PHP i MySQL. Dla każdego

Wydanie II

Dołącz do grona twórców dynamicznych serwisów WWW — poznaj najbardziej popularne technologie w branży!

- Opanuj zasady programowania w języku PHP
- Poznaj sposoby korzystania z bazy MySQL
- Naucz się łączyć możliwości tych technologii
- Dowiedz się, jak za ich pomocą tworzyć praktyczne rozwiązania

PHP i MySQL

Dla każdego

Marcin Lis

Helion

Redaktor prowadzący: Ewelina Burska
Projekt okładki: Maciej Pasek
Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie?phmdk2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Materiały do książki można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/phmdk2.zip

ISBN: 978-83-246-4797-2

Printed in Poland.

Spis treści

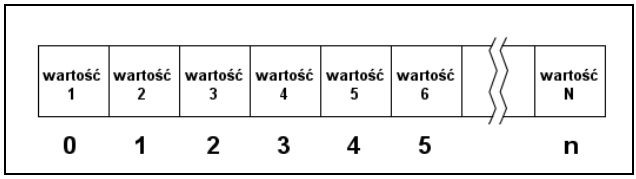
Rozdział 4.

Tablice

Rodzaje tablic w PHP

Tablice to występujące w większości języków programowania struktury, pozwalające na przechowywanie zbioru danych określonego typu. Tablicę można sobie wyobrazić jako wektor elementów, taki jak zaprezentowany na rysunku 4.1. Zawartością pojedynczej komórki tablicy może być wartość dowolnego typu danych (inaczej niż w klasycznych językach programowania). W PHP dostępne są dwa rodzaje tablic: klasyczne (indeksowane numerycznie) oraz asocjacyjne. Dostęp do poszczególnych danych zawartych w tablicy uzyskuje się poprzez podanie indeksu (inaczej klucza), pod którym dana wartość została zapisana.

Rysunek 4.1. *Struktura typowej tablicy*



Tablice zwykłe

Aby utworzyć prostą tablicę indeksowaną numerycznie, należy użyć słowa kluczowego `array` w schematycznej postaci:

```
$tablica = array(wartość1, wartość2,..., wartośćN);
```

gdzie: *tablica* to nazwa zmiennej tablicowej, dzięki której będzie można się do tej tablicy odwoływać, natomiast *wartość1*, *wartość2* itd. to wartości kolejnych komórek. W przypadku dużej liczby wartości w celu zwiększenia czytelności można również zastosować zapis w postaci:

```
$tablica = array
(
  wartość1,
  wartość2,
```

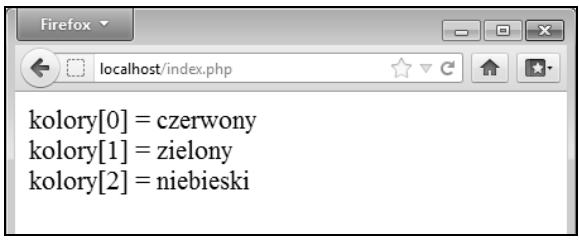
```
  ...,
  wartośćN
);
```

Zobaczmy, jak to będzie wyglądać w praktyce. Zobrazowano to w skrypcie widocznym na listingu 4.1.

Listing 4.1. *Deklaracja prostej tablicy*

```
<?php
$kolory = array("czerwony", "zielony", "niebieski");
echo "kolory[0] = $kolory[0] <br />";
echo "kolory[1] = $kolory[1] <br />";
echo "kolory[2] = $kolory[2] <br />";
?>
```

Powstała tutaj tablica `$kolory`, której kolejnym komórkom zostały przypisane ciągi znaków określające kolory: `czerwony`, `zielony` i `niebieski`. Aby uzyskać dostęp do wartości zapisanej w danej komórce, należy podać jej numer (indeks) w nawiasie kwadratowym występującym za nazwą tablicy. Należy przy tym pamiętać, że indeksowanie tablicy zaczyna się od zera, co oznacza, iż indeksem pierwszej komórki jest `0`, a NIE `1`. Aby zatem odczytać zawartość pierwszej komórki, piszemy `$kolory[0]`, drugiej komórki — `$kolory[1]`, a trzeciej komórki — `$kolory[2]`. Dzięki temu po uruchomieniu skryptu na ekranie ukaże się widok przedstawiony na rysunku 4.2.



Rysunek 4.2. *Wyświetlenie zawartości tablicy kolory*

Do odczytu zawartości tablicy można wykorzystać również pętle. Są one przydatne w szczególności wtedy, gdy tablica ma duże rozmiary. Na listingu 4.2 został przedstawiony skrypt realizujący takie samo zadanie jak skrypt z listingu 4.1 (czyli utworzenie tablicy i wyświetlenie jej zawartości), który wykorzystuje jednak pętlę typu `for`.

Listing 4.2. *Wykorzystanie pętli for do wyświetlenia zawartości tablicy*

```
<?php
$kolory = array("czerwony", "zielony", "niebieski");
for($i = 0; $i < 3; $i++){
  echo "kolory[$i] = $kolory[$i] <br />";
}
?>
```

Tablica może zostać również utworzona poprzez bezpośrednie przypisywanie wartości jej komórkom. Przykładowo zamiast pisać:

```
$kolory = array("czerwony", "zielony", "niebieski");
```

można wykorzystać serię instrukcji w postaci:

```
$kolory[0] = "czerwony";
$kolory[1] = "zielony";
$kolory[2] = "niebieski";
```

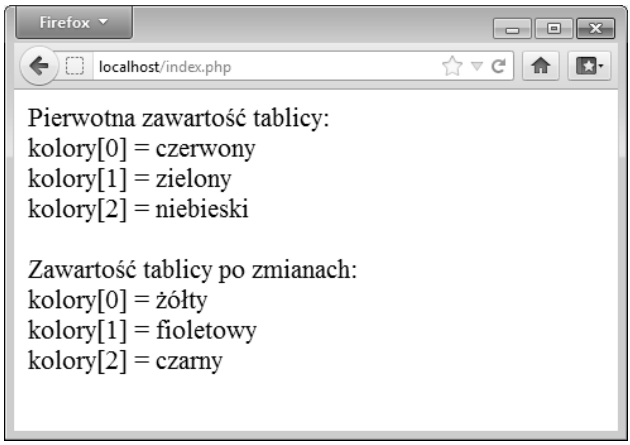
W ten sam sposób można również zmieniać zawartość poszczególnych komórek. Taką technikę zobrazowano w skrypcie przedstawionym na listingu 4.3.

Listing 4.3. *Modyfikacja komórek tablicy*

```
<?php
$kolory[0] = "czerwony";
$kolory[1] = "zielony";
$kolory[2] = "niebieski";

echo "Pierwotna zawartość tablicy: <br />";
for($i = 0; $i < 3; $i++){
  $kolor = $kolory[$i];
  echo "kolory[$i] = $kolor <br />";
}

$kolory[0] = "żółty";
$kolory[1] = "fioletowy";
$kolory[2] = "czarny";

echo "<br />Zawartość tablicy po zmianach: <br />";
for($i = 0; $i < 3; $i++){
  $kolor = $kolory[$i];
  echo "kolory[$i] = $kolor <br />";
}
?>
```

W pierwszej części skryptu powstała tablica `kolory`, której kolejnym indeksom przypisane zostały wartości `czerwony`, `zielony` i `niebieski`. Ściślej rzecz ujmując, tablica powstała po wykonaniu instrukcji `$kolory[0] = "czerwony";`. Po napotkaniu tej instrukcji i stwierdzeniu, że w skrypcie nie ma tablicy o nazwie `$kolory`, PHP tworzy ją, a następnie komórce o indeksie 0 przypisuje wartość z prawej strony operatora przypisania `=`. Następne dwie instrukcje to nic innego jak utworzenie kolejnych dwóch komórek w tablicy `$kolory` i przypisanie im wskazanych wartości. Po przeprowadzeniu wymienionych operacji wykonywana jest pętla `for`, która wyświetla zawartość całej tablicy w przeglądarce.

