използват следните функции:

- ***isset****(var)* – връща ***true***, ако променливата *var* е установена и не е ***NULL***, в противен случай връща ***false***;
- ***empty****(var)* – връща ***true***, ако променливата *var* не е установена, т.е. ако няма стойност, както и когато променливата приема някоя от следните стойности: *„0“, „0.0“, „ “,* ***FALSE****,* ***NULL****, празен масив*.
- ***unset****(var1,var2,…)* – унищожава се стойността на специфицираната променлива. Поведението на функцията варира в зависимост от това какъв тип променлива ще се унищожава.

Примери за използването на тези функции са отразени в някои от следващите упражнения.

В PHP съществуват променливи чрез които може да се изведе специфична информация за посетителя на една страница, като: уеб адрес на потребителя, IP адреса му, каква операционна система и браузър използва и др. Тези променливи се наричат променливи **в обкръжението.**

- $_SERVER[‘HTTP_USER_AGENT’] - дава информация за браузъра и ОС
- $_SERVER[‘REMOTE_ADDR’] - дава информация за IP адреса
- $_SERVER[‘SERVER_SOFTWARE’] - дава информация за сървъра
- $_SERVER[‘HTTP_REFERER‘] - дава информация за URL, откъдето идва user-a.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част са показани примери, чрез които са обяснени нагледно отделните елементи в PHP. За целта са решени следните задачи:

ЗАДАЧА 1: *Създайте PHP страница, със стандартни HTML тагове <head>, <title> и <body> tags. Запишете скриптовия файл в директорията на уебсървъра с разширение .php. Напишете скрипта, така че страницата да има заглавие "Hello Boys and Girls!" и да изведе съобщението „Hello World" чрез PHP код.*

КОД

```
<html>
    <head>
        <title><?php echo "Hello, Boys and Girls!";  ?></title>
    </head>
    <body>
        <?php
            echo "Hello, World!";
        ?>
    </body>
</html>
```

Горният код може да служи за пример как чрез команда **echo** се извежда символен низ на екрана. Първата част от кода създава заглавието на html страницата, а втората служи за извеждане на съобщението.

ЗАДАЧА 2: *Създайте PHP скрипт, който изчислява сумата, разликата, произведението, частното и остатъка от делението на две променливи със стойности $x=10 и $y=7. Резултатът от действията да се представи по следния начин.*

Сумата е: 10+7=17
Разликата е: 10-7=3
*Произведението е:10*7=70*
Частното е: 10/7=1.4285714285714
Остатъка е: 10%7=3

КОД

```
<?php
    $x=10;
    $y=7;
    $result=$x+$y;
    echo "<br> Сумата е: $x+$y=$result";
    $result=$x-$y;
    echo "<br> Разликата е: $x-$y=$result";
    $result=$x*$y;
    echo "<br> Произведението е: $x*$y=$result";
    $result=$x/$y;
    echo "<br> Частното е: $x/$y=$result";
```

```
    $result=$x%$y;
    echo "<br> Остатъка е: $x%$y=$result";
?>
```

ЗАДАЧА 3: *Напишете PHP скрипт който да извлича следната информация за потребителя: име на сървъра, ip адреса му, информация за уеб сървъра, за неговия браузър и операционна система, които използва.*

КОД

```
<?php
    echo "SERVER_NAME=".$_SERVER['SERVER_NAME'];
    echo "<br>";
    echo "SERVER_ADDR=".$_SERVER['SERVER_ADDR'] ;
    echo "<br>";
    $x=$_SERVER['SERVER_SOFTWARE'];
    echo "Вашия уеб сървър е: $x";
    echo "<br>";
    echo "Вашите браузър и операционна система ca:".$_SERVER['HTTP_USER_AGENT'];
    echo "<br>";
?>
```

Решението на последните две задачи демонстрира различни варианти за извеждане на стойност от променлива чрез команда **echo**. При поставянето на променливата в символния низ за извеждане се достъпва нейната стойност.

Горните скриптове е необходимо да бъдат записани в файлове с разширение „*.php*“ в уеб директорията на сървъра. За изпълнението е необходимо да бъдат заредени в адресната лента на уеббраузъра. Например, ако един скриптов файл има наименование „*example1.php*“, то в браузъра е необходимо да се въведе следния адрес: *localhost/example1.php*.

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Разгледайте и изпълнете задачите от практическата част.
2. Създайте PHP скрипт, който да дефинира константа „*AUTHOR*“ със стойност „*Simon Stibart*“. Достъпете я с малки и с главни букви. Анализирайте получените резултати.
3. Създайте PHP скрипт в който се декларира една променлива с име *$num* и стойност *8*. Извършете необходимите операции върху променливата, така че да се визуализира следната информация.
   Value is now 8.
   Add 2. Value is now 10.
   Subtract 4. Value is now 6.
   Multiply by 5. Value is now 30.
   Divide by 3. Value is now 10.

Increment value by one. Value is now 11.
Decrement value by one. Value is now 10.

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Какво е предназначението на PHP интерпретатора?
2. Какво се подава към първия параметър на функция **define()**?
3. От какво зависи типа на една променлива в PHP?
4. С коя функция се разбира дали една променлива е установена?
5. За какво служат променливите в обкръжението?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 2. Управляващи конструкции. Организиране на програмни цикли. Оператори за включване

ЦЕЛ

Запознаване на студентите със синтаксиса на операторите, чрез които може да се повлияе на хода на изпълнението на PHP скриптовете. Придобиване на практически умения за работа с операторите за управляващи конструкции и използване програмни цикли.

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;*

Изразът се преобразува в логически тип в процеса на обработка на скрипта. Ако в резултат на преобразуване стойността на израза е истина ***true***, то се изпълнява *блок_за_изпълнение*. В противен случай този блок се 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**.

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

- Логическа стойност ***false***;
- Целочислена нула (*0*);
- Реална нула (*0.0*);
- 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 за сравнение на стойността на дадена променлива с 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 изразът не се преобразува към логически тип, а се сравнява с поредица от константни изрази след **case** (*значение1*, *значение2*, и т.н.). Ако стойността на израза съвпада с някой вариант, то се изпълнява съответстващият блок от оператори – от двоеточието на съответния **case** до края на конструкцията **switch**, или до първото срещане на оператор **break**. Ако израза не съвпада с нито един от вариантите, то се изпълнява *блок_от_оператори_по_подразбиране* намиращ се след конструкция **default**.

Логиката на **switch** би могла да бъде реализирана и чрез поредица от конструкции **if**, но тогава кодът би станал по-сложен и труден за разбиране.

3. Оператор за цикъл While

Синтаксисът на оператора за цикъл е следният:

while *(израз) {блок_за_изпълнение;}*

Командите в *блок_за_изпълнение* се изпълняват докато резултата от *израза* е ***true***. Подобно на конструкция **if**, *изразът* се преобразува 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***, действието на цикъла се прекратява и управлението се предава на следващия оператор след **while**.

4. Оператор за цикъл do-while

Операторът за цикъл **do-while** е подобен на **while** с разликата, че истинността на *израза* се проверява след края на всяка итерация. Така блокът за изпълнение ще се изпълни поне веднъж. Синтаксисът е следния:

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

5. Оператор за цикъл for

Операторът за цикъл **for** съдържа три израза в условието си. Синтаксисът му е следният:

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

Първият израз служи да инициализира цикъла. Той се изпълнява само веднъж в началото. Вторият израз съдържа самото условие. Той контролира колко пъти да се изпълни цикълът. Третият израз обикновено се използва за да се инкрементира или декрементира променливата, от която зависи броя на итерациите в цикъла. Действието на оператора **for** 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*. В началото на всяка следваща итерация се изпълнява условието в *израз2*. Ако не е изпълнено се прекратява действието на цикъла и управлението се предава към следващия оператор след **for**.

Всеки от изразите може да бъде пропуснат. Ако *израз1* и *израз3* липсват, то те трябва да бъдат поставени на определени места в тялото на цикъла, така че да не се променя логическия смисъл на решаваната задача. Ако *израз2* е пропуснат, то неговата стойност се приема за **true**. Поради това действието на цикъла ще се изпълнява неопределен брой пъти (получава се безкраен цикъл). В някои случаи това би могло да бъде полезно, но е необходимо в един момент действието на цикъл **for** да бъде спряно с оператор **break**.

6. Оператори за предаване на управление – break и continue

Оператор **break** се използва за прекратяване действието на някои оператори. В точка 2 се забелязва как се приключва действието на конструкция **switch**, след края на съответния **case** израз. Освен при конструкция **switch**, **break** може да се използва и за предварително прекратяване действието на операторите за цикъл.

Оператор **continue** се използва за приключване на текущата итерация и преминаване към следващата.

7. Оператори за включване – include и require

Оператор **include** се използва за включване на код, съдържащ се в указания файл. Дефинирането му 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 възникване на грешка. Една от честите причини за това е липсата на файл който се достъпва. В този случай оператор **include** ще изведе предупреждение и ще продължи изпълнението на останалата част от скрипта. При оператор **require** грешката е фатална и се прекратява изпълнението на скриптовия файл. Синтаксисът на оператор require 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**, **switch**. За всеки от тях алтернативата се отразява със замяна на фигурна скоба „{“ с двоеточие „:“, а затварящата – с „endif“,

„endwhile“, и т.н. Смисълът 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 за използване на операторите от теоретичната част.

***ЗАДАЧА1:** Декларирайте две променливи от целочислен тип с произволни стойности. Задайте стойността на едната целочислена променлива като низ. Напишете 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>";
?>
```

В PHP няма значение, това че едната променлива е зададена като целочислена а другата като низ. Проверката се реализира успешно. Резултатът от изпълнението на горния скрипт ще изведе съобщението „*100 is larger than 80*“.

***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;
?>
```

Задачата демонстрира използването на оператор **switch** чрез алтернативния му синтаксис. Резултатът от изпълнението на кода е „*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." ";
?>
```

За изпълнение на задачата са използвани 3 различни оператора за цикъл. За отпечатването на числата е използвана командата **print**. За първия вариант на кода е използван алтернативния синтаксис на оператор **while**.

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ.

1. Разгледайте и изпълнете задачите от практическата част.
2. Редактирайте решението 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 декларацията на променливите да се съхранява в друг файл. Реализирайте задачата като използвате операторите за включване за да достъпите променливите от другия файл.
4. Като използвате цикъл **while**, създайте скрипт реализиращ таблицата за умножение по 2. Направете го достатъчно гъвкав, за да може при промяна стойността на дадена променлива да се реализира таблицата за умножение по друго число.
   1 x 2 = 2
   2 x 2 = 4
   …….
   10 x 2 = 20

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. За какво служи конструкция **elseif**?
2. Кой оператор реализира цикъл със след-условие?

3. Кой от изразите в условието за цикъл **for** се изпълнява само веднъж?
4. Какво е предназначението на оператор **continue**?
5. Какво е предназначението на оператор **require**?
6. С какво се заменя символа **„{“** при алтернативния синтаксис на операторите?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 3. Функции и масиви в PHP. Оператор foreach.

ЦЕЛ

Запознаване на студентите със синтаксиса на функциите и масивите в езика за програмиране PHP. Придобиване на практически опит за деклариране и използване на функции и масиви.

I. ТЕОРЕТИЧНА ЧАСТ

1. Функции в PHP

Функциите представляват част от скрипт, който се изпълнява само при извикване. Ако една функция не е извикана, кодът в нея никога няма да се изпълни. В PHP е възможно и рекурсивно ѝ извикване, т.е. тя да извика сами себе си. За създаването на функциите се използва ключова дума **function** и се декларират по следния начин:

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

Всяка функция се състои от име *fname*, списък с аргументи (*$arg1*, *$arg2)* и тяло оградено във фигурни скоби.

Функциите имат следните характеристики:

- Започват винаги с ключова дума **function**;
- Имената им не са чувствителни към регистъра. Могат да започват с буква или символа „_“ следвани от цифри, букви или „_“;
- Тялото им е между фигурни скоби „{“ и „}“;
- Списъкът с аргументи се поставя в скобите „()“ след името и се спазват правилата за деклариране на PHP променливи;
- Извикването им се извършва чрез техните имена и списъка им с аргументи;
- Декларираните променливи в тях са достъпни само за самата функция.
- Променливите извън тях не са достъпни в самата функция.
- Ако една функция е декларирана вътре в друга, то тя може да бъде достъпна само, ако външната е извикана поне веднъж.

Всяка функция може да връща резултат, чрез оператор **return**. Върнатият резултат може да бъде адрес на някаква променлива. За целта е нужно, при декларирането на функцията, пред името ѝ да поставим знак „&“. Същият 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 деклариране на една функция, може да се зададат подразбиращи се стойности на аргументите. Това позволява при извикването ѝ да не се задават стойности на всички аргументи. Правилото е при декларацията, аргументите с подразбираща се стойност да се подават последни в списъка.

Пример:

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

При извикването на функцията, ако не се зададе стойност на аргумента „$sign“, то той ще приеме стойността по подразбиране. В противен случай ще приеме зададената стойност.

Възможно е задаването на статични променливи във функциите. Те запазват стойността си след изпълнението им. Декларират се чрез ключова дума **static**. Инициализират се само при първото ѝ изпълнение. Пример:

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

Инициализирането на променливата се извършва само при първия достъп на функцията. При всяко нейно извикване стойността ѝ ще се увеличава с единица.

2. Масиви в PHP

Масивът в PHP представлява подредена структура от данни, която е индексирана. Тя може да се използва за изграждане на списъци, дървета, стекове и др.

Масивът се описва чрез конструкция **array**. Най-простата му структура е:

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

Това е структурата на един масив, чиито елементи могат да бъдат достъпни чрез добавяне на индекс към името му. Индексирането започва от 0.

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

В PHP при създаване на масив може да се посочи собствен индекс (ключ). Той се задава посредством оператора „=>“. Ключът може да бъде 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 приема най-големия свободен целочислен индекс.

Основни характеристики за ключовете са:

- Ако ключ е представен като обикновено цяло число, то той се интерпретиран като такова (т.е. „12“ ще се интерпретира като 12);
- Ако се дефинира ключ, за който вече е присвоена стойност, то старата стойност се презаписва с новата;
- Дробните 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, то той се инициализира като целочислена 1, а ***false*** – 0.

3. Оператор за цикъл foreach

Този оператор се прилага за обхождане на масиви. Той може да се дефинира по два варианта. Първият е:

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

Операторът за цикълът **foreach** обхожда масива посочен в *$масив_име*. При всяка итерация стойността на текущия елемент от масива се присвоява на *$променлива_стойност*. Пример за този вариант е представен в задача 3 от практическата част.

Вторият вариант за синтаксиса е следния:

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

При него се обработват и ключовете.

4. Многомерни масиви

Многомерният масив представлява структура от едномерни масиви. Задаването на един двумерен масив е следното:

*$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 програмни езици.

5. Функции за масиви

В езика PHP съществуват различни функции за работа с масиви. Някои 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;

За допълнителна информация за функции за работа с масиви е достъпна на адрес: https://www.php.net/manual/en/function.array.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част са показани примери за използване на функции и масиви в PHP.

ЗАДАЧА1: *Създайте функция ref(), която връща адреса на променлива $name, със стойност „Petter“. Декларирайте променлива $var, в която се присвоява резултата върнат от функция ref(). Изведете стойността на двете променливи и анализирайте резултата. Предефинирайте променлива $var с друго име и отново изведете 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      ?>
```

При извикване на функцията „*ref()*“ се връща адреса на променливата *$name*. Той се присвоява за *$var*. Двете променливи сочат един и същи адрес от паметта. Резултатът за двете променливи ще бъде „*Petter*“ (ред 7 и 8) и „*Mimi*“ (ред 10 и 11).

