

PHP - Operacje na plikach

Tablice asocjacyjne

Beata Pańczyk - PHP (Wykład 2)

Plan wykładu

- Tablice asocjacyjne
- Praca z tablicami
- Podstawowe operacje na plikach
- Funkcje plikowe
- Operacje na ciągach

Tablice indeksowane numerycznie

- indeksowane numerycznie (od 0) inicjalizowanie tablicy np.:

```
$towar=array("Zeszyt","Blok","Kredki");
```

lub krócej:

```
$towar=["Zeszyt","Blok","Kredki"];
```

- dostęp do elementów tablicy np.:

```
$towar[0]="Farby"; //jeśli tablica nie istniała
```

//to zostanie utworzona

```
$towar[1]="Zeszyt";//dodanie kolejnego elementu tablicy
```

```
echo "$towar[0] $towar[1] ";
```

```
for ($i=0;$i<2;$i++) echo "$towar[$i]";
```

Tablice asocjacyjne

- inicjalizowanie np.
`$sceny=array("Zeszyt"=>2, "Blok"=>8, "Kredki"=>4);`
- dostęp do elementów tablicy np.:
`$sceny["Blok"]=6;`
- tworzenie tablicy:
`$sceny["Zeszyt"]=2;`
`$sceny["Blok"]=6;`
- indeksy tablicy nie są liczbami (nie można stosować zwykłego licznika w pętli), stosuje się funkcje
each() - zwraca bieżący element tablicy oraz nadaje następnemu atrybut bieżącego
list() - rozdziela tablicę na kilka wartości

Konstrukcja foreach

- iteracyjne przetwarzanie tablic asocjacyjnych

```
foreach ($tab as $klucz=>$wartosc) {  
    print ( ' $tablica['.$klucz.']='.$wartosc. '<br /> ');  
}
```

pętla foreach operuje na kopii zmiennej \$tablica
zastosowanie ' zamiast " pozwala wyświetlić
nazwę tablicy zamiast pobierać jej wartość
- modyfikacja zawartości tablicy wewnątrz pętli
(poprzez operator referencji &):

```
foreach ($tab as $klucz=>&$wartosc) {  
    //modyfikacja wartości  
}
```

Przykład 2a - funkcja each()

```
<?
$sceny=array("Zeszyt"=>2, "Blok"=>8, "Kredki"=>4);
while ($element=each($sceny)) // each() - wycofana w wersji PHP 7.2
{ echo $element["key"];
  echo " ";
  echo $element["value"];
  echo "<br />";
} ?>
```

```
Zeszyt 2
Blok 8
Kredki 4
```

Dla bieżącego elementu:

- Pozycje key i 0 (klucz)
- Pozycje value i 1 (wartość)

Przykład 2b- funkcja list()

```
<?php
$sceny=array("Zeszyt"=>2, "Blok"=>8, "Kredki"=>4);
while (list($towar,$cena)=each($sceny))
    echo "$towar $cena<br />";
    //lub: echo $towar."-".$cena."<br />";
?>
```

Efekt działania - jak w przykładzie 1

Przykład 2c- pętla foreach

```
<?php
$sceny=array("Zeszyt"=>2, "Blok"=>8, "Kredki"=>4);
foreach ($sceny as $klucz=>$wartosc)
{
    print (' $sceny['.$klucz.']='
        .$wartosc. '<br /> ');
}
?>
```

```
$sceny[Zeszyt]=2
$sceny[Blok]=8
$sceny[Kredki]=4
```


Tablice wielowymiarowe - przykład

```
//tablica numeryczna
```

```
//*****
```

```
$towar = array( array( "Kredki", 10), array( "Zeszyt", 2 ) );  
echo $towar[0][0]." ".$towar[0][1]."<br />";  
echo $towar[1][0]." ".$towar[1][1]."<br />";
```

```
//tablica asocjacyjna
```

```
//*****
```

```
$towar1 = array( array( "Nazwa"=>"Kredki", "Cena"=>10),  
                 array( "Nazwa"=>"Zeszyt", "Cena"=> 2 ) );  
for ( $i=0; $i<2; $i++)  
    echo $towar1[$i]["Nazwa"]." ".$towar1[$i]["Cena"]."<br />";
```