Za pętlą znajduje się instrukcja `$kolory[0] = "żółty";`. Ponieważ istnieje już tablica `$kolory`, instrukcja ta powoduje przypisanie komórce o indeksie `0` ciągu znaków `żółty`. W tym momencie zostaje również utracona poprzednia zawartość tej komórki, czyli ciąg `czerwony`. Podobnie działają dwie kolejne instrukcje. Zamieniają występujące w tablicy wartości z komórek `1` i `2` na ciągi znaków `fioletowy` i `czarny`. Po przeprowadzeniu tych operacji jest wykonywana druga pętla `for`, która wysyła do przeglądarki aktualną zawartość tablicy. Tym samym po wykonaniu skryptu na ekranie pojawi się widok zaprezentowany na rysunku 4.3.



Rysunek 4.3. *Ilustracja działania skryptu z listingu 4.3*

Tablice asocjacyjne

Oprócz tablic indeksowanych numerycznie istnieją w PHP również **tablice asocjacyjne**. W tablicach tego typu każdemu indeksowi można nadać unikalną nazwę, czyli zamiast indeksów `0`, `1`, `2` itd. mogą występować indeksy: `kolor`, `autor`, `procesor` itp. Najczęściej też zamiast terminu indeks stosuje się inny termin, a mianowicie **klucz**. Mówimy zatem, że w tablicy asocjacyjnej występują pary **klucz – wartość**, w których każdy klucz jednoznacznie identyfikuje przypisaną mu wartość. Tablicę tego typu tworzy się (podobnie jak w przypadku tablic klasycznych indeksowanych numerycznie) za pomocą słowa kluczowego `array`, konstrukcja ta ma jednak nieco inną postać. Schematycznie wygląda to następująco:

```
array
(
  klucz1 => wartość1,
  klucz2 => wartość2,
  ...
  kluczn => wartośćn
);
```

Na listingu 4.4 został przedstawiony krótki skrypt, w którym pokazano, w jaki sposób utworzyć tablicę asocjacyjną i odczytać zapisane w niej wartości.

Listing 4.4. *Utworzenie tablicy asocjacyjnej*

```
<?php
$kolory = array
(
  "kolor1" => "czerwony",
  "kolor2" => "zielony",
  "kolor3" => "niebieski"
);
echo "Zawartość tablicy:<br />";
echo "kolory['kolor1'] = ";
echo $kolory['kolor1'];
```

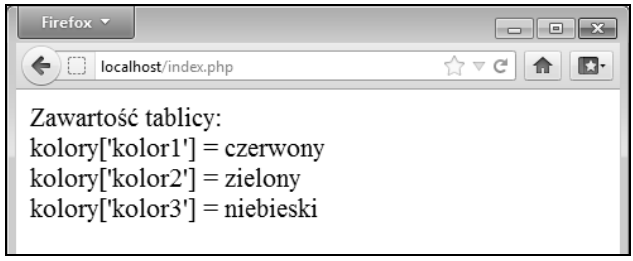
```
echo "<br />kolory['kolor2'] = ";
echo $kolory['kolor2'];

echo "<br />kolory['kolor3'] = ";
echo $kolory['kolor3'];
?>
```

W skrypcie została utworzona tablica `$kolory`, która zawiera trzy klucze o nazwach: `kolor1`, `kolor2` i `kolor3`. Kluczowi `kolor1` został przypisany ciąg znaków `czerwony`, kluczowi `kolor2` — ciąg znaków `zielony`, a kluczowi `kolor3` — ciąg znaków `niebieski`. Dzięki temu po zastosowaniu konstrukcji w schematycznej postaci:

```
nazwa_tablicy['nazwa_klucza']
```

otrzymujemy wartość odpowiadającą danemu kluczowi. Ta konstrukcja została użyta do wyświetlenia zawartości poszczególnych kluczy tablicy w przeglądarce. Instrukcja `echo $kolory['kolor1'];` wyświetla zawartość klucza `kolor1`, instrukcja `echo $kolory ↪['kolor2']` — klucza `kolor2`, a instrukcja `echo $kolory['kolor3']` — klucza `kolor3`. Tym samym na ekranie zobaczymy widok zaprezentowany na rysunku 4.4.

Rysunek 4.4. *Efekt działania skryptu z listingu 4.4*



Drugim ze sposobów tworzenia tablicy asocjacyjnej jest użycie składni z nawiasem kwadratowym, podobnie jak miało to miejsce w przypadku tablic indeksowanych numerycznie. Schematycznie taka konstrukcja ma postać:

```
nazwa_tablicy['nazwa_klucza'] = wartość_klucza;
```

Na listingu 4.5 został przedstawiony skrypt, który realizuje takie samo zadanie jak skrypt 4.4, czyli utworzenie tablicy asocjacyjnej i wyświetlenie jej zawartości, ale wykorzystuje zaprezentowaną powyżej składnię.

Listing 4.5. *Drugi sposób tworzenia tablic asocjacyjnych*

```
<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy:<br />";
echo "kolory['kolor1'] = ";
echo $kolory['kolor1'];

echo "<br />kolory['kolor2'] = ";
echo $kolory['kolor2'];
```

```
echo "<br />kolory['kolor3'] = ";
echo $kolory['kolor3'];
?>
```

Pierwsza instrukcja tego skryptu powoduje utworzenie tablicy asocjacyjnej `$kolory` oraz umieszczenie w niej klucza o nazwie `kolor1`, powiązanego z ciągiem znaków `czerwony`. Kolejne dwie instrukcje powodują umieszczenie w istniejącej już tablicy dwóch kolejnych kluczy: `kolor2` i `kolor3` oraz przypisanie do nich odpowiadających im wartości. Zawartość poszczególnych kluczy tak utworzonej tablicy jest następnie wysyłana do przeglądarki za pomocą serii instrukcji `echo`.

Do odczytu tablic asocjacyjnych można, podobnie jak w przypadku tablic klasycznych, użyć pętli. Nie może być to jednak pętla typu `for`, gdyż nie zdoła ona stwierdzić, jakie są wartości kluczy. Dlatego też tablice asocjacyjne są obsługiwane przez pętle typu `foreach` (por. rozdział 3, sekcja „Pętla typu foreach"). Taka pętla potrafi pobrać kolejne wartości kluczy. Jak to zrobić, zobrazowano w skrypcie z listingu 4.6.

Listing 4.6. *Wykorzystanie pętli typu foreach*

```
<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy:<br />";
foreach($kolory as $kolor){
  echo $kolor;
  echo "<br />";
}
?>
```

Konstrukcja tego typu pętli oznacza, że w każdym jej przebiegu pod zmienną `$kolor` będzie podstawiana wartość kolejnego klucza. A zatem zmienna `$kolor` w pierwszym przebiegu pętli będzie zawierała ciąg znaków `czerwony`, w drugim przebiegu — ciąg znaków `zielony`, a w trzecim przebiegu — ciąg znaków `niebieski`. W momencie gdy zostaną odczytane wartości wszystkich kluczy, pętla zakończy działanie. W ten sposób uzyskamy jednak jedynie wartości kluczy, nie zaś nazwy kluczy. Jeśli również ta informacja jest potrzebna, trzeba zastosować drugą wersję pętli `foreach`. Przykład tej konstrukcji został zaprezentowany na listingu 4.7.

