PHP I - zajęcia 7 – Tablice + operacje na łańcuchach znakowych

1. **Poruszanie się wewnątrz tablicy**
   - ★ current() / pos() - funkcja zwraca element tablicy na którym obecnie znajduje się wskaźnik
   - ★ reset() - funkcja ustawia wskaźnik w pierwszym elemencie tablicy
   - ★ end() - funkcja ustawia wskaźnik w ostatnim elemencie tablicy
   - ★ next() - funkcja przesuwa wskaźnik na kolejny od bieżącego w tablicy
   - ★ prev() - funkcja przesuwa wskaźnik na poprzedni od bieżącego w tablicy

```php
<?php
$ludzie = array("Piotr", "Jan", "Adam", "Marek");

echo current($ludzie) . "<br>"; // obecnym elementem jest Piotr
echo next($ludzie) . "<br>"; // Nastepnym elementem dla Piotra jest Jan
echo current($ludzie) . "<br>"; // Obecnym elementem jest Jan
echo prev($ludzie) . "<br>"; // Poprzedzajacym Jana elementem jest Piotr
echo end($ludzie) . "<br>"; // Ostatni element to Marek
echo prev($ludzie) . "<br>"; // Poprzedzającym Marka elementem jest Adam
echo current($ludzie) . "<br>"; // Obecnym elementem jest Adam
echo reset($ludzie) . "<br>"; // Ustawia wewnetrzny wskaźnik tablicy na
pierwszy element
echo next($ludzie) . "<br>"; // Nastepnym elementem dla Piotra jest Jan

?>
```

2. **Dołączenie dowolnej funkcji do każdego elementu tablicy**
   - ★ array_walk($Tablica, “Nazwa_funkcji”, [dane_uzytkownika]) - funkcja array_walk pozwala zastosować funkcję dla każdego z elementów tablicy. Istnieje tez opcjonalny 3ci parametr funkcji w którym możemy wprowadzić dodatkowe dane.

```php
<?php
function mojaFunkcja($value,$key,$dodatkowyParametr)
{
echo "$key $dodatkowyParametr $value<br>";
}
$a=array("a"=>"czerwony","b"=>"zielony","c"=>"niebieski");
array_walk($a,"mojaFunkcja","ma kolor");
?>
```

   - ★

★

3. **Filtrowanie elementów tablicy**
   - ★ array_filter($Tablica, “nazwa funkcji”) - odfiltrowuje elementy tablicy, które dają prawdę dla funkcji w drugim parametrze

```php
<?php
function nieparzyste($var)
{
    //zwraca nieparzyste
    return $var & 1;
}

function parzyste($var)
{
    // zwraca parzyste
    return !($var & 1);
}

$array1 = ['a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5];
$array2 = [6, 7, 8, 9, 10, 11, 12];

echo "nieparzyste :\n";
print_r(array_filter($array1, "nieparzyste"));
echo"<br>";
echo "parzyste:\n";
print_r(array_filter($array2, "parzyste"));
```

   - ★
4. **Liczenie elementów tablicy**
   - ★ count() / sizeof() - funkcja zwraca ilość elementów przekazanej im tablicy
   - ★ array_count_values() - zwraca tablice asocjacyjną częstości wystąpień poszczególnych wartości. Np.
     - $tablica(a,b,c,a);
     - $as=array_count_values($tablica);
     - $as ma postać (‘a’=>2,’b’=>1,’c’=>1)
5. **Manipulowanie łańcuchami znaków i wyrażenia regularne**
   - ★ *Ćwiczenie 1 - Utwórzmy następujący formularz pozwalający dodawać komentarze:*

   - ★
   - ★ *Oraz podstawowy skrypt pozwalający je obsługiwać: przetworzkomentarze.php*

```php
<?php
//utworzenei krótkich nazw zmiennych
$nazwa = $_POST['nazwa'];
$email = $_POST['email'];
$komentarz = $_POST['komentarz'];

//definicje danych statycznych
$adresdo = "komentarze@przyklad.pl";
$temat = "komentarze ze strony www";
$zawartosc = "Nazwa klienta: ".$nazwa."\n
    "."Adres pocztowy: ".$email."\n"."
    Komentarz klienta: ".$komentarz."\n";
$adresod = "serwerwww@przyklad.com";


//wywołanie funkcji mail() wysyłajacej
komentarz klienta
mail($adresdo, $temat, $zawartosc, $
    adresod);
?>
<!DOCTYPE html>
<html>
<head>
    <title>Komntarz przyjęty</title>
</head>
<body>
    <h1>Komentarz przyjęty</h1>
    <p>Państwa komentarz został wysłany</
    p>
</body>
</html>
```

- ★
- ★ mail() - funkcja natywna php służąca do wysyłania poczty elektronicznej.