Sortowanie tablic

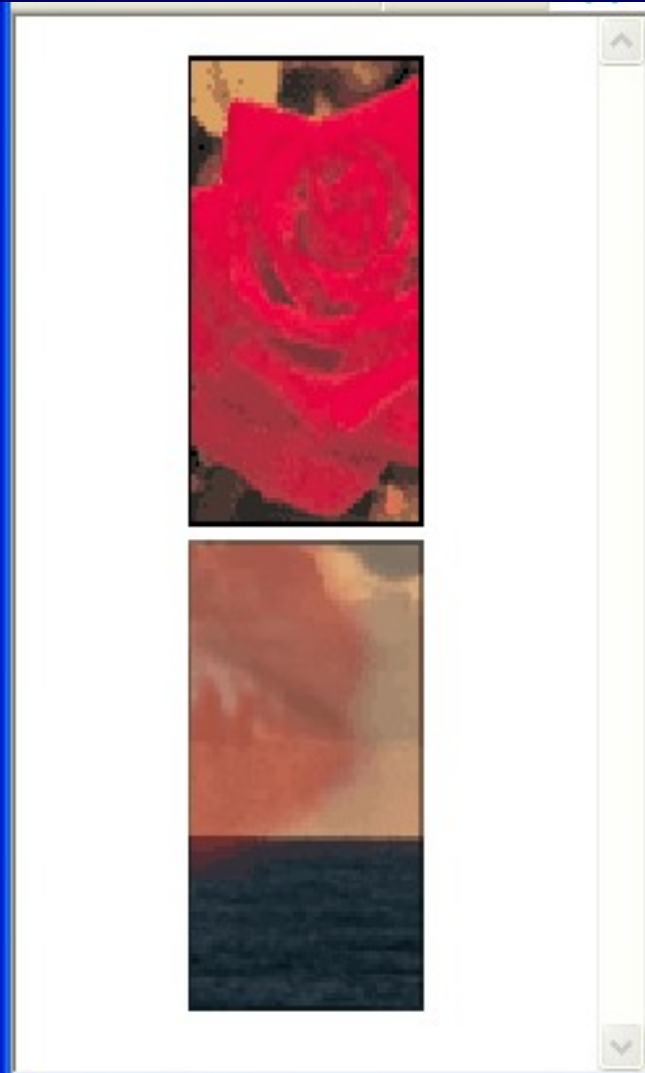
- tablice numeryczne:
sort() - sortowanie rosnące
rsort() - sortowanie malejące
np.
`$ciag=array(20,10,18); sort($ciag);`
- tablice asocjacyjne
asort() - sortowanie rosnące wg. wartości elementu
ksort() - sortowanie rosnące wg. klucza
arsort(), krsort() - sortowanie malejące
np.
`$sceny=array("Zeszyt"=>2, "Blok"=>8, "Kredki"=>4);
asort($sceny); //$sceny=array("Zeszyt"=>2, "Kredki"=>4, "Blok"=>8);
ksort($sceny); //$sceny=array("Blok"=>8, "Kredki"=>4, "Zeszyt"=>2);`

Zmiana kolejności elementów tablicy

- **shuffle()** - ustawia losowo kolejność elementów tablicy
- **array_reverse()** - pobiera tablice i tworzy nową o tych samych elementach umieszczonych w odwrotnej kolejności
np. utworzenie tablicy liczb od 10 do 1:
 - a) `$liczby=range(1,10); //tworzy sekwencję rosnącą`
`$liczby=array_reverse($liczby);`
 - b) `$liczby=array(); //utworzenie pustej tablicy`
`for ($i=10; $i>0; $i--)`
`array_push($liczby,$i); //dodanie elementu na koniec`

Przykład 3 - funkcja shuffle()

```
<?
$obrazki= array ( "muszla.gif",
    "roza.gif", "jesien.gif", "zachod.
    "gory.gif", "smolinos.gif" );
shuffle($obrazki);
for ( $i=0; $i<2; $i++)
{
    echo ' <br />';
}
?>
```



Otwieranie pliku - fopen

`$wp=fopen("ścieżka/nazwa_pliku","tryb",1);`//\$wp- wskaźnik pliku

- np.