Listing 4.7. *Inna wersja pętli foreach*

```
<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy:<br />";
foreach($kolory as $klucz => $kolor){
  echo "kolory['$klucz'] = $kolor";
  echo "<br />";
```

```
}
?>
```

Tym razem w każdym przebiegu pętli pod zmienną `$klucz` podstawiana jest nazwa kolejnego klucza, a pod zmienną `$kolor` — wartość przypisana temu kluczowi. Dzięki temu za pomocą instrukcji `echo` można wysłać do przeglądarki wszystkie istotne informacje o zawartości tablicy. Efekt działania kodu będzie taki sam jak skryptu z listingu 4.4 (rysunek 4.4).

Modyfikacji zawartości tablic asocjacyjnych dokonuje się tak samo jak zmian w przypadku tablic klasycznych. Oczywiście zamiast indeksów numerycznych trzeba zastosować wartości kluczy. Aby zatem przypisać nową wartość już istniejącemu kluczowi, trzeba skorzystać z konstrukcji, której schematyczna postać jest następująca:

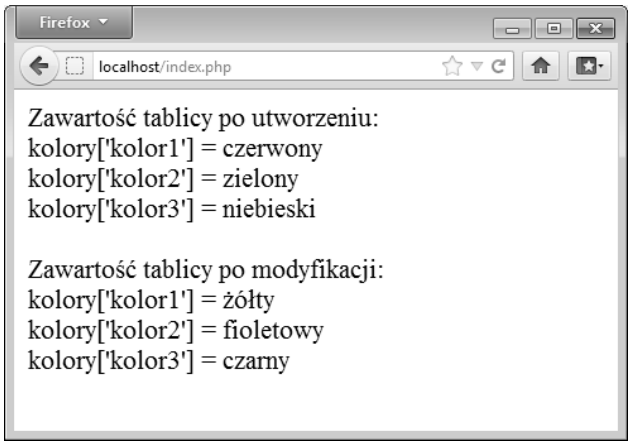
```
nazwa_tablicy['nazwa_klucza'] = wartość;
```

Na listingu 4.8 pokazany został przykładowy skrypt, który wykonuje modyfikację zawartości tablicy asocjacyjnej `$kolory`.

Listing 4.8. *Modyfikacja zawartości tablicy asocjacyjnej*

```
<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy po utworzeniu:<br />";
foreach($kolory as $klucz => $kolor){
  echo "kolory['$klucz'] = $kolor";
  echo "<br />";
}

$kolory['kolor1'] = "żółty";
$kolory['kolor2'] = "fioletowy";
$kolory['kolor3'] = "czarny";

echo "<br />Zawartość tablicy po modyfikacji:<br />";
foreach($kolory as $klucz => $kolor){
  echo "kolory['$klucz'] = $kolor";
  echo "<br />";
}
```

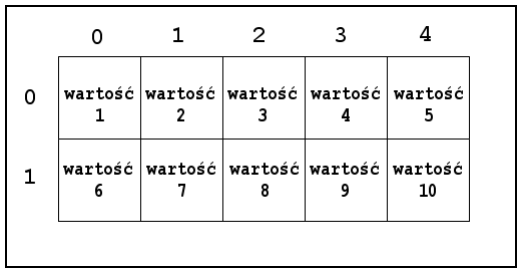
Tablica `$kolory` jest tu tworzona analogicznie jak w poprzednim przykładzie. Tak samo jest również wyświetlana jej pierwotna zawartość. Kluczowi `kolor1` został przypisany ciąg znaków `czerwony`, kluczowi `kolor2` — ciąg znaków `zielony`, a kluczowi `kolor3` — ciąg znaków `niebieski`. Po wykonaniu pętli `foreach`, która wyświetla te dane na ekranie, wykonywana jest instrukcja `$kolory['kolor1'] = "żółty";`. Ponieważ tablica `$kolory` już istnieje i jest w niej zawarty klucz `kolor1`, następuje modyfikacja przypisanej do niego wartości z `czerwony` na `żółty`. Analogiczne operacje wykonywane są z wartościami kluczy `kolor2` i `kolor3`. Po wykonaniu tych modyfikacji zawartość tablicy jest ponownie wysyłana do przeglądarki. Tym samym po uruchomieniu skryptu na ekranie pojawi się widok zaprezentowany na rysunku 4.5.



Rysunek 4.5. *Zawartość tablicy została zmodyfikowana*

Tablice wielowymiarowe

Do tej pory omawiane były tablice jednowymiarowe, czyli takie, które są wektorami elementów, o strukturze przedstawionej na rysunku 4.1. Aby odczytać dane z pojedynczej komórki, wystarczyło podać jej indeks lub w przypadku tablic asocjacyjnych — nazwę klucza. PHP umożliwia jednak budowanie bardziej skomplikowanych struktur — **tablic wielowymiarowych**. Przykładowa struktura prostej tablicy dwuwymiarowej została przedstawiona na rysunku 4.6. Jak widać, aby otrzymać wartość danej komórki, trzeba znać dwie liczby określające jej położenie: numer rzędu i numer kolumny. Na przykład komórka zawierająca `wartość 8` znajduje się w rzędzie o indeksie `1` i kolumnie o indeksie `2`.



Rysunek 4.6. *Struktura przykładowej tablicy dwuwymiarowej*

Tworzenie tablic wielowymiarowych

Do tworzenia tablic wielowymiarowych w PHP wykorzystuje się fakt, że pojedyncza komórka zwykłej tablicy jednowymiarowej może zawierać dane dowolnego typu, a zatem również inną tablicę. Wynika z tego, że tablica dwuwymiarowa to nic innego jak tablica jednowymiarowa, w której komórkach zawarte zostały inne tablice jednowymiarowe.

Spróbujmy wykonać prosty przykład. Na listingu 4.9 został zaprezentowany kod tworzący tablicę dwuwymiarową, w której komórkach zostały zawarte kolejne liczby od 1 do 6, a następnie wyświetlający jej zawartość na ekranie.

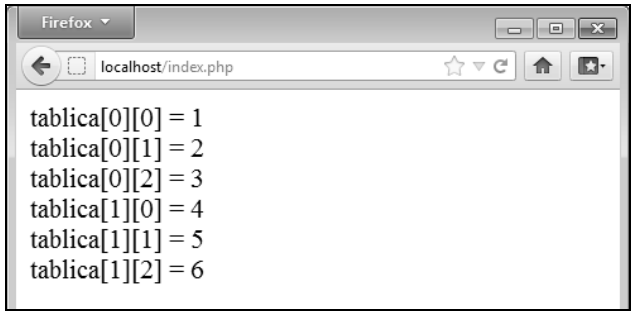
Listing 4.9. *Tworzenie tablicy dwuwymiarowej*

```
<?php
$tablica = array
(
  array(1, 2, 3),
  array(4, 5, 6)
);
echo "tablica[0][0] = " . $tablica[0][0] . "<br />";
echo "tablica[0][1] = " . $tablica[0][1] . "<br />";
echo "tablica[0][2] = " . $tablica[0][2] . "<br />";
echo "tablica[1][0] = " . $tablica[1][0] . "<br />";
echo "tablica[1][1] = " . $tablica[1][1] . "<br />";
echo "tablica[1][2] = " . $tablica[1][2] . "<br />";
?>
```

Konstrukcja tworząca tablicę `$tablica` dokładnie odzwierciedla sposób, w jaki ona powstaje. W pierwszej komórce (o indeksie `0`) została umieszczona tablica trójelementowa zawierająca liczby `1`, `2`, `3`, natomiast w komórce drugiej (o indeksie `1`) została umieszczona tablica (również trójelementowa) zawierająca liczby `4`, `5`, `6`. Powstała więc w ten sposób struktura o dwóch rzędach i trzech kolumnach. Dostęp do poszczególnych komórek wymaga zatem podania numeru wiersza i kolumny, co schematycznie wygląda następująco:

```
$tablica[wiersz][kolumna]
```

Ten sposób odwoływania się do komórek tablicy jest wykorzystywany (w instrukcjach `echo`) do wyświetlenia wszystkich zawartych w niej wartości w przeglądarce (rysunek 4.7).