6. **Przycinanie łańcuchów znakowych**
   - ★ trim() - usuwa znaki odstępu z początku i końca po czym zwraca łańcuch wynikowy. Można do niej przekazać jako drugi parametr listę znaków które mają zostać usunięte. Zastąpi ona domyślna listę z spacjami, tabulacją, znakami nowego wiersza
   - ★ ltrim() – działa podobnie jak trim() przy czym usuwa jedynie znaki początkowe (left)
   - ★ rtrim() / chop()- jak wyżej, przy czym usuwa jedynie znaki końcowe (right)
7. **Formatowanie łańcuchów znakowych**
   - ★ htmlspecialchars() - konwersuje znaki mające specjalne znaczenie w html na odpowiadające im symbole html. Np. Znak “<” zostanie przez nią zamieniony na symbol ”&lt”
   - ★ str_replace() - np. str_replace(“\r\n”, “”, “nazwa”) - zastąpi znaki nowego wiersza oraz powrotu karetki(końca linii) ich brakiem.
   - ★ nl2br() - zastępuje znaki końca wiersza html-owymi znacznikami
   - ★ *Ćwiczenie 2 – zabezpiecz obsługę naszego formularza korzystając z funkcji trim() dla wartości z niego pobranych oraz str_replace() dla zawartości a następnie poprawnie wyświetl treść komentarza na stronie używając funkcji htmlspecialchars() ora znl2br() (kolejność jest ważna)*
8. **Formatowanie łańcuchów znakowych do wyświetlania**
   - ★ echo – konstrukcja językowa zwracająca do przeglądarki dany ciąg znaków
   - ★ print() - działa tak samo jak echo ale z racji bycia funkcją zawsze zwraca 1
   - ★ printf() - wyświetla sformatowany ciąg znaków
   - ★ sprintf() - zwraca sformatowany ciąg znaków

- ★ Do powyższych funkcji można dołożyć dodatkowe parametry będące zmiennymi które chcemy wypisać np..
- ★ printf(“Wartość zamówienia wynosi %s”, $wartość);
- ★ Znak %s jest tzw. specyfikatorem konwersji zamienia on wartość na ciąg znaków.
- ★ %.2f - zwróci wartość w postaci liczby zmiennoprzecinkowej do drugiego miejsca po przecinku
- ★ Można użyć większej ilości specyfikatorów, ważna jest wtedy kolejność
- ★ Każdy specyfikatopr konwersji jest tworzony według formatu: [ ] oznaczają że element jest opcjonalny”
    - ○ %[+][‘znak_dopełnienia][-][szerokość][.dokładność]typ
        - Wszystkie specyfikatory konwersji zaczynają się od znaku % by wyświetlić procent (%) należy użyć %%
        - + jest opcjonalny, dla liczb<0 będzie tu dodany ‘-’ natomiast dla dodatnich nie będzie wyświetlany ‘+’. Gdy go użyjemy, i i + będą wyświetlane
        - ‘znak_dopełnienia - jest opcjonalny, będzie on rozszerzał zmienna do zadanej szerokości. Jeżeli chcemy użyć spacji lub zera nie musimy ich poprzedzać apostrofem, każdy inny znak należy nim poprzedzić
        - - jest opcjonalny, gdy go użyjemy dane zostaną wyrównane do lewej strony zamiast domyślnego wyrównania do prawej.
        - Szerokosc -opcjonalny, rezerwuje ilość miejsca (w znakach) które należy pozostawić dla danych.
        - .dokładność - należy poprzedzić znakiem . I podać ile miejsc po przecinku ma być widocznych
        - Używając specyfikatorów nie musimy zachowywać kolejności wymieniania w liście np.. %3\$.2f zwróci nam trzeci argument z listy w formacie liczby zmiennoprzecinkowej do 2giego jej miejsca. Zapis: %Które_miejsce_na_liście\$
        - Typy kodów specyfikatorów konwersji
            - % - znak %
            - b – zinterpretować jako int i wyświetlić jako liczbę binarną
            - c – zinterpretować jako int i wyświetlić jako znak
            - d – zinterpretować jako int i wyświetlić jako liczbę dziesiętną
            - e - zinterpretować jako double i wyświetlić w zapisie naukowym
            - E – jak e tyle że wyświetlona zostanie wielka litera E
            - f - zinterpretować jako float i wyświetlić z uwzględnieniem ustawień lokalnych
            - F - zinterpretować jako float i wyświetlić bez uwzględniania ustawień lokalnych
            - g - krótszy zapis dla e lub f
            - G - krótszy zapis dla E lub F
            - o – zinterpretować jako int i wyświetlić ósemkowo
            - s – zinterpretować i wyświetlić jaki string
            - u –zinterpretować jako int i wyświetlić jako liczbę dziesiętną bez znaku