`$wp=fopen("$doc_root/../dane/dane.txt","w");`

`$doc_root`- zmienna PHP wskazuje na podstawowy element drzewa katalogów serwera WWW (ustawiana w konfiguracji **php.ini**);

`".."` - katalog nadrzędny katalogu `$doc_root` (dla bezpieczeństwa poza drzewem katalogów); **ścieżka względna** (opisuje miejsce w systemie plików w zależności od `$doc_root`); można stosować ścieżkę bezwzględną -nie polecane ze względów bezpieczeństwa

- np. `$wp=fopen("dane.txt","a",1);`

trzeci parametr (opcjonalny) - poszukiwanie pliku w lokalizacjach podanych w opcji `include_path` (**php.ini**); nie trzeba podawać ścieżki dostępu do pliku

Tryby otwarcia pliku

- **r** - odczyt (od początku pliku)
- **r+** - odczyt i zapis (od początku pliku)
- **w** - zapis (od początku pliku); jeśli plik nie istnieje - próba jego utworzenia, jeśli istnieje - bieżąca zawartość zostanie skasowana
- **w+** - zapis i odczyt (od początku pliku); (j.w.)
- **a** - dodawanie zawartości (od końca istniejącej zawartości pliku); jeśli plik nie istnieje - próba jego utworzenia
- **a+** - dodawanie zawartości i odczyt (od końca istniejącej zawartości pliku); jeśli plik nie istnieje - próba jego utworzenia
- **b** - tryb binarny (w połączeniu z powyższymi jeśli system rozróżnia pliki tekstowe i binarne)

Poprawność otwarcia pliku

- `fopen()`
 - zwraca wartość wskaźnika pliku (zmienna np. `$wp`) jeśli operacja otwarcia pliku zakończona sukcesem
 - zwraca wartość `false` w przeciwnym razie
- np.:

```
@ $wp=fopen("dane.txt","a",1);
if (!$wp)
{
    echo "<p>Zamowienie nie może zostać przyjęte.
        Spróbuj później</p></body></html>";
    exit;
}
//wykonaj operacje na pliku i zamknij plik
fclose($wp);
```

Zapisywanie w pliku - fwrite, fputs

- **int fwrite (\$wp , \$łańcuch [, \$długość])**
 - **dlugosc** - parametr opcjonalny (max do zapisania ilość bajtów;
 - **wp** – wskaźnik pliku do zapisu
 - **łańcuch** – tekst do zapisu
 - funkcja zapisuje do pliku, dopóki nie osiągnie końca ciagu lub zapisze dlugosc bajtów (zależnie od tego co wystąpi wcześniej
- fputs – alias fwrite
- np. \$ciag=\$imie."\t".\$nazwisko."\t".\$data."\n";
fwrite(\$wp,\$ciag);

Odczyt z pliku - linia po linii

- `$z=fgets($wp, 100); /*czyta linię z pliku dopóki nie natrafi na \n, EOF, lub przeczyta 99 bajtów */`
- `string fgetss(int wk, int dlugosc, string [dozw_znaczni]); /*podobnie jak fgets, ale z czytanego ciągu usuwa wszystkie znaczniki PHP i HTML poza wyszczególnionymi*/`
- `array fgetcsv(int wk, int dlugosc, string [znak_podziału]); /* podobnie do fgets() tylko, że przetwarza odczytaną linię na pola i zwraca tablicę zawierającą odczytane pola. */`

Odczyt całego pliku

- **int readfile(string nazwa_pliku, int [include_path]);**
/* otwiera plik, wyświetla zawartość w okienku przeglądarki i zamyka plik; drugi parametr opcjonalny określa, czy PHP powinien szukać pliku przez opcję include_path (tak jak w fopen)
- po otwarciu pliku i przekazaniu wartości wskaźnika pliku do funkcji **fpassthru()**, wyświetla ona zawartość tego pliku w okienku przeglądarki i po zakończeniu działania zamyka plik. Funkcja zwraca wartość **true**, jeżeli odczyt powiedzie się lub **false** gdy się nie uda;
np. **\$wp=fopen("dane.txt","r");**
 fpassthru(\$wp);
- **\$tablica_pliku=file(\$wp);** /*działa jak readfile, ale zamiast wyświetlać zawartość pliku w przeglądarce zamienia ją na tablicę - każda linia pliku jako osobny element tablicy */

Odczyt z pliku

- `string fread(int wp, int dlugosc);` /*odczytanie zadanej liczby bajtów, chyba, że wcześniej napotkany zostanie znak końca pliku*/
- np.:

```
<?php
// pobierz zawartość pliku do łańcucha
$nazwa_pliku = "/local/jakis.txt";
$uchwyt = fopen($nazwa_pliku, "r");
$tresc = fread($uchwyt, filesize($nazwa_pliku));
fclose($uchwyt);
?>
```