Rysunek 4.7. *Wyświetlenie zawartości tablicy dwuwymiarowej*

Do odczytu zawartości takiej tablicy można również wykorzystać dwie zagnieżdżone pętle `for`. Taki sposób jest szczególnie przydatny wówczas, gdy tablica ma dużą liczbę wierszy i kolumn. W kodzie z listingu 4.10 zobrazowano, jak wykonać takie zadanie dla tablicy powstałej w poprzednim przykładzie.

Listing 4.10. *Wykorzystanie pętli for do odczytu tablicy*

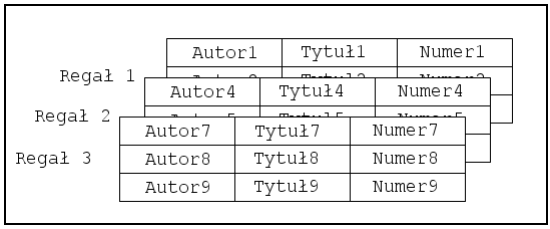
```
<?php
$tablica = array
(
  array(1, 2, 3),
  array(4, 5, 6)
);
for($i = 0; $i < 2; $i++){
  for($j = 0; $j < 3; $j++){
    $wart = $tablica[$i][$j];
    echo "tablica[$i][$j] = $wart";
    echo "<br />";
  }
  echo "<br />";
}
?>
```

Zewnętrzna pętla `for` (ze zmienną iteracyjną `$i`) kontroluje numer aktualnie odczytywanego wiersza tablicy, natomiast wewnętrzna pętla `for` (ze zmienną iteracyjną `$j`) kontroluje numer aktualnie odczytywanej kolumny tablicy. Wartości kolejnych komórek są po odczytaniu zapisywane w zmiennej pomocniczej `$wart`, która jest następnie wykorzystywana jako parametr w instrukcji `echo`.

Tablice wielowymiarowe nie muszą być indeksowane numerycznie — mogą być również strukturami asocjacyjnymi. Każdemu indeksowi można przypisać jego własną nazwę. Spróbujmy więc utworzyć i taką tablicę. Załóżmy, że mają być w niej przechowywane dane dotyczące książek w bibliotece. Pojedynczy wiersz będzie przechowywał dane dotyczące tytułu, autora i numeru katalogowego. Podstawowa tablica będzie mogła przechowywać wiele takich wierszy, a więc opisywać wiele książek. Jej struktura będzie zatem następująca:

```
$tablica = array(
  array("Autor" => "Autor1",
        "Tytuł" => "Tytuł1",
        "Numer" => "Numer1"),
  array("Autor" => "Autor2",
        "Tytuł" => "Tytuł2",
        "Numer" => "Numer2"),
  array("Autor" => "Autor3",
        "Tytuł" => "Tytuł3",
        "Numer" => "Numer3")
);
```

W ten sposób otrzymalibyśmy sam zbiór książek. Książki w bibliotece zazwyczaj stoją jednak na regałach. Można by więc wprowadzić dodatkową tablicę opisującą regały. Jej zawartością byłyby tablice opisujące książki. Powstałaby w ten sposób struktura trójwymiarowa, schematycznie przedstawiona na rysunku 4.8. Spróbujmy zbudować taką tablicę, wprowadzając do niej przykładowe dane, a następnie wyświetlmy jej zawartość na ekranie. To zadanie realizuje skrypt widoczny na listingu 4.11.



Rysunek 4.8. *Schematyczna struktura tablicy trójwymiarowej*

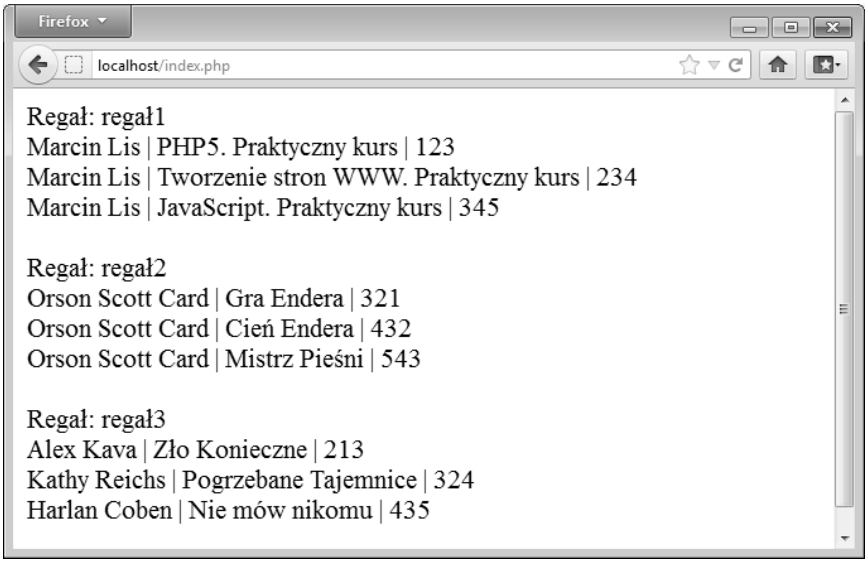
Listing 4.11. *Skrypt obsługujący tablicę trójwymiarową*

```
<?php
$biblioteka = array(
  'regał1' => array
  (
    array("Autor" => "Marcin Lis",
          "Tytuł" => "PHP5. Praktyczny kurs",
          "Numer" => "123"),
    array("Autor" => "Marcin Lis",
          "Tytuł" => "Tworzenie stron WWW. Praktyczny kurs",
          "Numer" => "234"),
    array("Autor" => "Marcin Lis",
          "Tytuł" => "JavaScript. Praktyczny kurs",
          "Numer" => "345")
   ),
  'regał2' => array
  (
    array("Autor" => "Orson Scott Card",
          "Tytuł" => "Gra Endera",
          "Numer" => "321"),
    array("Autor" => "Orson Scott Card",
          "Tytuł" => "Cień Endera",
          "Numer" => "432"),
    array("Autor" => "Orson Scott Card",
          "Tytuł" => "Mistrz Pieśni",
          "Numer" => "543")
   ),
  'regał3' => array
  (
    array("Autor" => "Alex Kava",
          "Tytuł" => "Zło Konieczne",
          "Numer" => "213"),
    array("Autor" => "Kathy Reichs",
          "Tytuł" => "Pogrzebane Tajemnice",
          "Numer" => "324"),
    array("Autor" => "Harlan Coben",
          "Tytuł" => "Nie mów nikomu",
          "Numer" => "435")
   )
);

foreach($biblioteka as $regal_nazwa => $regal){
  echo "Regał: $regal_nazwa<br />";
  foreach($regal as $ksiazka){
    $autor = $ksiazka['Autor'];
    $tytuł = $ksiazka['Tytuł'];
```

```
            $numer = $ksiazka['Numer'];
            echo "$autor | $tytuł | $numer";
            echo "<br />";
        }
        echo "<br />";
    }
```

W kodzie została umieszczona główna tablica o nazwie `$biblioteka`. Zawiera ona trzy klucze o nazwach `regał1`, `regał2` i `regał3`. Pod każdym kluczem znajdują się kolejne tablice, które zawierają informacje opisujące książki w danym regale. Każda taka tablica składa się z serii tablic jednowymiarowych o kluczach `Autor`, `Tytuł` i `Numer`. Razem tworzy to pełny opis książek w bibliotece. Ponieważ ręczne pobieranie danych w celu wyświetlenia całej zawartości tablicy `$biblioteka` byłoby bardzo niewygodne i czasochłonne, do ich prezentacji zostały wykorzystane dwie zagnieżdżone pętle `foreach`.