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];
?>
```

Задачата демонстрира използването на многомерни масиви. Достъпването на елементите става чрез отделните ключове. При изпълнението на задачата се визуализират цените на артикулите.

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

КОД

```
<?php
```

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

Създава се масив с елементи чиито стойности са цветовете: червено, зелено и синьо, а ключовете са първите букви на цвета. В тялото на оператор **foreach** е командата **echo**, която извежда като резултат съответния ключ и стойността за него. Понеже тялото на цикъла се състои само от един оператор, позволява да не се използват фигурни скоби.

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Разгледайте и изпълнете задачите от практическата част.
2. Създайте PHP скрипт съдържащ функция с наименование *recArea*, в която се задават два аргумента 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 см и ширина 4 см има лице 8 см.

   Изпълнете скрипта като подадете и други стойности за аргументите.
3. Създайте PHP скрипт с едномерен масив, който съдържа потребителски имена и пароли към тях. Създайте променливи, за които ръчно им посочвате определено потребителско име и парола. Усъвършенствайте скрипта, така че да проверя стойностите им дали съвпадат с някои от елементите на масива.

   Указание: *Направете така, че ключовете на елементите от масива да се състоят от потребителските имена, а стойностите към тях от – техните пароли.*
4. Създайте PHP скрипт с двумерен 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 оператор **foreach**.

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. С какво започват функциите в PHP?
2. Как се достъпват вложените една в друга функции?
3. Възможно ли е функциите в PHP да връщат резултат? Ако да как и от какъв тип е той?
4. За какво служат ключовете на масивите в PHP?
5. Ако един ключ представлява дробно число, как ще се интерпретира той?
6. Как се декларират многомерни масиви в PHP?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 4. PHP като обектно ориентиран език

ЦЕЛ

Запознаване на студентите с PHP като обектно ориентиран език. Придобиване на практически знания и умения за реализация на класове и обекти, както и използване на техните методи и член променливи.

I. ТЕОРЕТИЧНА ЧАСТ

1. Създаване на класове

За създаването на клас се използва ключова дума **class**. Непосредствено след нея се посочва името на класа. Тялото на класа се огражда във фигурни скоби. Има следния синтаксис:

class *Име_На_Класа* {
}

В дефиницията на класа се съдържат член променливите и методите. Те трябва да започват с ключова дума, която отговаря на тяхната област на видимост. Такива са:

- **public** – разрешава достъп както от вътре, така и от вън класа;
- **protected** – разрешава достъп само от наследниците и оригиналния клас;
- **private** – член променливата или метода е достъпен само от рамките на класа.

Когато за някоя член променлива или метод липсва област на видимост, то тя се подразбира като **public**. Методите на класа се дефинират по същия начин, както и функциите извън класовете – с ключова дума "**function**". В следващия пример е показана дефиниция на член променливи и методи в клас:

```
class Име_На_Класа {
    private $име_променлива1;
    private $име_променлива2;
    function Име_на_метод{
    }
}
```

Функциите (методите) могат да достъпват член променливите на класа по специфичен начин. Те не могат да имат достъп до тях понеже не са дефинирани вътре в самите функции. Използват се чрез псевдопроменливата **$this**. Достъпът на функцията до член променлива от клас, става по следния начин:

$this -> *име_променлива*;

Символът $ се поставен пред „**this**", а не непосредствено пред член променливата.

2. Създаване на обект от клас

За използването на клас е необходимо създаване на негов екземпляр. Той се нарича още обект. Това се извършва с ключова дума **new** по следния начин:

$име_обект = **new** *Име_На_Класа* ;

За един клас могат да се създават множество екземпляри. Извикването на член от класа става чрез знака „->“.

$име_обект->Име_на_метод();

За достъп, до член променливите или методите на класа, то те трябва да бъдат декларирани като **public**. В противен случай ще се генерира съобщение за грешка.

3. Конструктори и деструктори

Конструкторът в PHP е функция, която се извиква автоматично в момента на създаване на обекта. Синтаксисът му е следния начин:

function__constructor (*списък с параметри*) {
}

Пример за извикване на конструктор от клас е следния:

$*име_обект* = **new** *Име_На_Класа* (*списък параметри*);

Деструкторът е функция, която се извиква при унищожаване на обекти или премахване на всички препратки към него.

Пример за функция деструктор е:

function__*destructor* () {
    echo *“ Destroying Class ”*
}

След използване на функцията ще се визуализира моментът на изтриване на обект от класа.

Деструкторите освобождават паметта използвана от обектите.

4. Наследяване

Основно предимство на обектно ориентираното програмиране е способността на класовете да се наследяват. Това спестява време на програмиста, опростява решението на задачите и подобрява качеството, като ограничава повторението на код и възникването на грешки. Наследяващият клас има достъп до член променливите и методите на родителския клас. Това е възможно ако областта на видимост в оригиналния клас е **protected.** Наследяването на клас се извършва чрез ключова дума **extends**:

class *Име_На_Клас* **extends** *Име_На_Оригиналния_Клас*

Достъпът до метод на родителския клас от наследяващия се извършва чрез ключова дума **parent**. Необходимо е преди името на функцията да се използва двойно двоеточие „::“. Методът се достъпва с неговото име:

parent :: *име_Метод()*;

5. Абстрактен клас

Абстрактният клас се използва като градивен в структурата на наследяване. Той не позволява да се създават екземпляри от него. Задаването на клас като абстрактен се извършва чрез ключова дума **abstract**:

```
abstract class Име_На_Клас{
}
```

6. Членове и методи от тип static.

Член променливите и методите на класа могат да се декларират като статични чрез ключова дума **static**. Това позволява тяхното използване без да е необходимо създаването на екземпляри. За разлика от член променливите статични методи е възможно да се използват и чрез обекти от класа. Това е показано в следния пример:

```
class Име_На_Клас{
    public static $име_променлива1= "статична променлива";
    public static function имеМетод(){
            echo "<p>".self::$име_променлива1. „</p>“;
    }
}
echo "<p>".Име_На_Клас::$име_променлива1."</p>"
Име_На_Клас::имеМетод();
$обект=new Име_На_Клас;
$обект->имеМетод();
```

Достъпването на статичен член на променлива от методи на класа се извършва чрез псевдопроменливата **self::**, а не с **&this**.

II. ПРАКТИЧЕСКА ЧАСТ

ЗАДАЧА 1: *Напишете PHP скрипт който създава клас Person. Създайте две член променливи с област на видимост* ***private****. В тях да се съхранява информация за име на човек. Създайте методи на класа извличащи информацията от член променливите, както и методи позволяващи тяхната промяна.*

КОД

```
<?php
//----------------------file Person.php-------------------------
class Person {
        private $strFirstname = "Simon";
        private $strSurname = "Stobart";
        function getFirstname(){
                return $this->strFirstname;
        }
        function getSurname(){
                return $this->strSurname;
        }
```

```
        function setFirstname($strFirstname){
                $this->strFirstname=$strFirstname;
        }
        function setSurname($strSurname){
                $this->strSurname=$strSurname;
        }
}
?>
<?php
//-------------------------file script.php----------------------
function my_autoloader($class_name){
        require_once $class_name. '.php';
}
spl_autoload_register('my_autoloader');

$objSimon=new Person;
echo "<p>".$objSimon->getFirstname()."</p>";
echo "<p>".$objSimon->getSurname()."</p>";
$objSimon->setFirstname("Elizabeth");
$objSimon->setSurname("Hall");
echo "<p>".$objSimon->getFirstname()."</p>";
echo "<p>".$objSimon->getSurname()."</p>";
?>
```

В PHP е прието всеки клас да се записва в отделен файл именуван с името на класа. Поради това кодът е съхранен в два отделни файла „*Person.php*“ и “*script.php*”. В скриптовия файл е необходимо да се достъпят класовете, които ще се използват. Това се извършва чрез операторите **include** или **require**. В конкретния случай това е само един клас, но в практиката често е възможно да се извиква списък от класове. За удобство в PHP е предвидена функцията *my_autoloader*. Тя се извиква автоматично за достъпване на клас, който не е зареден.

Във файла „*Person.php*“ е създаден клас *Person*. Достъпването на член променливите в методите му се извършва чрез псевдопроменливата **$this**. В скриптовия файл се създава обект от класа. Чрез методите му *getFirstname()* и *getSurname()* се извличат стойностите на член променливите. Промяната им е извършена чрез функциите *setFirstname()* и *setSurname()*.

ЗАДАЧА2: *Реализирайте един клас ключалка, описващ състоянието на ключалката (отключено и заключено). Реализирайте и друг клас врата, описващ състоянието на врата (отворена и затворена), както и в какво състояние е ключалката на вратата.*

КОД

```
<?php
//-----------------------file Lock.php-------------------------
```

```
  class Lock{
        private $strLockStatus;
        function __construct($strLockStatus){
              $this->strLockStatus=$strLockStatus;

        }

        function display(){
              echo " and the lock is ". $this->strLockStatus. "</p>";
        }
}
?>
<?php
//----------------------------door.php-------------------------
class Door {
        private $strDoorStatus;
        private $objLock;
        function __construct($strDoorStatus, $strLockStatus){
              $this->strDoorStatus=$strDoorStatus;
              $this->objLock=new Lock ($strLockStatus);
              $this->display();
        }

        function display(){
              echo "<p> The door is ".$this->strDoorStatus;
              $this->objLock->display();
        }
}
?>
<?php
//--------------------------file script.php---------------------
function my_autoloader($class_name){
        require_once $class_name. '.php';
}
spl_autoload_register('my_autoloader');

$objDoor=new Door ("Closed","Locked");
$objDoor=new Door ("Open","Unlocked");
?>
```

Задачата визуализира как се вмъква обект в друг обект като част от дефиницията на клас. Във файла "*script.php*" са създадени два обекта отговарящи за двете врати. Вратите са описани чрез класа *Door* във файл "*door.php*". Той се състои от две член променливи.

Едната съдържа статуса на вратата, а другата *$objLock* представлява обект от тип *lock*. Конструктура за клас *door* определя статуса на вратата, както и създава нов обект ключалка. Той извиква и метода *display()* на клас *door*, който извежда състоянието на вратата. Извиква се и метод *display()* на обекта *lock*. Обърнете внимание как се случва това:

```
$this->objLock->display();
```

Във файл *lock.php* е създаден клас *Lock*, който се създава обект за ключалка. Той описва нейното състояние.

ЗАДАЧА3: *Създайте клас Person съдържащ член променливи за имена на човек. Направете класа така че да не може да му се създава екземпляр. Създайте клас Students съдържащ член променливи за факултетен номер, специалност и наследяващ клас Person. Създайте скрипт, който да отпечатва данните за студент чрез използване на екземпляр от клас Students.*

КОД

```
<?php
//----------------------file Person.php-------------------------
Abstract class Person {
        private $strFirstname;
        private $strSurname;
        function __construct($strFirstname,$strSurname){
                $this->strFirstname=$strFirstname;
                $this->strSurname=$strSurname;
        }
        function display(){
                echo " Името на студента е: ". $this->strFirstname . " " .$this-
        >strSurname. ".";
        }
}
?>
<?php
//----------------------file Student.php-------------------------
class Student extends Person{
        private $strFack;
        private $strSpec;
        function __construct($strFirstname,$strSurname,$strFack,$strSpec)
        {
                parent::__construct($strFirstname,$strSurname);
                $this->strFack=$strFack;
                $this->strSpec=$strSpec;
                parent::display();
```

```
            $this->display();
        }
        function display(){
            echo " Неговия факултетен номер е: ". $this->strFack . ", а неговата
специалност: " .$this->strSpec. ".";
        }
}?>
<?php
//-------------------------file script.php----------------------
function my_autoloader($class_name){
        require_once $class_name. '.php';
}
spl_autoload_register('my_autoloader');

$objStudent=new Student("Иван","Иванов","201911111","КСТ");
//objPerson=new Person  - това е код който следва да генерира грешка
?>
```

За да не може да се създава екземпляр от клас *Person*, то той е дефиниран като абстрактен. Клас *Students* наследява *Person*. Наследяващият клас може да достъпва член променливите и методите на наследения. Същото важи и за конструктора на наследения клас. Достъпването му от наследяващия клас се извършва с оператор "**parent::**". В скриптовия файл е създаден екземпляр на *Students* чрез неговия конструктор.

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Променете кода на задача 2 от практическата част така, че да изпълнява следните условия:
    - Вратата да може да бъде отворена само, ако ключалката е отключена;
    - При отворена врата, ключалката може да се намира в отключено или заключено състояние. Вратата не може да бъде затворена, ако ключалката е заключена.

    Указание: Приемете, че в момента на създаването вратата е затворена, но ключалката отключена.
2. Създайте абстрактен клас *Vehicle* (превозно средство) и класове които го наследяват: *Car*(кола), *Boat*(лодка), *Bicycle*(велосипед). Създайте клас *HybridCar* (хибридна кола), който да наследява *Car*. Добавете методи *display()*, които да показват данните на съответното превозно средство. За създаването на класовете използвайте следния списък с член променливи.
    - IntWeight (тегло);
    - IntLength(дължина);
    - IntTopSpeed(максимална скорост);
    - intDisplacement(водоизместимост);

- intRemainingFuelLitres(оставащо гориво в литри);
- intRemainingBatterylife(оставащо електричество в акумулатора);
- strDescription(описание).

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Какво представляват областите на видимост които се задават за член променливите и методите на класа?
2. Как функциите достъпват член променливите на класа?
3. Как се създава екземпляр (обект) от клас?
4. Какво е предназначението на деструкторите?
5. С коя ключова дума се извършва наследяване на клас?
6. Какво е предназначението на ключова дума **parent**?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 5. Взаимодействие на PHP скрипт с формуляри

ЦЕЛ

Запознаване на студентите с процеса на създаване на форми и взаимодействие на PHP с тях. Придобиване на практически умения за обработка на данни от различни формуляри чрез PHP скрипт.

I. ТЕОРЕТИЧНА ЧАСТ

В упражненията по дисциплината за създаването на формуляри ще се използват HTML документи. В тях могат да се вграждат PHP скриптове. Това позволява гъвкавото взаимодействие между потребителя и сървъра.

За създаването на формуляри се използва елемента ***form***, който се състои от отварящ и затварящ таг и обхваща всички останали елементи между тях:

```
<form action="име_на_файл" method="тип">
……
</form>
```

Началният таг на елемента ***form*** съдържа два задължителни атрибута ***action*** и ***method***. Първият определя действието, което трябва да се извърши при изпращане на скрипт, а вторият – метода, според който данните да бъдат изпратени към скрипта. Най-често използваните методи за заявки са **GET** и **POST**. В настоящото ръководство ще се използва метод **POST** като по надежден от гледна точка на сигурността.

Между началния и крайния таг на ***form*** се поставят редица други елементи, които изграждат структурата на формуляра. Някои от най-често използваните са:

- етикети;
- текстови полета;
- радиобутони;
- полета за отметка;
- списъци;
- многоредови текстови полета;
- полета от тип изображение ;

Таг ***label*** и се използва най-често за поставянето на етикети за текстовите полета. Препращането на етикета към съответното текстово поле става чрез атрибут ***for*** на таг ***label***. Синтаксисът му е следния:

```
<label for="идентификатор_на_полето">Етикет</label>
```

За създаването на по-голямата част от елементите в един формуляр, се използва таг ***input*** с задължителен атрибут ***type***. Към тага могат да се добавят и други препоръчителни атрибути като ***name*** и ***id***. Синтаксисът на таг ***input*** е следния:

*<input type=”**тип**” name=”**име_на_полето**” id=“**идентификатор_на_полето**“/>*

Атрибут ***id*** представлява идентификатор, който свързва елемента с етикета ***label***. Атрибут ***name*** служи за уникално име на въведения елемент. То се използва при подаването на информация към скрипта. Чрез атрибут ***type*** се посочва типа на полето. Някои от най-често използваните типове са:

- *text* – за създаване на текстово поле;
- *password* – за създаване на текстово поле, чиято информация се скрива с точки при въвеждането ѝ;
- *radio* – за създаването на радиобутони;
- *checkbox* – за създаването на полета за отметка;
- *image* – за създаването на полета от тип изображение;
- *hidden* – за създаването на скрито текстово поле;
- *submit* – за създаването на бутон за изпращане.

Типовете *radio* и *checkbox* на атрибут ***type*** създават специално поле във формуляра, с което потребителят може да взаимодейства. Те не позволяват въвеждането на текст, а предлагат няколко възможни варианта за избор. За да се предостави информация на потребителя за отделните възможности се използва атрибут ***value*** в тага ***input***. Разликата между радиобутон и *checkbox* е в броя на избраните възможности от дадена група в определен момент. При радиобутон може да бъде избрана само една възможност от определена група, а при *checkbox* – повече от една. За групираните полета се посочват еднакви имена за атрибут ***name***.

Типът *image* на атрибут ***type*** позволява на формулярите да обработват изображения като входни данни. За целта към тага ***input*** трябва да се добави още един атрибут ***src***, в който се посочва файлът с изображението. Синтаксисът на таг ***input*** за обработка на изображение е следния:

*<input type=’**image**’ src=’**файл_с_изображение**’ name=‘**име**‘/>*

Типът *hidden* на атрибут ***type*** предоставя на уеб разработчиците да включват данни към уебсървъра, които не могат да се видят или модифицират от потребителите. Скритото поле е възможно да съхранява информация за записа, които трябва да бъде актуализиран в базата данни, когато формулярът се изпрати. Такава стойност се подава чрез атрибут ***value*** включен в ***input*** тага. Въпреки, че стойността не се показва на потребителя в съдържанието на страницата, то тя е видима и може да се редактира.

Тип *submit*, позволява данните от формуляра да бъдат изпратени към сървъра. Този елемент се изобразява на екрана във вид на бутон. При задействане на бутона информацията се изпраща към скриптовия файл, посочен в атрибут ***action*** на тага ***form***.

Изпратените данни към PHP скрипта са достъпни за обработка чрез използването на вградени PHP функции. Тези функции представляват асоциативен масив, който съхранява изпратените от формуляра данни. Тъй като информацията се предава с POST заявки, то данните ще се достъпват като елементи от масива $_POST. Ключовете на данните от масива съответстват на имената на полета от формуляра. Така например, въведената информация за текстово поле с атрибут ***name****=”име”* се съдържа в:

$_POST['име']

PHP скриптът обработващ данните подадени от формуляра, може да бъде отделен от html документа. Тогава във формуляра в таг ***form*** за атрибут ***action*** се задава името на скриптовия файл. PHP скрипта може да е вграден в самия html документ. Това означава, скрипта и формуляра да бъдат създадени в един общ файл. В такъв случай, в атрибут ***action*** на тага ***form*** се посочва неговото име. В PHP съществува сървърна променлива, която съдържа името на текущия скрипт. Нейното използване предотвратява проблеми възникнали от преименуването на файлове. Тази сървърна променлива е следната:

$_SERVER["PHP_SELF"]

Вграждането на горната променлива в тага ***form*** може да се извърши чрез командата **echo** по следния начин:

```
<form action='<?php echo $_SERVER["PHP_SELF"]?>' method="mun">
```

Това гарантира, че скриптът ще извиква винаги себе си, независимо от името на файла в който се съхранява.

За да се предотврати изпълнението на PHP скрипта, преди подаването на данните от формуляра е необходимо да се направи проверка дали те вече са изпратени. За целта може да се използва функция ***isset()***, по следния начин:

```
if( isset($_POST["име_на_бутона"])){
……..}
```

Функцията връща ***true***, ако елемента от масива е установен. Това означава, че бутонът във формуляра за изпращане на информацията е използван. Необходимо е целият PHP скрипт да бъде поставен в тялото на конструкция ***if***. Желателно е в него да бъдат направени допълнителни проверки за коректно попълнени полетата на формуляра.Това ще гарантира, че няма как да се изпълни PHP скрипта без да се подаде информацията към уебсървъра. Ако проверките не бъдат направени, това може да доведе до неочаквани резултати или предупреждения, че някои променливи не са инициализирани.

За създаването на елементи от тип списък се използват двата тага ***select*** и ***option***, като вторият е вграден в първия. Създават се толкова на брой тагове ***option***, колкото искаме да е голям списъкът. Структурата на конструкцията е следната:

```
<select name='име'>
    <option value='стойност'>съдържание</option>
    ……
</select>
```

Атрибутът ***name*** в тага ***select*** е уникално име за този елемент, а атрибутът ***value*** на тага ***option*** съдържа избраната стойност за съответния ред от списъка. Тя ще се изпрати към сървъра, при евентуално нейно избиране от потребителя. ***Съдържанието*** е стойността, която се визуализира в браузъра на потребителя за съответния ред от списъка.

За създаването на многоредови текстови полета се използва тага ***textarea***. Той има следната структура:

```
<textarea name='име' rows='брой_редове' cols='брой_колони'>
</textarea>
```

Атрибутите в тага не са задължителни. Първият атрибут ***name*** представлява уникално име за полето, а атрибутите ***rows*** и ***cols*** съдържат броя на редовете и колоните.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част е показан пример, чрез които се демонстрира взаимодействието на формуляри с PHP скриптове.

ЗАДАЧА1: *Създайте форма предоставяща възможност за регистрация на потребител чрез въвеждането на име, потребителско име и парола. Чрез падащ списък потребителя да избере своята рождена дата. Във формуляра да се предвиди възможност и за избор на пол. Създайте PHP скрипт вграден в html формуляра, който извежда поздравително съобщение за новия потребител, както и неговото потребителско име и парола.*

КОД

```
<html>
    <body>
        <h2> Please enter your personal details:</h2>
        <form action='<?php echo $_SERVER["PHP_SELF"]?>' method='post'>
            <p>
                <label for="strFirstName">Firstname</label>
                <input type="text" name="strFirstName" id="strFirstName"/>
            </p>
            <p>
                <label for="strSurName">Surname</label>
                <input type="text" name="strSurName" id="strSurName"/>
            </p>
            <p>
                <label for="strUserName">Username</label>
                <input type="text" name="strUserName" id="strUserName"/>
            </p>
            <p>
                <label for="strPassword">Password</label>
                <input type="password" name="strPassword" id="strPassword"/>
            </p>
            <p>
                <label for="date">My Birthday is</label>
                <select id="date">
                    <?php       for($i=1;$i<32;$i++){       ?>
                        <option value="<?php echo $i;?>"><?php echo $i;?></option>
                    <?php       }       ?>
                </select>
```

```
            <select id="date">
              <?php       for($i=1;$i<13;$i++){       ?>
                  <option value="<?php echo $i;?>"><?php echo $i;?></option>
              <?php       }       ?>
            </select>
            <select id="date">
              <?php       for($i=2000;$i<2100;$i++){       ?>
                  <option value="<?php echo $i;?>"><?php echo $i;?></option>
              <?php       }       ?>
            </select>
          </p>
          <p>
            <label for="gender">My gender is</label>
            <input type="radio" name="gender" id="gender" value="male"/>
            <input type="radio" name="gender" id="gender" value="female"/>
          </p>
          <p>
            <input type="submit" name="Submit"/>
          </p>
        </form>
    </body>
</html>
<?php
if(isset($_POST['Submit'])){
        $strFirstName=$_POST['strFirstName'];
        $strSurName=$_POST['strSurName'];
        $strUserName=$_POST['strUserName'];
        $strPassword=$_POST['strPassword'];
        echo "<p>Greatings $strFirstName $strSurName</p>";
        echo"<p>Your username is $strUserName and your password is $strPassword</p>";
}
?>
```

За създадения формуляр в отделни параграфи са отделени двойки елементи с тагове ***input*** и ***label***. Те създават съответните текстови полета и етикети към тях. В текстовите полета потребителят може да напише своите данни. Чрез таг ***select*** и вложеният в него таг ***option*** се създава падащ списък, чрез който потребителят да избере своята рождена дата, месец и година. За намаляване броя на редовете, създаващи отделните тагове ***option***, се използвал оператор за цикъл **for**, чрез вграждане PHP скрипт. За избора на пол се използват радиобутони. Те са създадени чрез таг ***input*** с атрибут ***type***=*radio*. Важно е двата радиобутона да имат еднакви имена за атрибут ***name***.

При натискане на бутона за изпращане, данните за обработка от формуляра се подават към PHP скрипта, вграден в същия файл. Това се извършва чрез сървърната

променлива *$_SERVER["PHP_SELF"]* вградена за атрибут ***action*** на тага ***form***. При изпълнението на скрипта първо се проверява дали бутонът с атрибут ***name****='Submit'* е задействан, чрез функцията ***isset()***. Ако бутонът е използван, то ще се изпълни PHP скрипта, в противен случай – ще бъде зареден само формулярът. Скриптът присвоява стойностите на елементите от масива **$_POST** в отделни променливи и извежда необходимите съобщения чрез команда **echo**.

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Разгледайте и изпълнете задачата от практическата част. Редактирайте кода, така че да отпечатва датата на раждане и полът на съответния потребител. Добавете текстово поле във формуляра, позволяващо на потребителя да въвежда повече от един ред допълнителна информация. Нека допълнителна информация също да се извежда.
2. Променете втора задача от самостоятелна работа в *тема 4*, така че скрипта да използва формуляр от който потребител да въвежда данните за правоъгълника.
3. Създайте формуляр представляващ форма за вход за потребителско име и парола. Променете PHP скрипта от третата задача за самостоятелна работа в *тема 4*, така че да проверява дали данните от формуляра съвпадат с някои от елементите на масива.
4. Със софтуер за рисуване или обработка създайте изображение, съдържащо следните фигури: кръг, квадрат, триъгълник, петоъгълник. Направете форма съдържаща изображението. Създайте PHP скрипт, проверяващ каква фигура е избрана и извеждащ текст посочващ нейното наименование.
   Упътване: *За да разберете координатите на курсора на мишката за избраната фигура е необходимо да знаете размера на самото изображение в пиксели. Можете да го видите от свойствата на файла с изображения. Позиция (0,0) се намира в горния ляв ъгъл на изображението и спрямо нея се определя къде се намира курсора.*

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Кои са задължителните атрибути на таг **form**?
2. Какво е предназначението на атрибут **action** от таг **form**?
3. Какво е предназначението на атрибут **for** на таг **label**?
4. Как се създава текстово поле предназначено за парола чрез таг **input**?
5. Какво трябва да се направи, за да може при натискане на бутон от формата да се задейства PHP скрипт, намиращ се във файла на документа съставящ самата форма?
6. С кой таг се създават многоредови текстови полета?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 6. Изграждане на PHP връзка с MySQL база данни

ЦЕЛ

Запознаване на студентите с начините за достъпване на информация между PHP скрипт и MySQL бази данни. Придобиване на практически знания и умения за изпращане на SQL заявки чрез скрипт към MySQL сървър.

I. ТЕОРЕТИЧНА ЧАСТ

Връзка на PHP с MySQL сървър може да се извършва чрез някои от следните два подхода:

- **PDO** (PHP Data Objects)
- **MySQLi** разширения.

Основната разлика между тях е че **PDO** може да работи с различни софтуери за управление на бази данни, докато **MySQLi** работи единствено с MySQL бази данни. Ако е необходимо да се превключи проекта да работи към друга база данни, то **PDO** прави процеса по-лесен, понеже при **MySQLi** е необходимо да се пренапише целия код. И двата подхода са обектно ориентирани, но **MySQLi** предлага и процедурен подход. В настоящото ръководство е използван **MySQLi** процедурен подход.

Връзката между PHP скрипта и MySQL сървъра се осъществява с функцията **mysqli_connect()** по следния начин:

*ресурс=**mysqli_connect**(сървър, потребителскоИме, парола);*

Тя приема три параметъра. Първият е името на сървъра, вторият е потребителското име с което се осъществява връзката, а третият е неговата парола. Функцията **mysqli_connect()** връща стойност от тип ресурс. При осъществяване на връзка от локалната машина, където е инсталиран MySQL сървъра, то първият параметър трябва да има стойност „*localhost*“ или „*127.0.0.1*“. Параметрите на функцията трябва да бъдат оградени с апострофи или двойни кавички. Ако липсва стойност за даден параметър, то той се приема за празен низ, както е в израза:

*$con=**mysqli_connect**(‘localhost’,’root’,’ ’);*

За осъществяване на връзката е използвано потребителско име root без парола.

ЗАБЕЛЕЖКА: В реална среда липсата на парола за потребителско име към MySQL сървър представлява заплаха от гледна точка на сигурността. Желателно е да се използват достатъчно силни пароли.

Функцията **mysqli_connect()** е възможно да има и четвърти параметър, посочващ база данни по подразбиране. Ако не е посочен, може да се зададе по-късно в скрипта чрез функцията **mysqli_select_db()**. Тя приема два параметъра, ресурсът върнат от функцията **mysqli_connect()** и името на базата данни. Синтаксисът е следният:

mysqli_select_db*(ресурс, ИмеНаБазатаОтДанни);*

Горната функция връща стойност ***true***, ако изборът на базата е бил успешен и ***false***, ако не е. Ако заявената връзка с база от данни не може да бъде осъществена, е необходимо скриптът да бъде прекратен. Това се извършва чрез функция **die()**:

die *(“Съобщение за грешка”);*

Тази функция има само един параметър – съобщението което се показва, преди изпълнението на скрипта да бъде прекратено. Стандартна практика е да се използва комбинация от функцията **die()** с функцията **mysqli_error()**. Втората връща съобщението за грешка от настъпило събитие. Синтаксисът ѝ е следният:

mysqli_error*(ресурс);*

Функцията най-често приема за параметър, ресурсът върнат от **mysqli_connect()**. Друг вариант за извеждането на съобщението за грешка е чрез функцията **mysqli_connect_error()**. Комбинацията между тази функция и функция **die()** е следната:

die *(“Неуспешно свързване:”****.mysqli_connect_error****());*

За изпращане на SQL заявки към MySQL сървър се използва функция **mysqli_query()**. Тя приема два параметъра, ресурсът указващ базата от данни, който е получен от функцията **mysqli_connect()** и низ със SQL заявка. При изпълнение на функцията се получава ресурс, който води към обработените данни.

*връзкаКъмДанни=****mysqli_query****(ресурс, заявка);*

В упражнението са разгледани следните оператори ***CREATE DATABASE***, ***CREATE TABLE*** и ***INSERT INTO.*** Те се използват за SQL заявки към MySQL сървър.

Операторът ***CREATE DATABASE*** се използва за създаване на нова база данни. След него се посочва името ѝ:

CREATE DATABASE *ИмеНаБазаОтДанни;*

Операторът ***CREATE TABLE*** се използва за създаване на нова таблица в съществуваща база данни. След него се посочва името на таблицата, последвано от описанието на полетата на таблицата, разделени със запетая. Цялото описание се огражда в скоби:

CREATE TABLE *ИмеНаТаблица*
(поле1 ТИП, поле2 ТИП, поле3 ТИП, ключов индекс (поле))

Операторът ***INSERT INTO*** се използва за вмъкване на записи във вече съществуваща таблица. Той има следния вид:

INSERT INTO *таблица (поле1, поле2,…)* ***VALUES****(‘стойност1’,‘стойност2‘,…)*

След ключовите думи ***INSERT INTO*** се посочва името на таблицата, в която ще се добавя записа. Полетата в първите скоби съответстват на структурата на таблицата. Стойностите след ключова дума ***VALUES*** съответстват на стойностите за съответните полета посочени в първите скоби.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част е решена задача която демонстрира осъществяване на достъп от PHP скрипт към MySQL база от данни.

ЗАДАЧА: *Осъществете достъп от PHP скрипт към MySQL сървъра инсталиран с WAMP пакета на локалната машина. Създайте БД "University". Към нея добавете таблица "Students" показана на табл.1. Създайте HTML форма с 4 полета. В нея да се съдържа информация за студентите, която да се добави в таблица Students.*

Табл.6.1. Students

Fnom	*name*	*age*	*grade*
2344	*Dian*	*21*	*3.56*
2234567	*Ivan*	*21*	*4.59*

Решението на поставената задача е реализирано чрез следната последователност от действия:

1. *Осъществяване на достъп до MYSQL сървъра*

КОД