Ustawianie wskaźnika w pliku

- `rewind()` - ustawienie wskaźnika na początku pliku
- `ftell()` - podaje jak daleko (w bajtach) został przesunięty wskaźnik
np.
`rewind($wp);`
`echo "Pozycja wskaźnika pliku wynosi:". (ftell($wp));`
- `int fseek(int wp,int b);` - ustawia wskaźnik pliku w punkcie `b` bajtów, licząc od początku pliku

Inne funkcje plikowe

- `file_exists()` - sprawdzenie istnienia pliku bez otwierania np.
`if (file_exists("plik")) echo "komunikat 1";`
`else echo "komunikat 2";`
- `filesize()` - określenie wielkości pliku w bajtach
np.
`$wp=fopen("plik","r");`
`echo fread($wp,filesize("plik"));`
`fclose($wp);`

Blokowanie pliku - flock

- `bool flock(int wp, int blokada); /*zwraca true, jeśli blokada jest prawidłowa*/`

Blokada:

- **LOCK_SH** (1)- pozwala na dzielenie pliku z innymi czytającymi (shared lock)
- **LOCK_EX** (2) - wyłącza plik z użytku - nie może być dzielony (exclusive lock)
- **LOCK_UN** (3) - zwolnienie istniejącej blokady (release a lock)

Blokowanie pliku

flock() należy dodać do wszystkich skryptów korzystających z pliku

np.

```
$wp=fopen("plik","a");  
//blokada zapisu:  
flock( $wp, LOCK_EX);  
fwrite($wp,"ciag");  
//zwolnienie blokady zapisu:  
flock( $wp, LOCK_UN);  
fclose( $wp );
```

Przykład 1 - Licznik odwiedzin

```
<?php // w pliku httpd.conf:
      // DocumentRoot "C:/xampp/htdocs"
      // Plik tekstowy ma być umieszczony poza katalogiem
      htdocs
      // w folderze: C:/xampp/Mojepliki
$d_root = $_SERVER['DOCUMENT_ROOT'];
if (!(file_exists("$d_root/../Mojepliki/liczba.txt")))
{
    $plik=fopen("$d_root/../Mojepliki/liczba.txt","w+");
    fputs($plik,"0"); fclose($plik);
}
$plik=fopen("$d_root/../Mojepliki/liczba.txt", "r+");
if (!$plik) { echo "Nie da się otworzyć pliku."; }
flock($plik, LOCK_EX);
$file=fgets($plik,255);
$file++;
print "Licznik wskazuje: $file";
fseek($plik,0);
fputs($plik,$file);
flock($plik,LOCK_UN);
fclose($plik);      ?>
```

Licznik wskazuje: 3

Przykład 4a - Ładowanie tablic z plików

```
<?
$osoby=file("osoby.txt"); //załadowanie pliku do tablicy
                                //każda linia pliku to jeden element tablicy
$ile=count($osoby);           //określenie liczby elementów tablicy
for ( $i=0; $i<$ile; $i++)
{
    echo $osoby[$i];
    echo "<br /> ";
}
?>
```

```
1 Ania Kowalska
2 Magdalena Nowak
3 Jan Abacki
4 Adam Babacki
```

Przykład 4b - formatowanie danych z pliku

```
<?php
$osoby=file("osoby.txt");
$ile=count($osoby);
echo "<table border='1'><tbody>";
echo "<tr><th> ID</th> ";
echo "<th>Imię</th>";
echo "<th> Nazwisko</th></tr>";
for ( $i=0; $i<$ile; $i++)
{
    $linia = explode(" ",$osoby[$i]); //rozbicie każdej linii
    $id = intval($linia[0]);    //konwersja tekstu na wartość całkowitą
    echo "<tr><td>$id</td><td>$linia[1]</td><td>$linia[2]</td></tr>";
}
echo "</tbody></table>" ?>
```