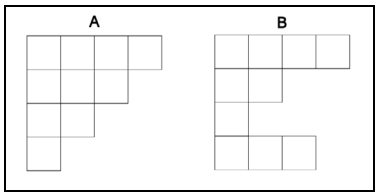
Pętla zewnętrzna odczytuje zawartość kluczy tablicy głównej `$biblioteka`. Pod zmienną `$regal_nazwa` podstawiane są nazwy odczytanych kluczy, natomiast pod zmienną `$regal` — ich zawartość. Zawartością każdego klucza jest tablica zawierająca spis książek z danego regału, a zatem do jej odczytania wykorzystywana jest wewnętrzna pętla `foreach`. Pętla ta odczytuje zawartość kolejnych komórek tablicy `$regal`, podstawiając je pod zmienną `$ksiazka`. Zawartość tej zmiennej jest po prostu tablicą jednowymiarową, która opisuje pojedynczą książkę. Indeksami tej tablicy są więc: `Autor`, `Tytuł` i `Numer`. Dane te są odczytywane, zapisywane w zmiennych pomocniczych i wysyłane do przeglądarki za pomocą instrukcji `echo`. Ostatecznie na ekranie zobaczymy zawartość całej biblioteki z podziałem na regały, tak jak zostało to przedstawione na rysunku 4.9.



Rysunek 4.9. *Efekt działania skryptu z listingu 4.11*

Tablice nieregularne

Tablice wielowymiarowe wcale nie muszą mieć takich regularnie prostokątnych kształtów jak dotychczas prezentowane. Prostokątnych, to znaczy takich, gdzie w każdym wierszu znajduje się taka sama liczba komórek (czyli struktur podobnych do prezentowanej na rysunku 4.6). Nic nie stoi na przeszkodzie, aby stworzyć strukturę trójkątną (rysunek 4.10 A) lub też całkiem nieregularną (rysunek 4.10 B). Przygotowywanie takich struktur wymaga jednak więcej pracy niż w przypadku tablic regularnych, gdyż przeważnie każdy wiersz trzeba tu tworzyć oddzielnie.



Rysunek 4.10. *Przykłady nieregularnych tablic wielowymiarowych*

Jak tworzyć tego typu struktury? Wiadomo już, że tablice wielowymiarowe to tak naprawdę tablice tablic jednowymiarowych. A zatem tablica dwuwymiarowa to tablica jednowymiarowa zawierająca szereg tablic jednowymiarowych, tablica trójwymiarowa to tablica jednowymiarowa zawierająca w sobie tablice dwuwymiarowe itd. Spróbujmy zatem stworzyć strukturę widoczną na rysunku 4.10 B, wypełnioną wartościami od 1 do 10, i wyświetlić jej zawartość w przeglądarce. To zadanie realizuje kod widoczny na listingu 4.12.

Listing 4.12. *Tworzenie tablicy nieregularnej*

```
<?php
$tablica = array
(
  array(1, 2, 3, 4),
  array(5, 6),
  array(7),
  array(8, 9, 10)
);
foreach($tablica as $tab){
  foreach($tab as $val){
    echo "$val ";
  }
  echo "<br />";
}
?>
```

Postać skryptu nie powinna być żadnym zaskoczeniem. Sposób tworzenia tablicy jest analogiczny do przedstawionego w poprzednich przykładach, z tą różnicą że tym razem tablice składowe mają różne wielkości. Pierwsza zawiera cztery komórki, druga — dwie, trzecia — jedną, a czwarta — trzy. Sposób odczytywania zawartości jest również podobny jak w przykładzie z listingu 4.11, a nawet nieco prostszy. Pętla zewnętrzna odczytuje kolejne komórki tablicy `$tablica`. Każda z tych komórek zawiera kolejną

tablicę o pewnej liczbie elementów, które odczytywane są za pomocą wewnętrznej pętli `foreach`. Tym samym po uruchomieniu skryptu zobaczymy taki widok jak na rysunku 4.11.

Rysunek 4.11. *Zawartość tablicy nieregularnej z przykładu 4.12*

Operacje na tablicach

Sortowanie tablic klasycznych

Jedną z operacji często wykonywanych na tablicach jest sortowanie, czyli ustawienie elementów w określonym porządku. PHP oferuje kilka wbudowanych funkcji sortujących. Zobaczmy, w jaki sposób można z nich korzystać. Funkcją podstawową jest `sort`. Działa ona zarówno na wartościach liczbowych, jak i na ciągach znaków. Jako argument jej wywołania należy podać nazwę tablicy. Spójrzmy na listing 4.13. Zawiera on kod sortujący dwie różne tablice.

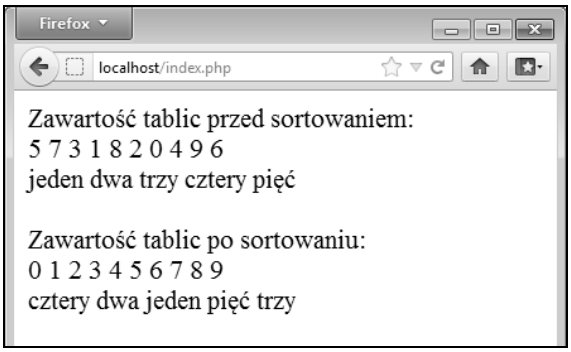
Listing 4.13. *Sortowanie za pomocą funkcji sort*

```
<?php
$tab1 = array(5, 7, 3, 1, 8, 2, 0, 4, 9, 6);
$tab2 = array('jeden', 'dwa', 'trzy', 'cztery', 'pięć');

echo "Zawartość tablic przed sortowaniem: <br />";
foreach($tab1 as $val){
  echo "$val ";
}
echo "<br />";
foreach($tab2 as $val){
  echo "$val ";
}

sort($tab1);
sort($tab2);

echo "<br /><br />Zawartość tablic po sortowaniu: <br />";
foreach($tab1 as $val){
  echo "$val ";
}
echo "<br />";
foreach($tab2 as $val){
```

```
    echo "$val ";
}
?>
```

Efekt wykonania skryptu został przedstawiony na rysunku 4.12. Jak widać, obie tablice zostały poprawnie posortowane. Oczywiście w przypadku tablicy `$tab1` sortowane były liczby, więc wartości zostały ustawione od najmniejszej do największej, natomiast w przypadku tablicy `$tab2` sortowane były ciągi znaków, a zatem słowa zostały ustawione w porządku alfabetycznym. Co jednak zrobić w sytuacji, gdyby trzeba było wykonać sortowanie odwrotne, czyli np. ustawić wartości z tablicy `$tab1` od największej do najmniejszej? Nie ma z tym najmniejszego problemu. Wystarczy użyć funkcji `rsort` (z ang. *reverse sort*), która wykona to zadanie.

Rysunek 4.12. *Efekt sortowania tablic*



Więcej problemów przysporzy sytuacja, w której konieczne będzie ustawienie elementów tablicy w specyficznej kolejności, odmiennej od standardowego porządku. Do wyboru są wtedy dwie drogi. Trzeba albo samodzielnie napisać całą funkcję wykonującą sortowanie, albo też wykorzystać specjalną wersję funkcji sortującej — `usort` — w połączeniu z funkcją porównującą dwa elementy. Schematyczne wywołanie takiej funkcji ma postać:

```
usort($tablica, 'nazwa_funkcji')
```

gdzie *`$tablica`* to nazwa tablicy, której elementy będą sortowane, a *`nazwa_funkcji`* to nazwa funkcji dokonującej porównania dwóch elementów. Ta ostatnia funkcja będzie otrzymywała dwa elementy sortowanej tablicy w postaci argumentów, musi natomiast zwracać:

- wartość mniejszą od zera, jeśli pierwszy argument jest mniejszy od drugiego;
- wartość większą od zera, jeśli pierwszy argument jest większy od drugiego;
- wartość równą zero, jeśli pierwszy argument jest równy drugiemu.