```
<?php
 $con = mysqli_connect('localhost', 'root', '', 'university') or die(mysqli_connect_error());
?>
```

За да се осъществи достъп до MySQL сървъра се използва функцията **mysqli_connect()**. В примера се осъществява достъп с потребител *root* без парола към MySQL сървър инсталиран на локалната машина. Задава се база данни *university* по подразбиране. Ако достъпът не се осъществи, PHP скрипта се прекратява и се връща причина за възникналата грешка.

ЗАБЕЛЕЖКА В реална среда липсата на парола за потребител *root* представлява заплаха за сигурността. Трябва да се използва достатъчно силна парола.

2. *Създаване на базата от данни „University"*

КОД

```
<?php
//1. Свързване със сървъра за MYSQL БД и проверка на връзката
$con=mysqli_connect('localhost','root','');
if (!$con) die(mysqli_connect_error());

//2. Създаване на заявка към сървъра за създаване на база от данни с име university
$str='CREATE DATABASE University';

/*3. Изпращане на заявката към сървъра за създаване на база от данни с име
University  */
mysqli_query($con,$str) or die('Error creating database: '. mysqli_error($con));
?>
```

В първата част от кода се осъществява достъп до същата база данни както в т.1. Във втората част от кода се декларира променлива **$str** от тип низ. Стойността ѝ представлява SQL заявка създаваща база данни „*University*“. В третата част от кода, тя се изпраща към MySQL сървъра. Ако заявката не се изпълни, функция **die()** прекратява изпълнението на скрипта и връща съответното съобщение за грешка. Анализирайте резултата от изпълнението на задачата.

3. *Проверка за съществуването на вече създадената база данни „university“.*

КОД

```
<?php
$con=mysqli_connect('localhost', 'root', ' ') or die(mysqli_connect_error());
 echo "Connection to the server was successful! <br/>";
 mysqli_select_db($con, "university") or die(mysqli_error($con));
 echo "Database was selected! <br/>";
 ?>
```

Този код осъществява достъп до база данни „*University*“ намиращ се на локалната машина. Ако поради някаква причина достъпът е неуспешен се извежда причината за това.

4. *Създаване на таблица „Students“ в базата от данни „University“.*

КОД

```
<?php
   // 1 част. Осъществяване достъп до базата от данни “University”
 $con =mysqli_connect('localhost', 'root', '','University') or die(mysqli_connect_error());
  echo 'Connected successfully! ';
   //2 част. Създаване на таблица Студенти
  $sql ="CREATE TABLE Students( fnom INT(10) NOT NULL,   name VARCHAR(32),
age INT(6),   grade DOUBLE (3,2),   PRIMARY KEY  (fnom) ) DEFAULT CHARSET=utf8";
 mysqli_query( $con, $sql ) or die(mysqli_error($con));
 echo "Table students created successfully";
   //3 част. Затваряне на връзката към базата от данни
 mysqli_close($con);
?>
```

Първата част от кода осъществява достъп до MySQL сървъра и задава базата данни *“University”* по подразбиране. С втората част от кода се създава SQL заявка, която се изпраща към MySQL сървъра. Ако се извърши успешно се връща съобщението „*Table students created successfully*“, в противен случаи скриптът се прекратява и се извежда причината за грешка.

Чрез функцията **mysqli_close()** в третата част от кода се затваря отворената връзка от **mysqli_connect()**.

5. *Вмъкване на един запис в таблицата Студенти.*

Кодът в тази точка е същия от точка 4 с различна SQL заявка:

КОД

```
$sql="INSERT INTO Students (fnom,name,age,grade) VALUES (2234567, 'Иван', 21, 4.59)";
mysqli_query( $con, $sql ) or die(mysqli_error($con));
```

6. *Добавяне на данни чрез използване на HTML форма:*

КОД

forma.php

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <form action="test.php" method="post">
            Въведи факултетен номер: <input type="text" name="fnom" />
            Име: <input type="text" name="name" />
            Възраст: <input type="text" name="age" />
            Успех: <input type="text" name="uspeh" />
            <input type="submit" />
        </form>
    </body>
</html>
```

test.php

```
<?php
    $con = mysqli_connect('localhost', 'root', ' ', 'university') or die(mysql_error());
    $sql="INSERT INTO Students (fnom, name, age, grade) VALUES ("$_POST ['fnom']", "$_POST ['name']", "$_POST ['age']", "$_POST ['uspeh'])";
     mysql_query($sql, $con) or die(mysql_error();
    echo "1 record added"; mysql_close($con);
?>
```

Създават се два файла *forma.php* и *test.php*. В първия файл се намира HTML документа, който създава формата, показана на *фиг.6.1*. Във втория файл се съдържа създаденият PHP скрипт за запис на информация в таблицата. Достъпът до въведената информация във формата се извършва чрез асоциирания масив **$_POST**. За ключ се използва името на съответното поле от формуляра.

Фиг.6.1 Визуализация на формата

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Създайте PHP скрипт за връзка към MySQL базата от данни.
2. Създайте база данни с наименование "*Firma*".
3. В базата данни създайте таблица „*Products*" със следните полета „ИмеНаПродукта", „Описание", „Цена". Задайте типа на полетата според предназначението им.
4. Създайте HTML форма с три полета и бутон. Напишете PHP скрипт, който да въвежда информация от формата като нов запис в таблица „Products".
5. Затворете създадената връзка към MySQL базата от данни.

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Какви са основните подходи за реализация на връзка между PHP скрипт и MySQL сървър?
2. От колко параметъра се състои функцията **mysqli_connect()**?
3. Какво е предназначението на функцията **mysqli_select_db()**?
4. Какво е предназначението на първия параметър на функцията **mysqli_query()**?
5. С коя функция се прекратява изпълнението на PHP скрипта?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 7. Извличане на информация от MySQL база данни чрез PHP скрипт

ЦЕЛ

Запознаване на студентите с методи за извличане на информация от MySQL сървър чрез PHP скрипт. Придобиване на практически знания и умения за реализиране на елементарни заявки чрез скрипт.

I. ТЕОРЕТИЧНА ЧАСТ

Извличането на информация от MySQL бази данни се извършва чрез SQL заявки. Те се изпращат към сървъра чрез функцията **mysqli_query()**, която беше разгледана в *тема 6.* За конструкцията на заявката се използва оператора SELECT. Синтаксисът му е следния:

SELECT [DISTINCT] *[поле1, …. , полеN]*
FROM *Име_на_таблица1 [, … , Име_на_таблицаN]*
*[**WHERE** условие]*
*[**GROUP BY** списък с полета]*
*[**ORDER BY** списък с полета]*
*[**LIMIT** количество]*

ЗАБЕЛЕЖКА. По правило квадратни скоби не би следвало да има. В конкретния случай те се използват за обозначение на фрагментите от заявката които не са задължителни.

След ключова дума SELECT се посочват наименованията на отделните полета от таблица за извличане на информация. Списъкът с полета би могъл да се замени със звезда, ако е необходима информация за всяко от тях. При необходимост за всяко поле може да се зададе псевдоним чрез ключова дума AS. След ключова дума FROM е задължително да се посочи името на таблицата, в която се намира дадената информация и евентуално името на базата данни.

Следващите фрагменти от заявката не за задължителни. Те служат за ограничаване и подреждане на извлечените записи. Един от основните методи за ограничаване е чрез задаване на условие след ключова дума WHERE. Извлечените записи трябва да отговарят на зададеното условие. С LIMIT се задава ограничение на броя записи. С ключова дума GROUP BY резултатът се групира спрямо списъка с полета, а чрез ORDER BY се подрежда спрямо тях. Подредбата може да се извърши по азбучен ред чрез ключова дума ASC, и обратно на азбучния ред – с DESC.

Резултатът от изпълнението на функцията **mysqli_query()** е ресурсен указател. Той сочи към записите, които отговарят на SQL заявката подадена като втори параметър към функцията. За извеждането на извлечените записи се използва функцията **mysqli_fetch_array()**. Синтаксисът ѝ е следният:

```
Масив = mysqli_fetch_array ( връзкаКъмДанни );
```

Функцията приема само един параметър – ресурсът, върнат от функцията **mysqli_query()**. Като резултат от нейното изпълнение се получава масив, чиито ключове са полетата на записите подадени в оператора SELECT. За да се изведат всички записи е необходимо функцията **mysqli_fetch_array()** да се използва в оператор за цикъл, както е показано в следващия пример:

```
While (Масив = mysqli_fetch_array ( връзкаКъмДанни ) )
  {  echo Масив[“поле1”].“ “;
  …….
  echo Масив[“полеN”].“ “;}
```

Функцията **mysqli_fetch_array()**се извиква като условие в оператора за цикъл **While**. Всеки запис извлечен от функцията се подава към масив, чиито елементи са съдържанието на отделните полета. Елементите на масива се достъпват и извеждат в тялото на цикъла, чрез ключове отговарящи на наименованието на отделните полета. Операторът за цикъл **While** се изпълнява, докато функцията връща записи получени от съответната **SELECT** заявка. След като бъдат изведени всички записи функцията **mysqli_fetch_array()** връща като резултат **false** и операторът **While** прекратява своето действие.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част е показан пример за достъп до определена MySQL база данни и за извличане на информация от нея. Формулирана е задача с няколко подусловия.

ЗАДАЧА: *Осъществете достъп от PHP скрипт към MySQL база от данни „University“, която бе създадена в практическата част на тема 6.*

1) *Изведете всички записи от таблица Students.*
2) *От таблица Students изведете всички студенти които имат успех по-голям от добър 4.00*
3) *Създайте html форма с падащ списък и бутон, която предоставя възможност на потребителя да избере факултетния номер на студент. Използването на бутона да позволява извеждането на името на студента и средния успех от следването му.*

Табл.7.1. Students

fnom	*name*	*age*	*grade*
2344	*Dian*	*21*	*3.56*
2234567	*Ivan*	*21*	*4.59*

1. *Извеждане на всички записи от таблица Students*

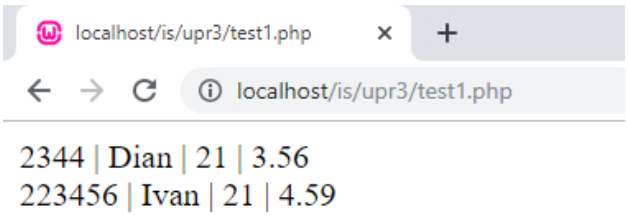
КОД

```
<?php
//1. Осъществяване връзка към база данни University
$con = mysqli_connect('localhost', 'root', '', 'university') or die(mysqli_connect_error());

//2. Задаване на необходимата SQL заявка.
$querySt="SELECT * FROM Students";

//3.Изпращане на необходимата SQL заявка към сървъра
$dbStRecords=mysqli_query($con,$querySt) or
die('Неуспешно запитване. Получи следната грешка'.mysqli_error($con));

//4. Прочитане на всички върнати записи от заявката чрез използване на цикъл
while($StRecords=mysqli_fetch_array($dbStRecords))
{   echo $StRecords["fnom"]." | ";
    echo $StRecords['name']." | ";
    echo $StRecords['age']." | ";
    echo $StRecords['grade']."<br>";
 }

//5. Затваряне на връзката с базата данни
mysqli_close($con);
?>
```

В първата част от кода се осъществява достъп до базата данни *"University"* от MySQL сървър. При неуспешно осъществяване на връзката скриптът се прекратява. Извежда се съобщението „*Неуспешна връзка*“ и точната причина за грешката.