ID	Imię	Nazwisko
1	Ania	Kowalska
2	Magdalena	Nowak
3	Jan	Abacki
4	Adam	Babacki

Poruszanie się wewnątrz tablicy asocjacyjnej

- Każda tablica asocjacyjna posiada wewnętrzny wskaźnik pokazujący aktualny element tablicy.
- Funkcje ustawiające wskaźnik:
 - **current(\$tab)** - pierwszy element
 - **each(\$tab)** - zwraca aktualny element i przeskakuje do kolejnego
 - **next(\$tab)** - przesuwa wskaźnik do następnego elementu i zwraca nowy element aktualny
 - **prev(\$tab)** - przesuwa wskaźnik o jeden element wstecz i zwraca nowy element aktualny
 - **reset(\$tab)** - pierwszy element
 - **end(\$tab)** - ostatni element

Zliczanie elementów tablicy

- **count(\$tab), sizeof(\$tab)** - wyznaczają ilość elementów tablicy; wynik przekazywany w formie zmiennej skalarnej; jeśli argumentem jest tablica pusta lub nie jest ustawiona zmienna o podanej nazwie - wynikiem jest 0
- **array_count_values(\$tab)** - oblicza ile niepowtarzalnych wartości występuje w tablicy; zwraca tablicę asocjacyjną zawierającą tabelę częstości (kluczami są wszystkie pojedyncze wartości, każdy klucz posiada wartość numeryczną zawierającą liczbę powtórzeń konkretnej wartości).

Przykład 5 - array_count_values()

```
<?
$a=array(4,5,2,4,4,5);
$licz=array_count_values($a);
echo "<table border='1'><tbody>";
echo "<tr><th>Klucz</th>";
echo "<th>Liczba powtórzeń</th></tr>";
while ($element=each($licz))
{
    echo "<tr><td>".$element["key"]."</td>";
    echo "<td>".$element["value"]."</td></tr>";
}
echo "</tbody></table>";
?>
```

Klucz	Liczba powtórzeń
4	3
5	2
2	1

Konwersja tablic na zmienne skalarne

- **extract()** pobiera tablicę asocjacyjną i tworzy wartości skalarne o nazwach takich jak klucze tablicy;
prototyp funkcji:
`extract (array tab[, int typ_ekstrakcji][, string przedrostek]);`
- np.
`$tab=array("klucz1"=>"wartosc1", "klucz2"=>"wartosc2");`
`extract($tab); //utworzone zostają zmienne $klucz1, $klucz2`
`//o wartościach odpowiednio wartosc1, wartosc2`
`echo "$klucz1 $klucz2";//wypisane będzie: wartosc1 wartosc2`

Parametry funkcji extract()

- przedrostek - nazwa tworzonej zmiennej będzie postaci:
przedrostek_klucz
- Typ ekstrakcji (sposób postępowania w przypadku kolizji nazw):
 - EXTR_OVERWRITE** - nadpisanie istniejącej zmiennej (domyślnie)
 - EXTR_SKIP** - ominięcie elementu
 - EXTR_PREFIX_SAME** - utworzenie zmiennej przedrostek_klucz (musi być podany przedrostek)
 - EXTR_PREFIX_ALL** - przedrostek umieszczany przed wszystkimi nazwami zmiennych
- np.

```
$tab=array("klucz1"=>"wartosc1", "klucz2"=>"wartosc2");  
extract($tab, EXTR_PREFIX_ALL,"bp");  
echo "$bp_klucz1 $bp_klucz2";
```

Przykład 6 - ankieta

```
<form method="post" action="dopisz.php">
```

```
  Czy podoba ci się nasz serwis? <br />
```

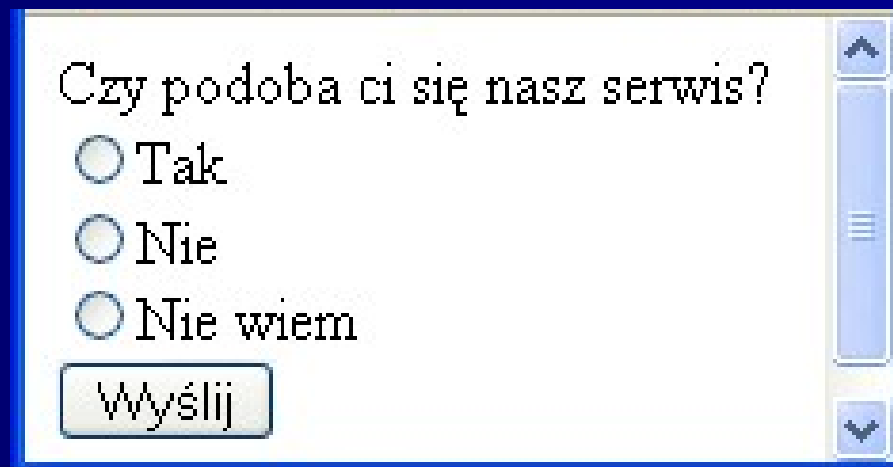
```
  <input type="radio" name="odp"      value="tak"/> Tak <br />
```

```
  <input type="radio" name="odp"      value="nie"/> Nie <br />
```

```
  <input type="radio" name="odp"      value="nw"/> Nie wiem <br />
```

```
  <input type="submit" value="Wyślij"/>
```

```
</form>
```