Zobaczmy, jak to działa na konkretnym przykładzie. Powstanie skrypt, który zawartość tablicy przechowującej liczby całkowite będzie sortował w taki sposób, że najpierw umieszczone będą wartości podzielne przez dwa (od najmniejszej do największej), a dopiero po nich wartości niepodzielne przez dwa — również od najmniejszej do największej. To zadanie realizuje kod z listingu 4.14.

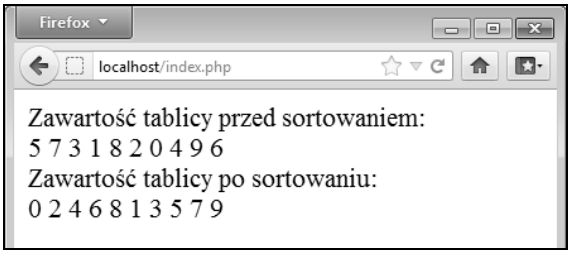
Listing 4.14. *Realizacja niestandardowego sortowania*

```
<?php
function sortuj($e1, $e2)
{
  if($e1 % 2 == 0){
    if($e2 % 2 == 0){
      return $e1 - $e2;
    }
    else{
      return -1;
    }
  }
  else{
    if($e2 % 2 == 0){
      return 1;
    }
    else{
      return $e1 - $e2;
    }
  }
}

$tab1 = array(5, 7, 3, 1, 8, 2, 0, 4, 9, 6);

echo "Zawartość tablicy przed sortowaniem: <br />";
foreach($tab1 as $val){
  echo("$val ");
}
echo "<br />";

usort($tab1, 'sortuj');

echo "Zawartość tablicy po sortowaniu: <br />";
foreach($tab1 as $val){
  echo "$val ";
}
?>
```

Tablica jest tworzona w standardowy sposób i jej początkowa zawartość jest wyświetlana na ekranie. Następnie wywoływana jest funkcja `usort`, która wykonuje operację sortowania, a potem zawartość posortowanej tablicy jest ponownie wyświetlana na ekranie. Tym samym w przeglądarce ukaże się obraz widoczny na rysunku 4.13. Jak widać, układ liczb jest zgodny z założeniami — najpierw umieszczone są liczby podzielne przez dwa (od najmniejszej do największej), a za nimi liczby niepodzielne przez dwa (również od najmniejszej do największej). Za takie uporządkowanie elementów tablicy odpowiada kombinacja funkcji `usort` i `sortuj`.

Funkcja `usort` realizuje algorytm sortowania typu QuickSort. Ponieważ sortowanie ma odbywać się według niestandardowych zasad, trzeba tej funkcji dostarczyć dodatkową funkcję, która będzie porównywała dwa dowolne elementy tablicy `tab1`. Tą funkcją jest `sortuj`. Przy porównywaniu dwóch dowolnych elementów tablicy `tab1` możliwe są cztery różne sytuacje:

Rysunek 4.13. *Efekt niestandardowego sortowania*



1. Pierwszy argument jest podzielny przez dwa (`e1 % 2` równe `0`) i drugi argument jest również podzielny przez dwa (`e2 % 2` równe `0`). W takiej sytuacji należy zwrócić wartość mniejszą od zera, jeśli pierwszy argument jest mniejszy; wartość większą od zera, jeśli drugi argument jest mniejszy; lub wartość `0`, jeśli argumenty są równe. Zapewnia to instrukcja `return $e1 - $e2;`.
2. Pierwszy argument jest podzielny przez dwa (`e1 % 2` równe `0`), natomiast drugi argument nie jest podzielny przez dwa (`e2 % 2` różne od `0`). W takiej sytuacji argument pierwszy zawsze powinien znaleźć się przed argumentem drugim, a zatem należy zwrócić wartość mniejszą od zera. Zapewnia to instrukcja `return -1;`.
3. Pierwszy argument nie jest podzielny przez dwa (`e1 % 2` różne od `0`), a drugi argument jest podzielny przez dwa (`e2 % 2` równe `0`). W takiej sytuacji argument pierwszy zawsze powinien znaleźć się za argumentem drugim, a zatem należy zwrócić wartość większą od zera. Zapewnia to instrukcja `return 1;`.
4. Pierwszy argument nie jest podzielny przez dwa (`e1 % 2` różne od `0`) i drugi argument również nie jest podzielny przez dwa (`e2 % 2` różne od `0`). W takiej sytuacji należy zwrócić wartość mniejszą od zera, jeśli pierwszy argument jest mniejszy; wartość większą od zera, jeśli drugi argument jest mniejszy; oraz wartość `0`, jeśli argumenty są równe. Zapewnia to instrukcja `return $e1 - $e2;`.

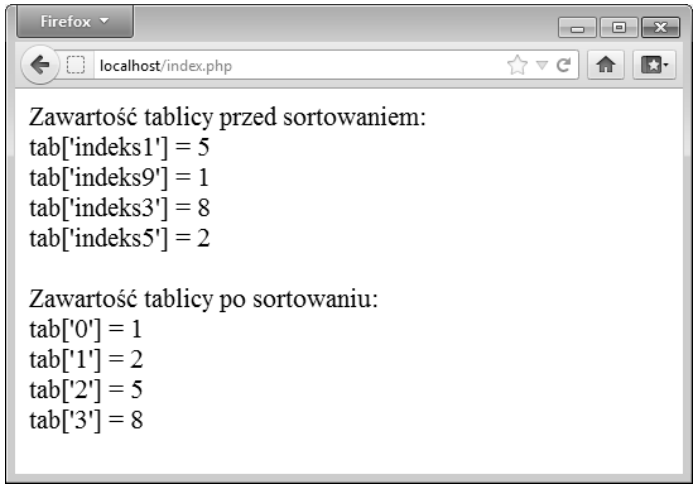
Sortowanie tablic asocjacyjnych

W przypadku tablic asocjacyjnych nie można użyć zwykłej funkcji `sort`, gdyż spowoduje ona utratę kluczy. Łatwo się o tym przekonać, uruchamiając skrypt widoczny na listingu 4.15. Została w nim utworzona tablica `$tab` zawierająca cztery klucze z przypisanymi wartościami całkowitymi. Tablica ta została następnie posortowana za pomocą funkcji `sort`. Zawartość przed sortowaniem i po nim została wyświetlona za pomocą pętli `foreach` i instrukcji `echo`. Jak widać na rysunku 4.14, efekt takiego działania nie jest zadowalający. Co prawda wartości zostały posortowane, ale jednocześnie zostały utracone nazwy indeksów.

Listing 4.15. *Użycie funkcji sort do sortowania tablicy asocjacyjnej*

```
<?php
$tab = array
(
  'indeks1' => 5,
  'indeks9' => 1,
```

```
  'indeks3' => 8,
  'indeks5' => 2
);
echo "Zawartość tablicy przed sortowaniem:<br />";
foreach($tab as $key => $val){
  echo "tab['$key'] = $val";
  echo "<br />";
}

sort($tab);

echo "<br />Zawartość tablicy po sortowaniu:<br />";
foreach($tab as $key => $val){
  echo "tab['$key'] = $val";
  echo "<br />";
}
?>
```



Rysunek 4.14. *Utrata nazw indeksów po nieprawidłowym sortowaniu tablicy asocjacyjnej*

Aby zatem posortować tablicę asocjacyjną, trzeba użyć innych funkcji, a mianowicie: `asort` i `ksort`. Pierwsza z nich sortuje tablicę względem wartości poszczególnych kluczy, natomiast druga względem samych kluczy. Oznacza to, że jeśli w skrypcie z listingu 4.15 zamieni się funkcję `sort` na `asort`, po sortowaniu kolejność będzie następująca:

```
tab['indeks9'] = 1
tab['indeks5'] = 2
tab['indeks1'] = 5
tab['indeks3'] = 8
```

Natomiast po zamianie funkcji `sort` na `ksort` uzyska się wynik:

```
tab['indeks1'] = 5
tab['indeks3'] = 8
tab['indeks5'] = 2
tab['indeks9'] = 1
```

Sortowanie może się odbywać również w porządku odwrotnym, czyli od wartości największej do najmniejszej. Służą do tego celu funkcje `arsort` (sortowanie względem wartości) i `krsort` (sortowanie względem kluczy).