- x – Zinterpretować jako int i wyświetlić w systemie szesnastkowym z małymi literami a-f
- X - Zinterpretować jako int i wyświetlić w systemie szesnastkowym z wielkimi literami A-F

★ **Zmiana wielkości liter w łańcuchu**
- strtoupper() - zmienia na wielkie litery
- strtolower() - zmienia na małe litery
- ucfirst() - zmienia na wielką pierwszą literę łańcucha o ile jest elementem alfabetu
- ucwords() - zmienia na wielkie litery wszystkie pierwsze wszystkich słów w ciągu o ile są one częścią alfabetu

9. Łączenie i rozdzielanie łańcuchów znakowych

★ explode(“separator”, “$tekst”,[int limit]) - funkcja pobiera ciąg znaków $tekst i dzieli według separatora tworząc tablicę składająca się z kolejnych odseparowanych ciągów znaków. Parametr limit jest opcjonalny i pozwala ograniczyć rozmiar tablicy np.
- Explode(“ “, “Hello world”) => $tablica[“Hello”,”world”]
- Explode(“ “, “Hello world kolejne słowa”, 2) => $tablica[“Hello”,”world”]

★ implode() / join() -maja funkcję odwrotną do explode pierwszy argument odpowiada za znak łączący tablicę w ciąg np.. :
- $tablica[“Hello”,”world”];
- implode(“ “, $tablica); => “Hello world”

★ strtok($wpis, “separator”) - funkcja pobiera z łańcucha części zwane żetonami, jej separatorem może być ciąg znaków jak i znak. Wyjściowy ciąg znaków zostanie użyty przy każdym z elementów łańcucha separacyjnego a nie przy całym łańcuchu jak to ma miejsce przy użyciu explode(). Aby otrzymać pierwszy żeton należy wywołać strok() z łańcuchem znaków który ma zostać rozbity oraz separatorem. W celu uzyskania kolejnych żetonów wystarczy podać już tylko separator. Funkcja utrzymuje swój wewnętrzny wskaźnik w odpowiednim miejscu łańcucha. Aby wyzerować wskaźnik należy podać funkcji łańcuch znaków do rozbicia. Używa się jej zazwyczaj w nastepujący sposób:

```php
$string = "To jest\tprzykładowy \nciag znaków";
/* użyjemy tabulacji i noiwego woersza do
dzielenia  */
$tok = strtok($string, " \n\t");

while ($tok !== false) {
    echo "Word=$tok<br />";
    $tok = strtok(" \n\t");
}
```

○

★ substr(“ciąg znaków”, int start, int [długość]) - pozwala na dostep do fragmentu łańcucha zawartego pomiędzy jego podanymi miejscami. Np..
- $ciag=”Wasza obsługa klientów jest wspaniała”
- substr($ciag,0)=> “Wasza obsługa klientów jest wspaniała”
- substr($ciag,-9)=> “wspaniała” - wartości ujemne powodują przejście na koniec łańcucha znaków.
- substr($ciag,0,5)=>"wasza”

10. **Porzadkowanie łąńcuchów znaków**
    - ★ strcmp("ciag1”, “ciag2”) - funkcja porównuje dwa ciągi znaków w porządku leksykograficznym zwracając uwagę na wielkość liter. Jeżeli są równe zwróci 0, Jeżeli ciag1 jest ustawiany za ciag2 lub jest większy zwróci liczbę większa od zera. Jeżeli jest mniejszy (przed) zwróci liczbę mniejszą od zera .
    - ★ strcasecmp() - jak strcmp() tyle ze nie bierze pod uwagę wielkości znaków
    - ★ strnatcmp() - jak strcmp() przy czym używa porządkowania naturalnego
    - ★ Strnatcasecmp() - jak strnatcmp() tyle że nie bierze pod uwagę wielkości liter
11. **Sprawdzanie długości ciągu znaków**
    - ★ strlen() - zwraca długość ciągu znaków w postaci int
12. **Dopasowywanie i zamiana łańcuchów znakowych za pomocą funkcji łańcuchowych**
    - ★ strstr(“igła w stogu siana”, “igła”) / strchr() - pierwszy parametr to ciąg przeszukiwany natomiast drugi to szukany ciąg znaków zwracansa jest wartość boolowska.
    - ★ stristr() - jak strstr() lecz nie zwraca uwagi na wielkosć liter

Ćwiczneie 3 – w naszym formularzu utwórz kod odpowiadający za wysyłanie wiadomości e –mail na wskazany adres zależnie od występujących w treści komentarza słów. Tzn. Jeżeli komentarz zawiera słowo lub jego synonimy i odmiany:
“dostawa” => dostawy@przykladowy.pl
“sklep” => sprzedaz@przykladowy.pl
“rachunek”=> ksiegowosc@przykladowy.pl
W pozostałych wypadkach mail ma być wysyłany na domyślny adres.