Przykład 6 - plik dopisz.php

```
<?php
    if (!(file_exists("sonda.txt"))) {        $plik=fopen("sonda.txt","w+");
                                                fputs($plik,"0#0#0"); fclose($plik); }

    $plik=fopen("sonda.txt", "r+") or exit ("Nie da się otworzyć pliku.");
    $linia = fgets($plik, 80);
    $tab = explode("#",$linia);                //rozbicie ciągu $linia na elementy tablicy
    $tbl["tak"] = $tab[0];                      // tworzenie tablicy asocjacyjnej
    $tbl["nie"] = $tab[1];
    $tbl["nw"] = $tab[2];
    ++$tbl[$_POST['odp']]; //zwiększenie ilości głosów na daną odpowiedź
    fseek($plik, 0);        //przesunięcie wskaźnika pliku do początku
    fwrite($plik, $tbl["tak"]."#".$tbl["nie"]."#".$tbl["nw"]); //zapis do pliku
    header("Location: sonda1.php"); //ustawienie nagłówka HTTP
```

3#2#2|

Przykład 6 - plik sonda1.php

```
<body>
<table border="1"><tbody>
<? $file = fopen("sonda.txt","r+"); //otwarcie pliku
    $linia = fgets($file, 80);
    $tab = explode("#",$linia); //zapis linii do tablicy
    $razem=$tab[0]+$tab[1]+$tab[2];
    echo "<tr><td>Tak</td><td>".$tab[0]."</td></tr>";
    echo "<tr><td>Nie</td><td>".$tab[1]."</td></tr>";
    echo "<tr><td>Nie wiem</td><td>".$tab[2]."</td></tr>";
    echo "<tr><td>Łącznie głosów:</td><td>".$razem."</td></tr>";
?> </tbody></table>
<br /><a href="sonda.php"> Powrót do formularza sondy</a>
</body>
```

Tak	3
Nie	2
Nie wiem	2
Łącznie głosów:	7

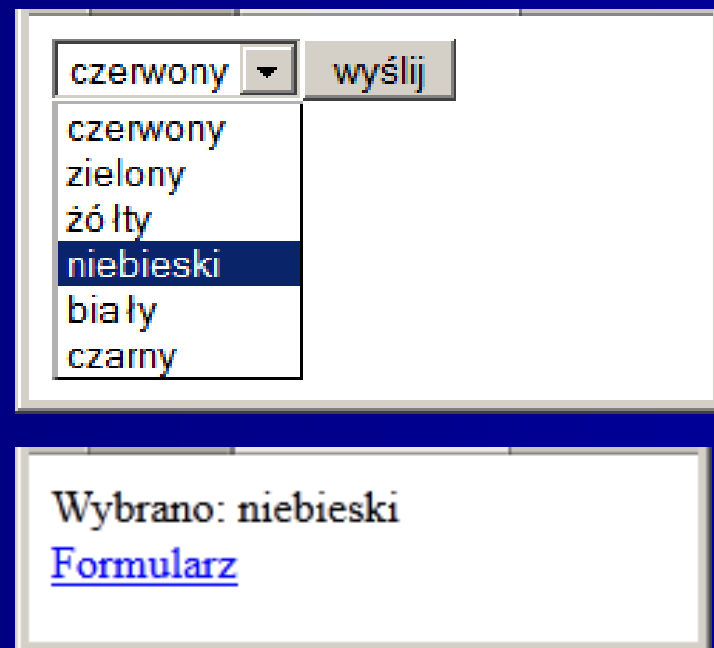
[Powrót do formularza
sondy](#)

Przykład 7 – obsługa listy

```
<?php
    $t=array("czer"=>"czerwony", "ziel"=>"zielony", "zol"=>"żółty",
    "nieb"=>"niebieski", "cialy"=>"biały", "czarny"=>"czarny");
    if (!isset($_POST['kolor']))
    {
        echo '<form action="lista.php" method="POST">';
        echo '<select id="" name="kolor">';
        foreach ($t as $klucz=>$wart)
        {
            echo "<option value='$klucz'>$wart</option>";
        }
        echo "</select>";
        echo '<input type="submit" value="wyślij" />';
        echo "</form>";
    }
```

Przykład 7 c.d.