Implozja i eksplozja

Bardzo ciekawymi i (jak się okaże przy realizacji praktycznych projektów) użytecznymi funkcjami są `implode` i `explode` (stąd też powyższy podtytuł). Pierwsza powoduje zwrócenie wszystkich elementów tablicy rozdzielonych znakami separatora jako ciągu znaków. Wywołanie ma schematyczną postać:

```
implode(separator, tablica)
```

Jeśli na przykład istnieje tablica `$arr` o postaci:

```
$arr = array('jeden', 'dwa', 'trzy');
```

to wykonanie instrukcji:

```
echo implode(', ', $arr);
```

spowoduje wyświetlenie ciągu znaków:

```
jeden, dwa, trzy
```

Podobnie, efektem działania instrukcji:

```
echo implode('_', $arr);
```

będzie ciąg:

```
jeden_dwa_trzy
```

Funkcja `explode` (jak sama nazwa wskazuje) działa odwrotnie niż `implode`. Tworzy więc tablicę składającą się z fragmentów ciągu znaków wydzielonych przez znaki separatora. Jej wywołanie ma ogólną postać:

```
explode(separator, ciąg[, ile]);
```

Jeśli na przykład istnieje ciąg `$str` w postaci:

```
$str = 'jeden, dwa, trzy, cztery';
```

to wykonanie instrukcji:

```
$arr = explode(', ', $str);
```

spowoduje powstanie tablicy o następujących elementach:

```
Array
(
  [0] => jeden
  [1] => dwa
  [2] => trzy
  [3] => cztery
)
```

Gdy opcjonalny argument `ile` ma wartość dodatnią, wskazuje maksymalną liczbę elementów tablicy wynikowej. Ostatni element będzie zawierał pozostałą część ciągu. Jeżeli zatem istnieje taki ciąg `$str` jak podany wyżej, efektem wykonania instrukcji:

```
$arr = explode(', ', $str, 3);
```

będzie tablica:

```
Array
(
  [0] => jeden
  [1] => dwa
  [2] => trzy, cztery
)
```

Jeśli argument `ile` ma wartość ujemną, określa, ile elementów należy usunąć z końca tablicy wynikowej. Przykładowo dla zdefiniowanego wyżej ciągu `$str` efektem działania instrukcji:

```
$arr = explode(', ', $str, -2);
```

będzie tablica:

```
Array
(
  [0] => jeden
  [1] => dwa
)
```

Operacje na elementach tablic

Zmiana kolejności elementów

Jeśli chcemy odwrócić kolejność elementów w tablicy, czyli spowodować, aby pierwszy stał się ostatnim, drugi przedostatnim itd., możemy zastosować funkcję `array_reverse`. Jako argument tej funkcji należy przekazać nazwę tablicy. Tablica ze zmienioną kolejnością elementów zostanie zwrócona jako wynik działania funkcji, a zawartość oryginalnej tablicy nie zostanie naruszona. Sposób działania funkcji `array_reverse` zobrazowano w przykładzie widocznym na listingu 4.16.

Listing 4.16. *Odwrócenie kolejności elementów tablicy*

```
<?php
$tab1 = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$tab2 = array_reverse($tab1);

echo "Zawartość tablicy tab1:<br />";
foreach($tab1 as $val){
  echo("$val ");
}
echo "<br />Zawartość tablicy tab2:<br />";
foreach($tab2 as $val){
```

```
  echo "$val ";
}
?>
```

Poruszanie się po tablicy

Każda tablica w PHP ma wewnętrzny wskaźnik wskazujący jej bieżący element. Po utworzeniu tablicy wskaźnik ten jest ustawiony na pierwszy element. Podczas wykonywania operacji na elementach tablicy jego położenie może się zmieniać. Istnieją funkcje, które wykorzystują go do własnych potrzeb, istnieje również możliwość bezpośredniej manipulacji pozycją wskaźnika. Jedną z takich funkcji jest `each`. Jej zadaniem jest pobranie aktualnego elementu tablicy i przesunięcie wskaźnika o jedno miejsce w przód. Jeżeli wskaźnik znajdzie się na końcu tablicy, wywołanie `each` powoduje zwrócenie wartości `false`. Takie działanie funkcji `each` umożliwia zastosowanie jej w pętli `while` przetwarzającej elementy tablicy. Należy jedynie pamiętać, że wynikiem działania `each` jest w rzeczywistości czteroelementowa tablica (!) zawierająca cztery klucze: `0`, `1`, `key`, `value`, gdzie `0` i `key` przechowują pobrany klucz (indeks), a `1` i `value` odpowiadającą mu wartość. Zobrazowano to w przykładzie z listingu 4.17.

Listing 4.17. *Użycie funkcji each do odczytu elementów tablicy*

```
<?php
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
while($val = each($tab)){
  echo "val[0] = $val[0] | ";
  echo "val[1] = $val[1] | ";
  echo "val['key'] = $val[key] | ";
  echo "val['value'] = $val[value] ";
  echo "<br />";
}
?>
```

Funkcje, które pozwalają na bezpośrednią modyfikację wewnętrznego wskaźnika tablicy, to:

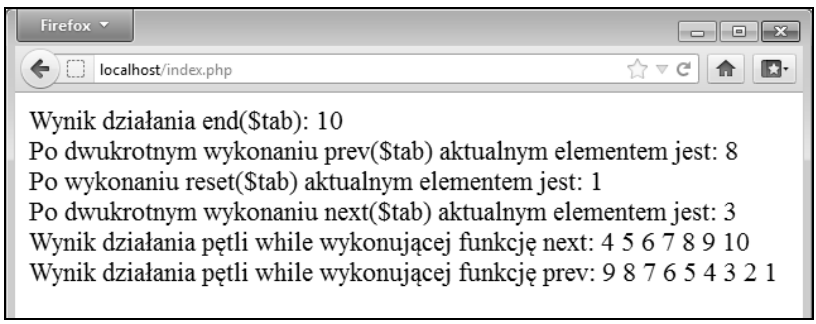
- `reset` — resetuje wskaźnik tablicy, ustawiając go na pierwszym elemencie. Funkcja jednocześnie zwraca wartość pierwszego elementu.
- `next` — przesuwa wskaźnik tablicy na następny element i zwraca wartość tego elementu. Jeśli aktualną pozycją wskaźnika tablicy jest jej ostatni element, funkcja zwraca wartość `false`.
- `prev` — przesuwa wskaźnik tablicy na poprzedni element (w stosunku do pozycji bieżącej) i zwraca wartość tego elementu. Jeśli aktualną pozycją wskaźnika tablicy jest jej pierwszy element, funkcja zwraca wartość `false`.
- `end` — ustawia wskaźnik tablicy na jej ostatnim elemencie i zwraca wartość tego elementu.

Oprócz wymienionych wyżej funkcji modyfikujących wewnętrzny wskaźnik istnieją również dwie funkcje, które pobierają aktualny element tablicy. Są to: `current` i `pos`.

Przykłady wykorzystania tego typu konstrukcji języka zostały zaprezentowane na listingu 4.18. Efekt działania skryptu jest natomiast przedstawiony na rysunku 4.15.