Във втората част се подготвя SQL заявката за изпращане към сървъра. Тя се записва в променливата $querySt. В третата част тази променлива се подава като втори параметър на функцията **mysqli_query()**. Ресурсът върнат като резултат от функцията се присвоява на променливата $dbStRecords. При неуспешно изпращане на заявката, скриптът се прекратява и се извежда съобщение за конкретната грешка.

В четвъртата част се използва функцията **mysqli_fetch_array()**, която извлича един запис от ресурса $dbStRecords, върнат от функцията **mysqli_query().** Този запис се записва в масива $StRecords. За извличането на всички записи е използван оператор за цикъл **While**. В тялото на оператора се извежда информацията на отделните записи. В петата част на кода се прекратява връзката към MySQL сървъра. На *фиг. 7.1* е показан резултатът от изпълнението на горния скрипт.



Фиг.7.1 Резултат от изпълнението на скрипта.

2. *Извеждане на всички студенти които имат успех по-голям от добър 4.0*

За изпълнението на условието се използва PHP скрипта от *подусловие 1)*. Разликата е единствено във втората част на кода. Необходимо е SQL заявката да се промени по следния начин:

*$querySt="SELECT * FROM Students Where grade > 4.00";*

Извеждат се имената само на тези студенти, които имат успех по-голям от добър 4.00.

3. *Създаване на формата и реализация на скрипта който я обработва*

КОД

forma.php

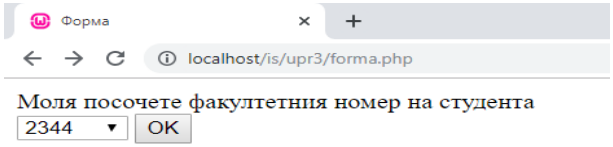
```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
    <?php
        $con = mysqli_connect('localhost', 'root', '', 'university')
        or die(mysqli_connect_error ());
        $querySt="SELECT DISTINCT fnom FROM Students";
        $dbStRecords=mysqli_query($con,$querySt) or
        die('Неуспешно запитване. Получи следната грешка'.mysqli_error($con));
    ?>
        <form method="post" action="test1.php">
     Моля посочете факултетния номер на студента<br>
         <select name="nomer">
        <?php
            While ($StRecords=mysqli_fetch_array($dbStRecords))
            {
         ?>
            <option value="<?php echo $StRecords['fnom']; ?>"><?php echo
$StRecords['fnom'];?></option>
```

```
        <?php
            }
        ?>
        </select>
            <input type="submit" name="submit" value="OK">
        </form>
</body>
</html>
```

test1.php

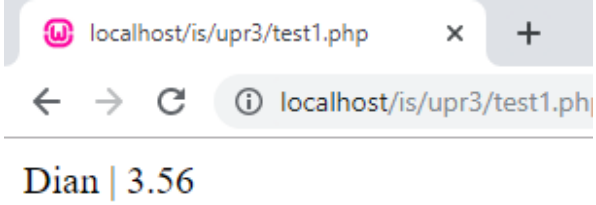
```
<?php
    $nom=$_POST['nomer'];
    $querySt="SELECT name, grade FROM students Where fnom=$nom";
    $dbStRecords=mysqli_query($con,$querySt) or
    die ('Неуспешно запитване. Получи се следната грешка'.mysqli_error($con));
    $StRecords=mysqli_fetch_array($dbStRecords))
    echo $StRecords['name']." | ";
    echo $StRecords['grade']."<br>";
?>
```

Създадени са два файла *forma.php* и *test1.php*. В първия файл се намира HTML документа, който създава формата, показана на *фиг.7.2*. За разработването на падащия списък е необходимо използването на толкова на брой тагове ***option***, колкото е и броят на студентите в таблицата *Students*. Реализацията на тага е извършена в тялото на оператор за цикъл **for** чрез вграден PHP скрипт в HTML документа.



Фиг.7.2 Визуализация на формата съдържаща падащ списък

Във втория файл е PHP скрипта, който изпраща необходимата SQL заявка и извежда резултата върнат от нея. Функцията **mysqli_fetch_array()** е използвана без оператор за цикъл понеже броят на върнатите записи от SQL заявката е само един. Резултатът от изпълнението на скрипта е показан на *фиг.7.3*.



Фиг.7.3 Визуализация на резултата от изпълнението на PHP скрипта

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

Като използвате базата данни „*Firma*“ създадена в *тема 6* решете следните задачи:

1. Създайте PHP скрипт, който да извлича съдържанието на таблица “*Products*”.
2. Създайте PHP скрипт, който да извежда само тези продукти, които имат цена по голяма от предварително зададена.
3. Създайте потребителска форма с падащ списък, който да предоставя възможност на потребителя да избере определен продукт. След използването на бутон от формата да се извежда името, описанието и цената на продукта.

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Какъв резултат връща функцията **mysqli_fetch_array()**?
2. Как се достъпва информацията върната от функцията **mysqli_fetch_array()**?
3. Какво е предназначението на ключова дума **DISTINCT** в заявка **SELECT**?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 8. Информационно търсене. Генериране на справки чрез PHP скрипт от MySQL база данни

ЦЕЛ

Запознаване на студентите с методи за реализиране на елементарни справки за търсене на информация от бази данни. Придобиване на практически знания и умения за използването на PHP методите за извеждане на търсената информация.

I. ТЕОРЕТИЧНА ЧАСТ

Извличането на данни от MySQL сървър се извършва чрез изпращане на определени SQL заявки. За целта се използва функцията **mysqli_query()**. Тя връща ресурсен указател към всички записи съответстващи на подадената SQL заявка. За извеждането на извлечените записи съществуват различни функции. Една от тях **mysqli_fetch_array()** беше представена в *тема 7*. Друга такава функция е **mysqli_fetch_row()**. Нейният синтаксис е следният.

```
Масив = mysqli_fetch_row ( връзкаКъмДанни );
```

Функцията приема само един параметър – ресурсът, получен от функцията **mysqli_query()**. Тя връща масив, чиито елементи съответстват на информацията в полетата на записите получени от подадената заявка. Ключовете на масива са целочислени индекси 0,1,2 и т.н. Индекс 0 съответства на първото поле в структурата на подадената заявка, индекс 1 – на второто поле и т.н. За извличане на всички записи от дадена таблица се използва чрез оператор за цикъл, подобно на функцията **mysqli_fetch_array().**

```
While (Масив = mysqli_fetch_row ( връзкаКъмДанни ) )
   {  echo Масив[0].“ “;
   …….
   echo Масив[N].“ “;}
```

Функцията **mysqli_data_seek()** служи за позициониране на точно определен запис от извлечените. Тя приема два параметъра. Първият е ресурсния указател върнат от функцията **mysqli_query()**, а вторият – целочислен индекс, който показва номера на съответния запис за позициониране. Функцията връща булев резултат в зависимост от това дали е извършено успешно позиционирането или не. Нейният синтаксис е следния:

```
$pos=mysqli_data_seek(връзкаКъмДанни,0) or die('Възникна грешка’);
```

Индексът 0, като втори параметър във функцията, означава да се локализира първия запис от ресурса с извлечени записи. За втория запис е необходимо да се посочи индекс 1, а за **n**-*тия* – n-1. Използването на **mysqli_data_seek()** преди извеждане на извлечената

информация чрез функциите **mysqli_fetch_row()** и **mysqli_fetch_array()** променя техния резултат. Той ще съдържа всички записи след позиционирания.

Функцията **mysqli_num_rows()** връща броя на извлечените записи. За нея се подава един параметър – ресурсът върнат от **mysqli_query()**. Синтаксисът ѝ е следния:

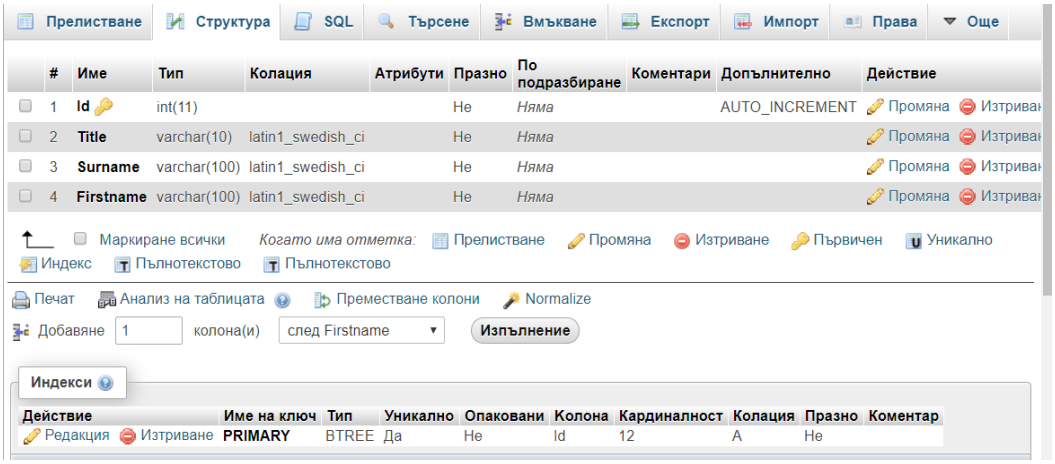
$number=**mysqli_num_rows(**връзкаКъмДанни**)**

Променливата $number съответства на броя на извлечените записи.

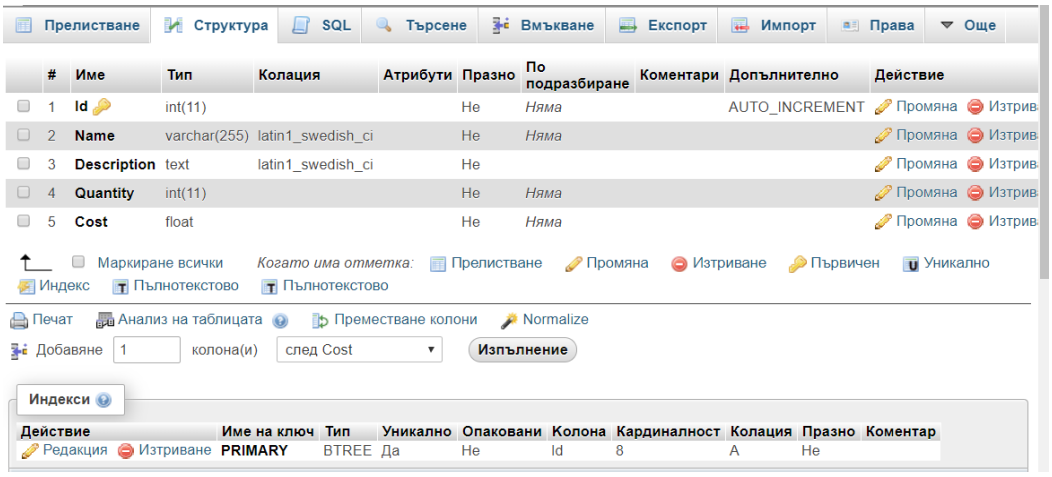
II. ПРАКТИЧЕСКА ЧАСТ

За целите на упражнението с phpMyAdmin е използвана база данни "*glasesrus*", съдържаща таблиците: "*customers*", "*products*", "*purchases*" и "*purchasesproducts*". Таблицата "*customers*" съхранява информация за клиентите на даден магазин, "*products*" – за продуктите които се продават, "*purchases*" – за поръчките на клиентите, а таблица"*purchasesproducts*" е връзката между продуктите и поръчките. Структурата на таблиците е показана на *фиг.8.1 а), б), в), г),* а съдържанието им на *фиг.8.2 а), б), в), г)*. Задачата която е решена е следната:

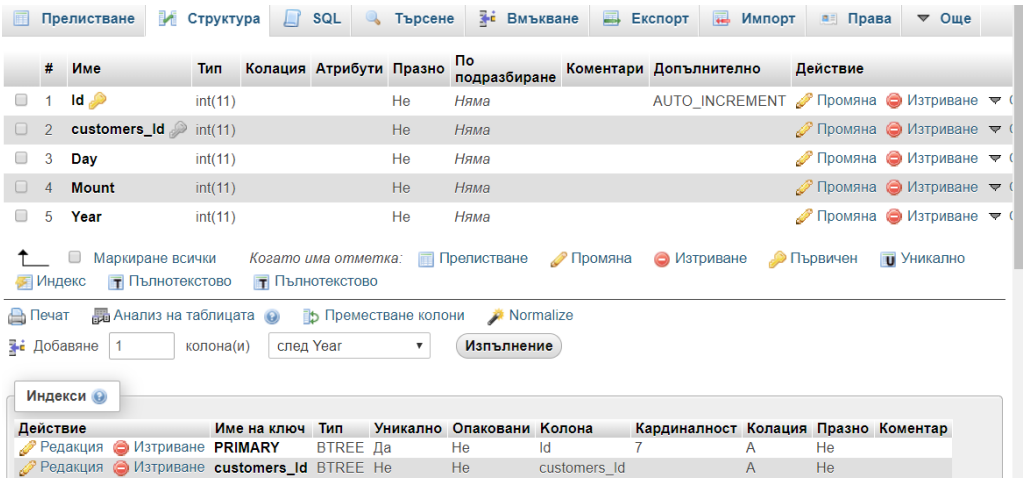
ЗАДАЧА: *Изведете всички поръчки на клиенти със Surname "Jones" от базата данни* ***glasesrus****,.*



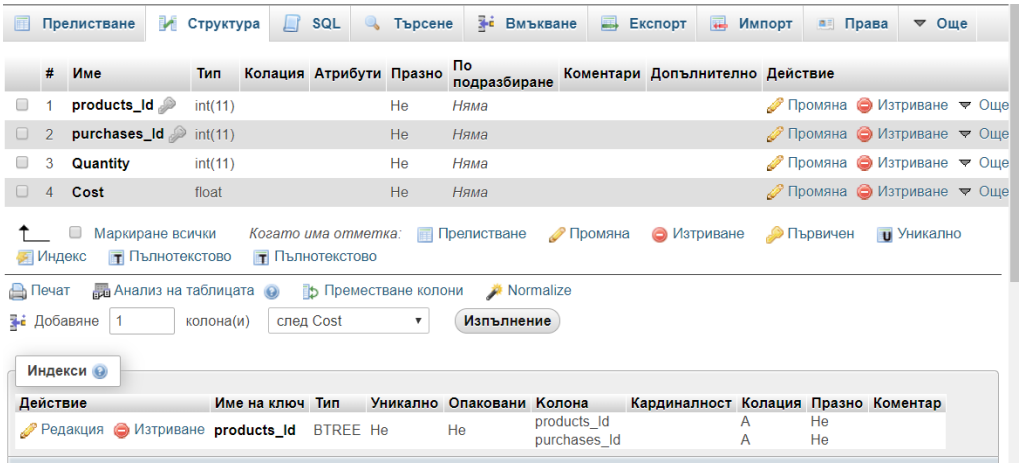
фиг.8.1.а) Структура на таблица "customers"