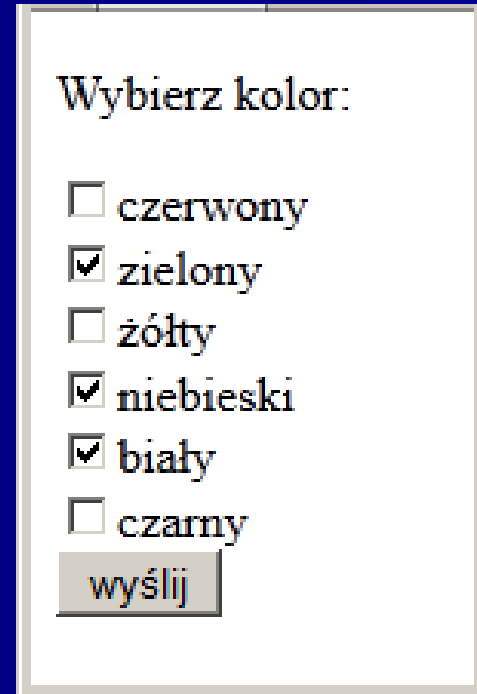
```
else
{
    $wybrano="nic";
    foreach ($t as $klucz=>$wart)
    {
        if ($_POST['kolor']==$klucz)
            $wybrano=$wart;
    }
    echo "Wybrano: $wybrano </br>";
    echo "<a href='lista.php'>
        Formularz</a>";
}
?>
```



The image shows a web form and its output. The form has a dropdown menu with the following options: czerwony, zielony, żółty, niebieski (selected), biały, and czarny. There is a 'wyślij' button next to the dropdown. Below the form, the output is displayed: 'Wybrano: niebieski' followed by a blue underlined link 'Formularz'.

Przykład 8 – pola wyboru

```
<?php
    $t=array("czer"=>"czerwony",
        "ziel"=>"zielony","zol"=>"żółty",
        "nieb"=>"niebieski",
        "bialy"=>"biały",
        "czarny"=>"czarny");
    $zaznaczono=0;
    $wybrano="";
    //sprawdzenie czy dokonano wyboru
    foreach ($t as $klucz=>$wart)
        if (isset($_GET[$klucz]))
        { $wybrano.=$wart." ";
          $zaznaczono=1;
        }
}
```



Wybierz kolor:

- ☐ czerwony
- ☒ zielony
- ☐ żółty
- ☒ niebieski
- ☒ biały
- ☐ czarny

Przykład 8 c.d.

```
if (!$zaznaczono)
{ echo '<form action="wybor.php" method="GET">';
  echo "<p>Wybierz kolor:</p>";
  foreach ($t as $klucz=>$wart)
    echo "<input type='checkbox' name='$klucz' value='1'>$wart<br/>";
  echo '<input type="submit" value="wyślij" /></form>'; }
else
{ echo "Wybrano: $wybrano<br/>";
  echo "<a href='wybor.php'>Formularz</a>";
}
?>
```

Wybrano: zielony niebieski biały
[Formularz](#)

Łączenie i rozdzielanie ciągów

- **array explode(string separator, string tekst)** - pobiera ciąg o nazwie tekst i rozбивa go na części według separatora
np. \$tab=explode("#",\$tekst);
- **implode(), join()** - efekt odwrotny do działania **explode()**,
np. \$tekst=implode("#",\$tab);
- **string strtok(string tekst, string separator)** - pobiera z ciągu (po jednym) fragmenty (żetony) oddzielone separatorem
- **string substr(string ciag, int start, int [dlugosc])** - zwraca podciąg skopiowany z ciągu np. \$tekst="Laboratorium z języka PHP";
substr(\$tekst, 3); //zwraca "oratorium z języka PHP"
substr(\$tekst,-3); //zwraca "PHP"
substr(\$tekst,0,12); //zwraca "Laboratorium "
substr(\$tekst,12,-3); //zwraca "z języka"

Porównywanie ciągów

- operator ==
- **int strcmp(string ciag1, string ciag2);** - jeżeli ciągi są równe funkcja zwraca 0, jeżeli ciag1 >= ciag 2 (w porządku leksykograficznym) funkcja zwraca liczbę >0, w przeciwnym razie zwraca liczbę < 0; uwzględniana jest wielkość liter
- **strcasecmp()** - j.w. tylko nie uwzględniane są wielkości liter
- **strlen()** - wyznacza długość ciągu:
np.
`$ciag="PHP";`
`strlen($ciag);`

Znajdowanie podciągów w ciągach

- **strstr(), strchr ()** - odnajdywanie tekstu lub znaku w ciągu z uwzględnieniem wielkości liter
prototyp:
string strstr(string ciag, string tekst);
jeżeli zostanie odnaleziony podciąg dokładnie pasujący do szukanego tekstu to funkcja zwraca fragment ciągu rozpoczynający się od pierwszego wystąpienia szukanego tekstu, w przeciwnym przypadku false;
- **stristr()** - j. w. bez uwzględniania wielkości liter
- **strrchr()** - j.w. lecz zwraca fragment ciągu rozpoczynający się od ostatniego wystąpienia szukanego tekstu

Odnajdowanie pozycji podciągu

- **strpos()** - działa podobnie jak strstr(), lecz zwraca numeryczną pozycję podciągu
prototyp:
int strpos(string ciag, string tekst, int n);
funkcja zwraca pozycję pierwszego wystąpienia **tekstu** w **ciagu** lub false; parametr **n** podaje pozycję, począwszy od której **ciag** ma być przeszukiwany,
np. \$ciag="Cześć świecie";
 echo strpos(\$ciag,"ś"); //wyświetli 3
 echo strpos(\$ciag,"ś",4); //wyświetli 6
- **strspos()** - j.w. lecz zwraca pozycję ostatniego wystąpienia **tekstu** w **ciagu**; inaczej niż strpos(), wykorzystuje jedynie pierwszy znak przekazanego **tekstu**

Zamiana podciągów

- **string str_replace(string tekst, string nowy_tekst, string ciag);** - zamienia wszystkie znalezione fragmenty **tekst** na **nowy_tekst** w **ciagu**
- **string substr_replace(string ciag, string zamiana, int start, int [dlugosc]);** - zamienia część ciągu **ciag** na ciąg **zamiana**, część modyfikowana zależy od parametru **start** (jeśli $\text{start} \geq 0$ pozycja zamiany liczona od początku ciągu, jeśli < 0 - od końca) i opcjonalnego parametru **dlugosc** (punkt zakończenia zamiany, jeśli nie jest podany - ciąg zostanie zamieniony od pozycji **start** do końca ciągu, jeśli jest równy 0 to zamiana zostanie wstawiona do ciągu bez nadpisania istniejącej zawartości, ujemna wartość parametru opisuje pozycję liczoną od końca ciągu, w której powinna zakończyć się zamiana)

Przykład 9 - operacje na ciągach

```
<?php
$ciag="Systemy informatyczne w sieciach rozległych";
echo "Ciag: $ciag <br/>";
echo "Długość ciągu: ".strlen($ciag)."<BR>";
echo "Tekst wyszukany: ".strstr($ciag,"informatyczne")."<br/>";
echo "Początkowa pozycja wyszukanego tekstu: "
        .strpos($ciag,"informatyczne")."<br/>";
echo "Po 1 zamianie: ".substr_replace($ciag," PHP i MySQL",-20)."<br/>";
$ciag=substr_replace($ciag," PHP i MySQL - ",0,0);
echo "Po 2 zamianie: $ciag <br/>";
echo "Po 3 zamianie: ".substr_replace($ciag,"s",15,1)."<br/>";
echo "Po 4 zamianie: ".substr_replace($ciag,"s",15);
?>
```

Przykład 9 - wynik

Ciąg: Systemy informatyczne w sieciach rozległych

Długość ciągu: 43

Tekst wyszukany: informatyczne w sieciach rozległych

Początkowa pozycja wyszukanego tekstu: 8

Po 1 zamianie: Systemy informatyczne w PHP i MySQL

Po 2 zamianie: PHP i MySQL - Systemy informatyczne w sieciach rozległych

Po 3 zamianie: PHP i MySQL - systemy informatyczne w sieciach rozległych

Po 4 zamianie: PHP i MySQL - s