фиг.8.1.б) Структура на таблица "products"



фиг.8.1.в) Структура на таблица „purchases“



фиг.8.1.г) Структура на таблица „purchasesproducts“

	Id	Title	Surname	Firstname
Редакция Копиране Изтриване	1	Mrs	Smith	Lunne
Редакция Копиране Изтриване	2	Miss	Jones	Ann
Редакция Копиране Изтриване	3	Mr	Brown	Simon
Редакция Копиране Изтриване	4	Mr	Smith	David
Редакция Копиране Изтриване	5	Mr	Bell	Peter
Редакция Копиране Изтриване	6	Mr	Hal	Elizabeth
Редакция Копиране Изтриване	7	Mr	Smith	Kevin
Редакция Копиране Изтриване	8	Mr	Jones	Jack
Редакция Копиране Изтриване	9	Mr	Green	Wiliam
Редакция Копиране Изтриване	10	Mr	Bell	Simon
Редакция Копиране Изтриване	11	Mr	Brown	Ian
Редакция Копиране Изтриване	16	Mr	asd	sda

Фиг.8.2.а) Записи в таблица “customers”

	Id	Name	Description	Quantity	Cost
Редакция Копиране Изтриване	1	Beer Glass	600ml Beer Glas	345	3.99
Редакция Копиране Изтриване	2	Wine Glass	125ml Wine Glass	236	2.99
Редакция Копиране Изтриване	3	Wine Glass	175ml Wine Glass	436	3.5
Редакция Копиране Изтриване	4	Shot Glass	50ml small Glass	132	1.5
Редакция Копиране Изтриване	5	Spirit Glass	100 ml spirit Glass	489	2.5
Редакция Копиране Изтриване	6	Long Glass	200 ml Tall Glass	263	2.5
Редакция Копиране Изтриване	7	Beer Glass	300 ml Beer Glass	247	2.99
Редакция Копиране Изтриване	8	Wine Glass	225 ml Wine Glass	96	3.99

Фиг.8.2.б). Записи в таблица „products“

	Id	customers_Id	Day	Mount	Year
Редакция Копиране Изтриване	1	2	3	9	2006
Редакция Копиране Изтриване	2	4	6	9	2006
Редакция Копиране Изтриване	3	6	13	9	2006
Редакция Копиране Изтриване	4	2	22	9	2006
Редакция Копиране Изтриване	5	1	28	9	2006
Редакция Копиране Изтриване	6	9	1	10	2006
Редакция Копиране Изтриване	7	7	1	10	2006

Фиг.8.2.в). Записи в таблица „purchases“

products_Id	purchases_Id	Quantity	Cost
2	1	20	2.99
3	2	10	3.5
8	2	30	4.5
6	3	25	2.5
3	4	10	3.5
4	4	100	1.5
5	4	40	3
1	5	22	3.99
1	6	6	3.99
3	7	15	3.5
4	7	25	2
5	7	10	2.5
7	7	55	2.5
8	7	1	3.99

Фиг.8.2.г).Записи в таблица „purchasesproducts“

КОД

```
<?php
//------------------------------ Част 1 ----------------------------
        require_once("database.php");
//------------------------------------------------------------------

//------------------------------ Част 2 ----------------------------
        $strSurname='Jones';
        $queryCust="SELECT * FROM customers WHERE Surname='$strSurname' ";
        $dbCustRecords=mysqli_query($con,$queryCust);
        if (!$dbCustRecords) die('Неуспешно запитване. Получи следната
грешка'.mysqli_error($con));
//------------------------------------------------------------------

//------------------------------ Част 3 ----------------------------
```

```
        While ($CustRecords=mysqli_fetch_array($dbCustRecords))
        {
                $intCustId=$CustRecords["Id"];
                echo "<p> Customers: ";
                echo $CustRecords["Title"]." ";
                echo $CustRecords["Surname"]." ";
                echo $CustRecords["Firstname"]."</p>";
//-----------------------------------------------------------------

//------------------------------ Част 4 ----------------------------
                $queryPur="SELECT * FROM purchases WHERE
customers_Id='$intCustId'";
                $dbPurRecords=mysqli_query($con,$queryPur);
                if (!$dbPurRecords) die('Неуспешно запитване. Получи следната
грешка'.mysqli_error($con));
//-----------------------------------------------------------------

//------------------------------ Част 5 ----------------------------
                while ($PurRecords=mysqli_fetch_array($dbPurRecords))
                {
                        $intPurId=$PurRecords["Id"];
                        echo "<p> Purchases On: ";
                        echo $PurRecords["Day"]." ";
                        echo $PurRecords["Mount"]." ";
                        echo $PurRecords["Year"]."</p>";
//-----------------------------------------------------------------

//------------------------------ Част 6 ----------------------------
                        $queryPurPro="SELECT * FROM purchasesproducts WHERE
purchases_Id='$intPurId'";
                        $dbPurProRecords=mysqli_query($con,$queryPurPro);
                        if (!$dbPurProRecords) die('Неуспешно запитване. Получи
следната грешка'.mysqli_error($con));
//-----------------------------------------------------------------

//------------------------------ Част 7 ----------------------------
                    while($PurProRecords=mysqli_fetch_array($dbPurProRecords))
                        {
                                $intProId=$PurProRecords["products_Id"];
                                echo "<p>".$PurProRecords["Quantity"]." ";
//-----------------------------------------------------------------

//---------------------------- Част 8 ----------------------------
                                $queryPro="SELECT * FROM products WHERE
Id='$intProId'";
                                $dbProRecords=mysqli_query($con,$queryPro);
                                if (!$dbProRecords) die('Неуспешно запитване. Получи
следната грешка'.mysqli_error($con));
//-----------------------------------------------------------------
```

```
//---------------------------- Част 9 ----------------------------
                                $ProRecords=mysqli_fetch_array($dbProRecords);
                                echo $ProRecords["Name"]." (".$ProRecords["Description"].")
at &#163;";
                                echo $ProRecords["Cost"]." each. </p>";
                        }
                }
        }
//-----------------------------------------------------------------

//---------------------------- Част 10 ----------------------------
        mysqli_close($con);
//-----------------------------------------------------------------
?>
```

В първа част от кода се осъществява достъп до файл "*database.php*". В него се съдържа скрипта, който създава връзка към MySQL сървъра. Във втората част на скрипта са декларирани две променливи *$strSurname* и *$queryCust*. В първата се посочва името на клиента за който се търси информация, а във втората – необходимата SQL заявка. Тя се изпраща към сървъра чрез функцията **mysqli_query()**. При неуспешна реализация на заявката скриптът се прекратява и се извежда съобщение за грешка.

В третата част на кода чрез оператор за цикъл **while** и функцията **mysqli_fetch_array()** се изваждат имената и титлата на клиентите. Тяхното ID се съхранява в допълнителна променлива `$intCustId`. То е използвано в четвърта част на кода за създаване на заявка към таблица purchases, която да извлича всички покупки съдържащи идентификатора на избрания клиент.

В пета част се извеждат датите на които са направени покупките. Идентификационният ключ на покупките се съхранява в променливата `$intPurId`. Той е използван за създаване на нова заявка към таблица "*purchasesproduct*" в шестата част. Заявката извлича цялата информация от таблицата (седма част от кода). В нея идентификаторът на продукта се използва за да се създаде последната заявка към таблица „*Products*". Заявката е изпратена към сървъра в осма част и изведена в девета. Функцията **mysqli_fetch_array()** от девета част не е използвана с оператора за цикъл **while** понеже е известно че резултатът от заявката ще бъде само един запис.

В десета част се затваря връзката към MySQL сървъра. Резултатът от изпълнението на PHP скрипта е визуализиран на *фиг.8.3*.

Фиг.8.3. Резултат от решението на задачата

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Допълнете скрипта от практическата част, така че да генерира подходящи съобщения за грешка в случай, че в някой от етапите не бъде открит запис. Пример „*Не са направени покупки*“, или „*Няма клиенти с такова име*“ и т.н.
   Упътване: *За проверка на съществуващи записи използвайте функцията* ***mysqli_num_rows()***.
2. Променете скрипта от практическата част така, че потребителите да имат възможност за избор на клиентите по име. Използвайте потребителска форма с падащ списък.
3. Като използвате базата данни „*Firma*“ от *тема 6* създайте PHP скрипт, който извлича описанието на продукта от третия запис в таблицата “*Products*”.
   Упътване*: За целта използвайте функциите* ***mysqli_data_seek()*** *и* ***mysqli_fetch_row()***.

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Колко параметъра приема функцията **mysqli_fetch_row()**?
2. Как се достъпва информацията върната от функцията **mysqli_fetch_row()**?
3. Какво е предназначението на функцията **mysqli_data_seek()** и колко параметъра приема?
4. Какво представлява резултатът от функцията **mysqli_num_rows()**?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 9. Разработка на модул за вход

ЦЕЛ

Студентите да придобият практически опит и умения за разработване на модули за вход чрез HTML и PHP скрипт.

I. ТЕОРЕТИЧНА ЧАСТ

Потребителите могат да имат различни привилегии на достъп до функционалността на информационните системите. За да се избегне неоторизиран достъп е необходимо да се направи автентикация на съответния потребител. Това се извършва чрез форми за вход.

Един от начините за реализация на такива форми е чрез HTML тагове. В практическата част е представено създаването на форма за вход и нейната комуникация чрез PHP скрипт.

II. ПРАКТИЧЕСКА ЧАСТ

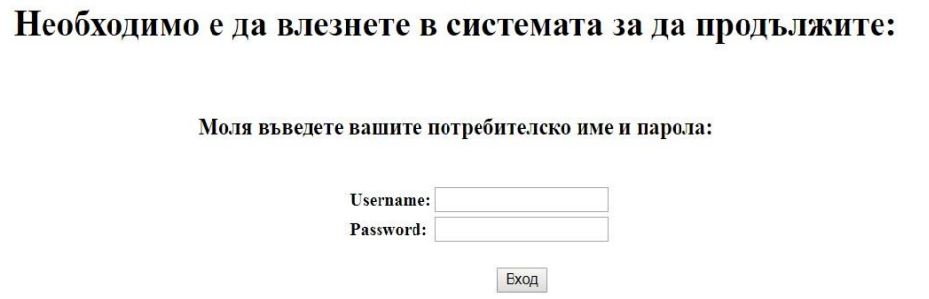
За целите на упражнението в MySQL сървъра предварително е създадена база данни „*BD*“ с таблица „*Users*“. Таблицата се състои от следните полета: *Username*, *Password*, *email*, *Role*. Типа на полетата е зададен спрямо тяхното предназначение. В таблицата са въведени следните данни.

Табл.9.1 Users

Username	*Password*	*Email*	*Role*
Admin	*Admin123456*	*Admin_test@example.com*	*admin*
User	*Pasusers123*	*User_test@example.com*	*users*

Чрез полето “*Role*” се задават привилегиите на достъп за съответния потребител. То е използвано в следващите теми. Решената задача е следната:

ЗАДАЧА: *Разработете форма за вход подобна на фиг.9.1. Създайте PHP скрипт който да проверява дали въведените данни в полетата съвпадат с тези от БД. Променете скрипта така, че при коректно зададени потребителско име и парола да изведе съобщението „Здравей, (Admin/Users) вие успешно влязохте в системата“, а в противен случай – „Невалидни потребителско име или парола“.*



Фиг.9.1 Форма за вход.

КОД

```
//----------------------file forma.php-------------------------
<html>
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8">
<title>Форма за вход</title>
</head>
<body>
   <div style="width:1400px">
                <div style="text-align: center">
                 <br>
                 <h1>Необходимо е да влезнете в системата за да продължите:</h1>
                 <br>
                       <p style="font-weight:bold;font-size:20px">Моля въведете
вашите потребителско име и парола:</p><br>
            </div>
                  <div style="height:300px;width:600px;float:left;">
            </div>
                   <div style="height:210px; width:800px; float:left;">
                    <form method="post" action="script.php">
                          <table style="text-align:left;font-weight:bold">
                    <tr>
                             <td><label for="users"> Username: </label></td>
                                 <td><input type="text" name="Username" id="users"></td>
                    </tr>
                    <tr>
                           <td><label for="pass"> Password: </label></td>
                                 <td><input type="password" name="Pass" id="pass"></td>
                    </tr>
                    <tr>
```

```
                    <td></td>
                        <td style="text-align:center">
                     <br>
                     <input type="submit" name="submit" value="Вход"> </td>
                  </tr>
                 </table>
               </form>
             </div>
           </div>
  </body>
</html>
<?php
//-------------------------file script.php----------------------
        $con=mysqli_connect('localhost','root','','BD');
        if(!$con)  die('Неуспешно свързване. Получи се следната грешка:
'.mysqli_connect_error());
        $query="Select*From Users";
        $dbreg=mysqli_query($con,$query);
        If(!$dbreg) die('Получи се следната грешка'.mysqli_error($con));
        $users=$_POST['Username'];
        $pass=$_POST['Pass'];
        while($value=mysqli_fetch_array($dbreg))
                if($users==$value['Username'])  break;
        if($users==$value['Username']  and  $pass==$value['Password'])  echo  'Здравей
'.$value['Username'].' Вие успешно влезнахте в системата';
        else echo "Невалиден потребителско име или парола";
        mysqli_close($con);
?>
```

Файлът „*forma.php*“ създава форма чрез HTML тагове. В тях са вградени елементи за стил на формата. Тагът <***div***> разделя страницата от браузъра на отделни блокове. В скриптовия файл се извършва проверката за автентикацията на потребителя и се извежда резултата.

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

Като използвате задачата от практическата част:

1. Добавете цветове на отделните тагове **div** във файла “*forma.php*” и анализирайте резултата от изпълнението.
2. Създайте допълнителни възможности за забравена парола и за регистрация на нов потребител във формата за вход.

3. Разработете допълнителна форма за регистрация на нови потребители. Създайте PHP скрипт към нея, който да записва информацията за новия потребител в базата данни.
4. Разработете допълнителна форма за възстановяване на забравена парола. Създайте PHP скрипт към нея, който да проверява дали съществува такъв потребител.

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Какво е предназначението на формите за вход?
2. Като имате предвид задачата от практическата част, каква е следващата стъпка след успешната реализация на връзката към MySQL сървъра?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 10. Разработка на модул за въвеждане на информация

ЦЕЛ

Студентите да придобият практически опит и умения за разработване на модули за въвеждане на информация в MySQL база данни чрез HTML и PHP скрипт.

I. ТЕОРЕТИЧНА ЧАСТ

Информационните системи се състоят от различни модули. Един от тях може да бъде форма за въвеждане на информация. Този модул се състои от текстови полета, които позволяват потребителя да въвежда данни в тях. Чрез скрипт вграден под различни HTML елементи (най-често бутон) тази информацията се записва в базата данни.

В практическата част е демонстрирано създаването на форма, която предоставя възможност за записването на данни в MySQL сървъра.

II. ПРАКТИЧЕСКА ЧАСТ

За целите на упражнението предварително е създадена база данни „*ShoppingCenter*“ с таблици: „*Employees*“ и „*Delivery*“. Полетата на таблиците са следните:

- *Employees – EmployeeID, FirstName, SecondName, LastName, ContractNumber, Position, Address, Telephone;*
- *Delivery – ProductName, Quantity, Supplier, Price.*

ЗАДАЧА: *Да се разработи модул, който позволява въвеждането на необходимата информация за назначаване на нови служители в търговски обект (магазин).*

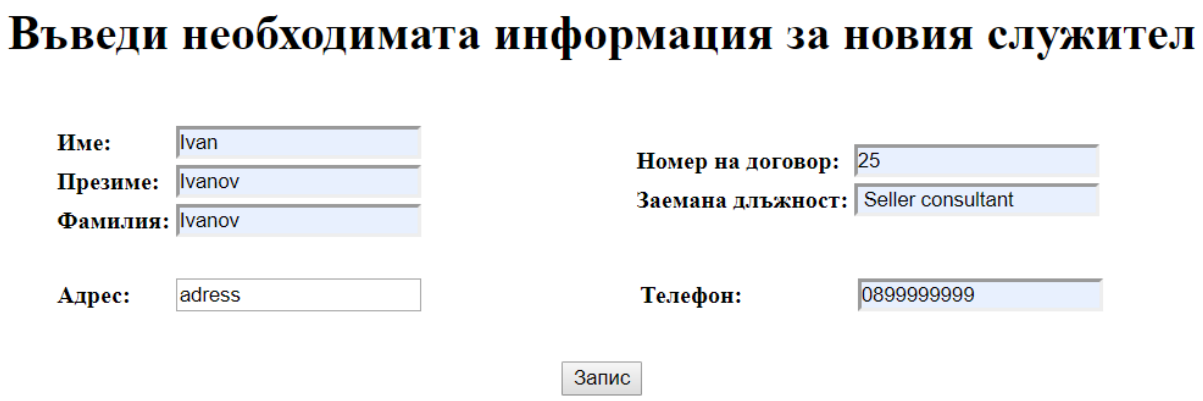
КОД

```
//-----------------------file forma.php-------------------------
<html>
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8">
<title>Нови служители</title>
</head>
<body>
   <div style="width:1000px">
        <div style="text-align:center">
          <h1>Въведи необходимата информация за новия служител</h1><br>
        </div>
        <form method="post" action="script.php">
          <div style="height:100px;width:140px;float:left;">
```

```
    </div>
    <div style="height:100px; width:380px; float:left;">
     <table style="text-align:left;font-weight:bold">
      <tr>
          <td><label for="FirstName"> Име: </label></td>
          <td><input type="text" name="FirstName" id="Firstname"></td>
      </tr>
      <tr>
          <td><label for="SecondName"> Презиме: </label></td>
          <td><input type="text"name="SecondName"id="SecondName"></td>
      </tr>
      <tr>
          <td><label for="LastName"> Фамилия: </label></td>
          <td><input type="text" name="LastName" id="LastName"></td>
      </tr>
     </table>
    </div>
<div style="height:100px; width:380px; float:left;">
     <table style="text-align:left;font-weight:bold">
      <tr style="height:10px">
      </tr>
      <tr>
          <td><label for="dogovor"> Номер на договор: </label></td>
          <td><input type="text" name="dogovor" id="dogovor"></td>
      </tr>
      <tr>
          <td><label for="dlajnost"> Заемана длъжност: </label></td>
          <td><input type="text" name="dlajnost" id="dlajnost"></td>
      </tr>
     </table>
    </div>
<div style="margin-left:140px;">
     <table style="text-align:left;font-weight:bold">
      <tr>
          <td style =”width:75px”>
           <label for="adress"> Адрес: </label>
          </td>
          <td><input type="text" name="adress" id="adress"></td>
          <td style="width:135px"></td>
          <td=”width:140px”>
           <label for="telephon"> Телефон: </label>
          </td>
```

```
                <td><input type="text" name="telephon" id="telephon"></td>
            </tr>
            </table>
            </div>
            <div style="height:30px"></div>
            <div style="text-align:center">
            <span> <input type="submit" name="submit" value="Запис">
        </span>
            </div>
      </form>
        </div>
  </body>
</html>
<?php
//-------------------------file script.php----------------------
        $con=mysqli_connect('localhost','root','','ShoppingCenter');
        if(!$con) die('Неуспешно свързване. Получи се следната грешка:
'.mysqli_connect_error());
        $Fname=$_POST['FirstName'];
        $Sname=$_POST['SecondName'];
        $Lname=$_POST['LastName'];
        $NDog=$_POST['dogovor'];
        $dlajnost=$_POST['dlajnost'];
        $adress=$_POST['adress'];
        $tel=$_POST['telephon'];
        $query="INSERT INTO employees (EmployeeID, FirstName, SecondName,
LastName, ContractNumber, Position, address, Telephone) VALUES ('0', '$Fname', '$Sname',
'$Lname', '$NDog', '$dlajnost', '$adress', '$tel')";
        $dbreg=mysqli_query($con,$query);
        If(!$dbreg) die('Получи се следната грешка'.mysqli_error($con));
        else echo "Вие успешно добавихте информацията за новия служител";
        mysqli_close($con);
?>
```

Създадената форма за въвеждане на нови служители е показана на *фиг.10.1.* Бутонът „*Запис*“ задейства изпълнението скрипта от файла „*script.php*“. Той осъществява достъп до базата данни с наименование „*ShoppingCenter*“ и чрез INSERT заявка въвежда информация от формата в таблица „*employees*“.



Фиг.10.1 Модул за въвеждане на информацията за нови служители

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Редактирайте кода от практическата част така, че да проверява дали всички полета са попълнени коректно. При некоректно попълнени полета да се извежда съобщение за тяхната редакция.
2. Променете кода от практическата част, така че преди записването на информацията в базата да се проверява дали потребителя има необходимите права.
   Указание: *Може да използвате login формата създадена в тема 9. Приемете, че само администраторът може да добавя нови служители.*
3. Разработете модул, който да подпомага въвеждането на необходимата информация за доставянето на нова стока в търговския обект.
   Указание: *Необходимо е да достъпите таблицата „Delivery“ от базата данни. Тя е описана в началото на практическата част от упражнението.*

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Как се реализира записването на въведената информация в базата данни?
2. Какво е предназначението на последния if от PHP скрипта на решената задача в практическата част?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 11. Разработка на модул за обработка на потребителски заявки

ЦЕЛ

Студентите да придобият практически опит и умения за разработване на модули за обработка за потребителски заявки.

I. ТЕОРЕТИЧНА ЧАСТ

Почти всяка информационна система съдържа модули за обработка на потребителски заявки. Те позволяват подаването на заявките към сървъра за базата данни, както и извеждането на обработената информация.

Един от начините за реализация на такива форми е чрез HTML тагове. В практическата част е демонстрирано създаването на форма, която предоставя възможност за търсене на информация и извеждане на получения резултат.

II. ПРАКТИЧЕСКА ЧАСТ

За решаването на долната задача е използвана база данни „Магазин“ от *тема 10*. Към нея са добавени четири нови таблици: Клиенти, Продукти, Поръчки, Поръчки_Продукти. Тяхната структура е подобна на таблиците от *тема 8*.

ЗАДАЧА: *Да се разработи модул, който позволява на потребителите да извършват справки за:*

- *поръчките на точно определен клиент от базата данни;*
- *всички поръчки от дата, до дата.*

КОД

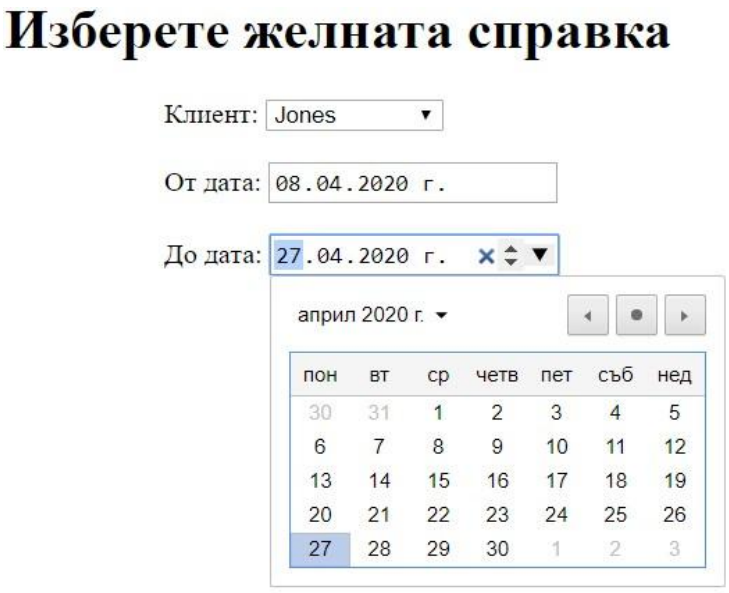
```
//-----------------------file forma.php--------------------------
<html>
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8">
<title>Форма за справки</title>
</head>
<body>
  <?php
        require_once("database.php");
        $queryCust="SELECT DISTINCT Surname FROM customers";
        $dbCustRecords=mysqli_query($con,$queryCust);
        if (!$dbCustRecords) die('Неуспешно запитване. Получи следната
грешка'.mysqli_error($con));
  ?>
```

```
<div style="width:1000px">
            <div style="text-align:center">
            <h1>Изберете желаната справка</h1><br>
        </div>
            <form method="post" action="script.php">
          <div style="height:130px;width:400px;float:left;">
           </div>
                  <div style="height:130px; width:600px; float:left;">
                <label for="Name"> Клиент:</label>
               <select id="Name">
                  <option value="noquery">Name</option>
                 <?php
                                While ($CustRecords=mysqli_fetch_array($dbCustRecords))
                   {
                 ?>
                      <option value="<?php echo $CustRecords['Surname'];
?>"><?php echo $CustRecords['Surname'];?></option>
                 <?php
                 }
                 ?>
                </select>
               <br><br>
                  <label for="date1"> От дата:</label>
                   <input type="date" name="date" id="date1">
               <br><br>
                   <label for="date2"> До дата:</label>
                    <input type="date" name="date" id="date2">
               <br>
          </div>
               <div style="text-align:center">
               <span> <input type="submit" name="submit" value="GO">
      </span>
      </div>
     </form>
        </div>
  </body> </html>
<?php
//----------------------file script.php-------------------------
        require_once("database.php");
        if (isset($_POST['submit']))
        {       If($_POST['name']!="Name") $strSurname=$_POST['name'];
                if (isset($_POST['date1'])) $ot_data=$_POST['date1'];
                if (isset($_POST['date2'])) $do_data=$_POST['date2'];
```

```
        }
        $queryCust="SELECT * FROM customers WHERE Surname='$strSurname' ";
        $dbCustRecords=mysqli_query($con,$queryCust);
        if (!$dbCustRecords) die('Неуспешно запитване. Получи следната грешка
'.mysqli_error($con));
        While ($CustRecords=mysqli_fetch_array($dbCustRecords))
        {
……
……
……
?>
```

Създадената форма чрез файла "*forma.php*" е показана на *фиг.11.1*. Модулът предоставя възможност на потребителя да избере име на клиент или времеви диапазон в дати за необходимата справка. Изборът на клиент се извършва чрез падащ списък реализиран с таговете **select** и **option**. За създаването на полетата за дата са използвани **input** тагове със стойност за атрибут ***type****="date"*. При достъпването на полето от потребителя се визуализира календар, който подпомага неговия избор (*фиг.11.1*).

В скриптовия файл „*script.php*" са направени проверки за празни полетата във формата.



Фиг.11.1 Модул за обработка на потребителска заявка

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Допълнете кода от практическата част, така че да се извеждат всички покупки на потребителя за избрания времеви диапазон.
2. Създайте форма, която позволява на потребителя да изведе всички стоки за които наличността им е под определен брой.

   Упътване: *Нека потребителя да има възможност да посочва определения брой от стоки:*

3. Създайте форма, която позволява на потребителя да изведе всички стоки за които цената им е под определена стойност.
   Упътване: *Нека потребителя да има възможност да посочва стойността на цената:*
4. Създайте модул за обработка, който да позволява на потребителя да извежда стоки от определен вид и да ги сортира по негова преценка (възходящ или низходящ ред)
   Упътване: *Използвате бутон за сортиране на елементите по избраната преценка на потребителя.*

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Как са реализирани полетата за дата при създаването на формата от фиг.11.1?
2. Какво е предназначението на първите два реда от PHP скрипта на решената задача от практическата част?

КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ТЕМА 12. Защита и сигурност на информационните системи.

ЦЕЛ:

Запознаване на студентите с техники против злонамерени действия към информационните системите. Придобиване на практически знания и умения за по сигурни начини на разработване и защита на системите чрез HTML и PHP.

I. ТЕОРЕТИЧНА ЧАСТ

При разработването на една информационна система разработчиците са отговорни за сигурността на техния софтуер и защитата на данните му. Необходимо е да следват цялостна стратегия за повишаване на качеството на сигурността, която се състои от:

- развитие на добри навици за програмиране, които гарантират, че приложенията са реализирани правилно;
- познаване на различни форми на атака и вземане на мерки за защита от тях;
- идентификацията на потребителите при формите за вход;
- реализация на сигурен механизъм за защита на паролите.

Добри практики при разработката на система чрез PHP скрипт са:

1. Докладване на всички грешки.

Необходимо е да се съобщава за всички възникнали грешки и предупреждения, които интерпретаторът на PHP генерира. В някои версии на PHP предупрежденията не са включени и се налага да се направи редакция в конфигурационния файл „*php.ini*“. При инсталацията на WAMP пакет, файлът се достъпва от менюто PHP. За зареждането му е необходимо да се използва текстов редактор. Редакциите в него трябва да се правят внимателно. Необходимо е да се намери раздела „*Error handing and loging*“.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; This directive informs PHP of which errors, warnings and notices you would like
; it to take action for. The recommended way of setting values for this
; directive is through the use of the error level constants and bitwise
; operators. The error level constants are below here for convenience as well as
; some common settings and their meanings.
; By default, PHP is set to take action on all errors, notices and warnings EXCEPT
; those related to E_NOTICE and E_STRICT, which together cover best practices and
; recommended coding standards in PHP. For performance reasons, this is the
; recommend error reporting setting. Your production server shouldn't be wasting
```

; resources complaining about best practices and coding standards. That's what
; development servers and development settings are for.
; Note: The php.ini-development file has this setting as E_ALL. This
; means it pretty much reports everything which is exactly what you want during
; development and early testing.
;
; Error Level Constants:
; E_ALL - All errors and warnings (includes E_STRICT as of PHP 5.4.0)
; E_ERROR - fatal run-time errors
; E_RECOVERABLE_ERROR - almost fatal run-time errors
……
; Common Values:
; E_ALL (Show all errors, warnings and notices including coding standards.)
; E_ALL & ~E_NOTICE (Show all errors, except for notices)
; E_ALL & ~E_NOTICE & ~E_STRICT (Show all errors, except for notices and coding standards warnings.)
; E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR (Show only errors)
; Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
; Development Value: E_ALL
; Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT
; http://php.net/error-reporting
error_reporting = E_ALL

За да се получават всички възможни грешки е необходимо стойността на *error_reporting* да бъде *E_ALL*. Разгледайте задача 1 от практическата част

2. Валидиране на данните от формуляр

Едно от основните средства за неправомерен достъп са разработените формуляри. Те се състоят от различни полета в които потребителя въвежда данни. Разработчикът, обаче не може да бъде сигурен за коректното им попълване. Необходимо е да се извърши валидиране на полетата, т.е. да се предвидят проверки, гарантиращи коректността на данните.

Представянето на некоректни данни може да бъде и въвеждането на HTML тагове или SQL елементи в текстови полета. Това може да промени действието на съответния код или дори да доведе до неоторизиран достъп до сървъра на базата данни. За преодоляването на тези проблеми може да се използват следните функции:

- *htmlentities(string низ)* – функцията преобразува всички приложими знаци в HTML обекти. Примерно знакът „<“ няма да бъде разпознат като HTML елемент и ще бъде интерпретиран като знака по-малко „<“.

- *mysqli_escape_string()* – функцията преобразува специалните SQL знаци в обикновен символ от низ.

3. Идентификация на потребители и пароли.

Проблемите свързани с идентификацията на потребителите може да се избегнат по следния начин:

- ***чрез проверка за референцията***

Гарантира, че потребителят се е идентифицирал само чрез формата за вход. Проверява се стойността на променливата на обкръжението **$_SERVER[“HTTP_REFERER”]**. Ако се установи, че потребителя е пропуснал идентификацията, той автоматично се препраща към формата за вход.

- ***чрез съхраняване на паролите в криптиран вид***

Съхраняването на паролите в криптиран вид гарантира тяхната сигурност. Дори да се извърши неправомерен достъп до базата данни те няма да могат да бъдат директно разчетени. Един често използван подход за криптиране са **hash** алгоритмите. Те представляват функции които кодират информацията. **Hash** алгоритми може да се използват чрез функциите **sha1()** и **md5()**.

- ***чрез използване на сигурни пароли***

За повишаване защитата на информационните системи потребителите трябва да използват сигурни пароли. Разработваната система трябва да изисква от потребителите да използват надеждна парола, която да отговаря на определени условия. Такива могат да бъдат: използването на определен брой символи; съдържание на главни и малки букви, специални символ и др. Контролирането на тези изисквания може да се извършва чрез функциите за регулярни изрази. Една такава функция е **preg_match_all()**, която проверява дали определен израз се съдържа в низ. Тя приема два параметъра. Първият е шаблонен израз, вторият е низ. Функцията връща като резултат броя на повторенията на шаблонния израз в низа.

Допълнителна информация относно функциите за регулярни изрази може да се достъпи на адрес: https://www.php.net/manual/en/ref.pcre.php.

II. ПРАКТИЧЕСКА ЧАСТ

ЗАДАЧА 1: *Дефинирайте променлива без да и задавате начална стойност. Проверете с оператор if дали стойността на дадената променлива е 0.*

КОД

```
<?php
        $intNoValue;
        If ($intNoValue==0)
                echo “<p>Equals zero</p>”
?>
```

Ако не е включено докладването на всички възникнали грешки, то кодът би се изпълнил и изведеният резулат ще бъде „*Equals zero*“. При включването на този доклад от грешки, кодът ще изведе предупреждение, че променливата не е инициализирана. Лоша практика на програмиране е да оставят неинициализирани променливи.

ЗАДАЧА 2: *Създайте формуляр съставен от три елемента: поле за потребителско име, парола и бутон. Създайте скрипт, който да проверява коректността на попълнените данни в полетата. Изведете съобщението"Hello World", само ако не е допусната нито една грешка в коректността на данните.*

КОД

```
<html>
<h1>Please login:</h1>
<form action="<?php echo $_SERVER["PHP_SELF"]?>" method="post">
<label for="users">USERNAME:</label>
<input name="strUserName" type="text" id="users" maxlength='9'>
<label for="pass">PASSWORD:</label>
<input name="strPassword" type="password" id="pass" maxlength='12'>
<input name="Submit" type="Submit">
</form>
<?php
if(isset($_POST["Submit"])){
        $Username=$_POST["strUserName"];
        $Password=$_POST["strPassword"];
        $intErrors=0;

        if(strlen($Username)<4){
                echo "Username less then 4 characters";
                $intErrors++;
        }
        if(strlen($Password)<6){
                echo "Password less then 6 characters";
                $intErrors++;
        }
        If($Username{0}!="5"){
                echo "Username doesn't start with a 5";
                $intErrors++;
        }
        $parrent="/[A-Z]/";
        if(preg_match_all($parrent, $Password)==0){
                echo "The password doesn't contain upper-case letters";
                $intErrors++;
        }
        If($intErrors==0)
                echo "Hello World";
}
?>
```

В скриптовия файл са направени следните примерни ограничения:

- Потребителското име трябва да съдържа между 4 и 9 символа, както и да започва с цифра 5;
- Дължината на паролата трябва да съдържа между 6 и 12 символа и поне една главна буква.

Ограничението за максимален брои символи в текстовите полета е постигнато чрез свойството ***maxlength*** на таг **input**. Функцията **strlen()** определя броя на символите в съответното поле. Чрез *$Username{0}* се достъпва първият елемент от потребителското име. Чрез функцията **preg_match_all()** се определят броят на главните букви в полето за парола. Променливата *$interrors* служи за брояч на възникналите грешки.

ЗАДАЧА 3: *Създайте форма с два елемента: многоредово текстово поле и бутон. В полето въведете следния текст "<i> Hello, this is in italics". Създайте PHP скрипт, който да извежда съобщението от текста. Анализирайте резултата и посочете начин за преодоляването на получения проблем.*

КОД

```
<html>
<body>
        <h2> Please enter your message:</h2>
        <form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='POST'>
        <p>
                <label for="strMessage"> Message: </label>
                <textarea name="strMessage" id="strMessage"></textarea>
        </p>
        <p><input type='submit' name='submit'/></p>
        </form>
</body>
</html>
<?php
if (isset($_POST["submit"])){
        $strMessage=$_POST['strMessage'];
        echo "<p>Your Message is $strMessage.</p>";
}
echo "<p>This is some text further down the web page</p>";
?>
```

Текстът добавен в полето съдържа HTML таг за накланяне на текста. Тъй като липсва затварящ таг, той повлиява на цялата форма. Резултатът е показан на *фиг.12.1*.

Please enter your message:

Message:

Изпращане

Your Message is *Hello, this is in italics.*

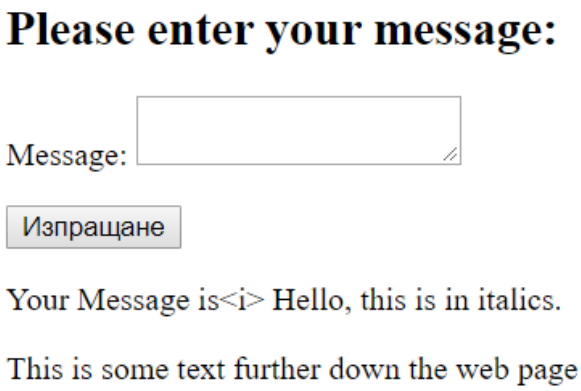
This is some text further down the web page

Фиг.12.1 HTML вкарвания на данни

Проблемът може да се реши чрез използване на функция „*htmlentities(string низ)*“ при командата за извеждане на съответното съобщение. Това ще стане по следния начин:

```
echo "<p>".htmlentities("Your Message is".$strMessage).".</p>";
```

На *фиг.12.2* тагът **<i>** се интерпретира като част от символния низ.



Фиг.12.2. Предотвратяване на HTML вкарване на данни

За изпълнението на задача 4 предварително е създадена база данни с таблица “Users”. Записите в нея са следните:

Табл.12.1 Users

Username	***Password***
Alan	*NULL*
David	*Regan*

ЗАДАЧА 4: *Създайте PHP скрипт, който ръчно задава стойности на двете променливи* `$strUsername` *и* `$strPassword`*. Допълнете скрипта така, че да проверява дали стойностите съответстват на потребител от базата данни. Тествайте кода като зададете стойност за променливата* `$strUsername`*: „’ OR ‘ 1=1“, променливата за парола празен низ. Анализирайте получения резултат от изпълнението на кода. Предложете мерки за преодоляване на проблема.*

КОД

```
<?php
$strUsername='David';
$strPassword='Regan';
$dblocalhost=mysqli_connect("localhost","root","","users")
or die ("Could not connect:".mysqli_connect_error());
$dbRecords=mysqli_query($dblocalhost,"Select * FROM users WHERE
Username='$strUsername'");
$arrRecords=mysqli_fetch_array($dbRecords);
if($strPassword != $arrRecords["Password"])
        echo "<p>Invalid Password/Username</p>";
else echo "<p>Password and Username match!</p>";
?>
```

Скриптът реализира проверка на потребителското име и парола. При тестването на скрипта с коректните данни той ще се изпълни. Резултат от тестването на скрипт няма

да бъде очаквания, ако зададената стойност за променливата `$strUsername` е *' OR '1=1*, променливата `$strPassword` – приема празен низ. Това е така понеже изпратената заявка към MySQL сървъра се променя на следната:

*SELECT * FROM users WHERE Username='' OR '1=1';*

Заявката ще извлече като резултат всички записи от таблицата, от които първият съответства на посочената парола в `$strPassword`.

Манипулирането на SQL заявки може да се избегне чрез функцията ***mysqli_escape_string****(низ)*. Тя добавя **escape** оператори преди всеки от апострофите. Новата SQL заявка ще изглежда така:

*SELECT * FROM users WHERE Username='\' OR \'1=1' ESCAPE '/'*

Преди да се подаде съдържанието на променливата `$strUsername`, следва да се постави следния ред в скрипта:

```
$strUsername=mysqli_real_escape_string($strUsername);
```

III. ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Коригирайте грешките в дадения скрипт и го изпълнете. Анализирайте получения резултат.

```
<?php
$intCount;
$arrFirstSurname = array(Simon=>"Jones", James=>"Smith", Alan=>"Barclay", Gema =>"West");
For($intA=0;$intA<=$intCount;$intA++)
Echo $arrFirstSurname[Alan]. " ";
?>
```

2. Създайте формуляр със следните три елемента: падащ списък, текстово поле и бутон. Падащият списък трябва да съдържа следните титли: *Mr*, *Miss*, *Mrs*, *Dr* and *Other*. Създайте PHP скрипт, който да извежда избраната титла. Ако от списъка е избран елемент *Other*, то да се изведе съдържанието на текстовото поле. Направете текстовото поле което да съдържа между 2 и 5 символа без цифри.
3. Криптирайте паролите в базата данни от задача 4. Създайте PHP скрипт който прочита криптирани пароли.

   Упътване: *Използвайте функциите за двата hash алгоритъма SHA1() и MD5()*
4. Създайте скрипт, който определя силата на генерирана парола.

   Упътване: *Определете силата на паролата като поставяте по 1 точка за всяко от следните изисквания:*
   - *Минимална дължина (не по малко от 6 символа)*
   - *Съдържа главни букви*
   - *Съдържа малки букви*
   - *Съдържа поне едно число*
   - *Съдържа поне един от следните символи !$%^&*@#+*

IV. ВЪПРОСИ ЗА САМОПОДГОТОВКА

1. Какво е предназначението на функцията **htmlentities()**?
2. Как може да се криптират паролите чрез **hash** алгоритми?
3. Какво е предназначението на функцията **preg_match_all()** и колко параметъра приема тя?
4. Как може да се контролира въведената информация в текстовите полета?
5. Какво е предназначението на функцията **mysqli_escape_string()**?

ЛИТЕРАТУРА

1. Academy D.K., Съвременни подходи за програмиране на PHP 7, Асеневци, 2018
2. Academy D.K.,Laravel – създаване на PHP приложения, Асеневци, 2022
3. Арнаудов Д. , А. Крумова - Сигурност и защита на информационните системи, ВСУ "Ч. Храбър", 2007
4. Георгиева А., Информационни системи и бази данни, ТУ-София, 2006
5. Джамбазов Венцислав, Уеб базирани потребителски интерфейси, Нов Български Университет, София, 2012
6. Илиева С., В. Лилов, И. Манова, Подходи и методи за реализация на софтуерни системи, Колектив, УИ "Св. Климент Охридски", София, 2010
7. Колисниченко Д., PHP 7 & MySQL - практическо програмиране, Асеневци, 2018
8. Колисниченко Д. MySQL 8 – практическо програмиране в примери, Асеневци, 2021
9. Контиери М., Clean Code Cookbook, Асеневци, 2024
10. Орозова Д., Бази от данни, Бургаски Свободен Университет, 2011
11. Туджаров Х. ,И С. Калчев, Информационни системи – анализ и проектиране, ПИК, В. Търново, 1999
12. Филипова Н., С. Парушева, Я. Александрова, Основи на информационните системи, Наука и икономика, Икономически университет Варна, 2017
13. Цеков Л. , Информационни системи – тезиси и лекции, ТУ-Габрово, 2000
14. Соловьев И. В., А. А. Майоров, Проектирование информационны систем. Фундаментальны курс, Академ. Проект, Москва, 2009