Listing 4.18. *Wykorzystanie funkcji operujących na wewnętrznym wskaźniku tablicy*

```
<?php
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$val = end($tab);

echo "Wynik działania end(\$tab): $val<br />";

prev($tab);
prev($tab);

$val = current($tab);
echo "Po dwukrotnym wykonaniu prev(\$tab) ";
echo "aktualnym elementem jest: $val<br />";

$val = reset($tab);
echo "Po wykonaniu reset(\$tab) aktualnym elementem jest: $val<br />";

next($tab);
next($tab);

$val = current($tab);
echo "Po dwukrotnym wykonaniu next(\$tab) ";
echo "aktualnym elementem jest: $val<br />";

echo "Wynik działania pętli while wykonującej funkcję next: ";
while($val = next($tab)){
  echo "$val ";
}

end($tab);

echo "<br />Wynik działania pętli while wykonującej funkcję prev: ";
while($val = prev($tab)){
  echo "$val ";
}
?>
```



Rysunek 4.15. *Efekt działania skryptu wykorzystującego funkcje operujące na wskaźniku tablicy*

Na początku skryptu powstała tablica `$tab` zawierająca 10 kolejnych liczb całkowitych. Następnie została wykonana operacja `end($tab)`, a jej wynik — przypisany zmiennej `$val`. Wartością tej zmiennej stała się więc wartość znajdująca się w ostatniej komórce tablicy, czyli 10. W kolejnym kroku zostały wykonane dwie operacje `prev($tab)`, co oznacza, że wewnętrzny wskaźnik tablicy został przesunięty o dwie pozycje do tyłu. Przekonujemy się o tym, pobierając aktualny element tablicy (`$val = current($tab);`) i wyświetlając go w przeglądarce za pomocą instrukcji `echo`.

Kolejny krok to wykonanie funkcji `reset`, która przesuwa wskaźnik na początek tablicy (a zatem aktualnym elementem staje się komórka o indeksie `0`). Po wykonaniu funkcji `reset` dwukrotnie wykonywana jest funkcja `next`, czyli wskaźnik jest przesuwany o dwie pozycje do przodu i wskazuje na trzeci element (o indeksie `2`). Dalej w kodzie została umieszczona pętla `while` przeglądająca kolejne elementy tablicy. Wykorzystuje ona fakt, że funkcja `next` przesuwa wskaźnik o jedno miejsce i zwraca wartość wskazanego elementu. W przypadku gdy wskaźnik zostanie przesunięty za ostatni element, funkcja zwraca wartość `false`, co jest warunkiem zakończenia pętli.

Ponieważ po ostatnim wykonaniu funkcji `next` wskaźnik tablicy został przesunięty za ostatni element, po zakończeniu pętli jest wykonywana funkcja `end`, która przesuwa go z powrotem na ostatni element. Dzięki temu może poprawnie zadziałać kolejna pętla `while`, która wykonuje serię funkcji `prev`, przesuwających wskaźnik tablicy do tyłu, za każdym wywołaniem o jedną pozycję. Gdy wskaźnik znajdzie się przed pierwszym elementem, wywołanie funkcji `prev` zwróci wartość `false` i tym samym pętla zakończy działanie.

Dodawanie i pobieranie elementów

W PHP istnieją wbudowane funkcje, które pozwalają na dodawanie i usuwanie elementów z początku i z końca tablicy. Są to: `array_pop`, `array_shift`, `array_put` i `array_unshift`. Funkcja `array_pop` pobiera element znajdujący się na końcu tablicy i zwraca jego wartość. Tym samym tablica zostaje skrócona o ostatni element. Schematycznie operacja taka ma postać:

```
$zmienna = array_pop($tablica);
```

Podobne zadanie wykonuje `array_shift`, ale z tą różnicą że usuwany jest pierwszy element. Jeżeli tablica była indeksowana numerycznie, wówczas wszystkie elementy zostaną przenumerowane, czyli indeks każdego z nich zmniejszy się o jeden.

Funkcja `array_push` działa odwrotnie niż `array_pop`. Dodaje ona elementy przekazane w postaci argumentów na końcu tablicy. Schematycznie operację tę można przedstawić jako:

```
array_push($tablica, element1, element2,..., elementN);
```

Funkcja zwraca wartość określającą liczbę elementów w powiększonej tablicy. Podobnie jak `array_push` działa `array_unshift` — dodaje ona określoną liczbę elementów na początku tablicy. Jeśli tablica była indeksowana numerycznie, zostanie ona również odpowiednio przenumerowana. Wywołanie funkcji `array_unshift` ma schematyczną postać:

```
array_unshift($tablica, element1, element2,..., elementN);
```

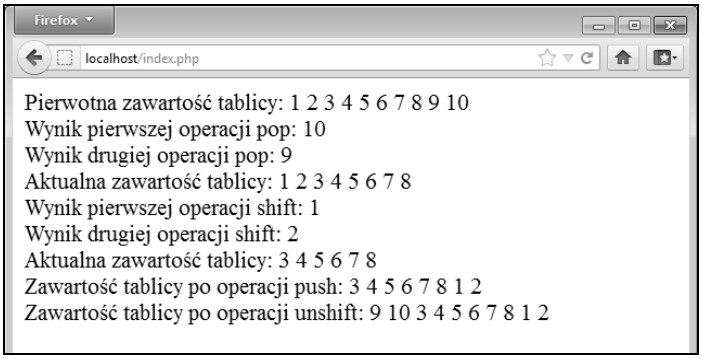
Sposób wykorzystania wymienionych funkcji w działającym skrypcie zobrazowano w kodzie widocznym na listingu 4.19. Efekt jego działania został natomiast zaprezentowany na rysunku 4.16.

Listing 4.19. *Ilustracja działania funkcji modyfikujących zawartość tablicy*

```
<?php
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
echo "Pierwotna zawartość tablicy: ";
foreach($tab as $val){
  echo "$val ";
}

$val = array_pop($tab);
echo("<br />Wynik pierwszej operacji pop: $val <br />");
$val = array_pop($tab);
echo "Wynik drugiej operacji pop: $val <br />";

echo "Aktualna zawartość tablicy: ";
foreach($tab as $val){
  echo "$val ";
}

$val = array_shift($tab);
echo "<br />Wynik pierwszej operacji shift: $val <br />";
$val = array_shift($tab);
echo "Wynik drugiej operacji shift: $val <br />";

echo "Aktualna zawartość tablicy: ";
foreach($tab as $val){
  echo "$val ";
}

array_push($tab, 1, 2);
echo "<br />Zawartość tablicy po operacji push: ";
foreach($tab as $val){
  echo "$val ";
}

array_unshift($tab, 9, 10);
echo "<br />Zawartość tablicy po operacji unshift: ";
foreach($tab as $val){
  echo "$val ";
}
?>
```

W skrypcie tworzona jest tablica $tab, która początkowo zawiera uporządkowane rosnąco wartości od 1 do 10. Wykonanie dwóch operacji `array_pop($tab);` powoduje usunięcie dwóch ostatnich wartości, a zatem pozostają komórki z wartościami od 1 do 8. Następnie są wykonywane dwie operacje `array_shift($tab);`, które usuwają dwie pierwsze komórki; tym samym w tablicy pozostają wartości od 3 do 8. Należy zwrócić uwagę, że przenumerowaniu uległy również indeksy komórek. Wartość 3 znajduje się

Rysunek 4.16. *Efekt działania skryptu z listingu 4.19*



obecnie pod indeksem `0`, wartość `4` pod indeksem `1` itd. Kolejną wykonywaną operacją jest `array_push($tab, 1, 2);`, która powoduje dodanie na końcu tablicy dwóch komórek, pierwszej o wartości `1` i drugiej o wartości `2`. Operacja `array_unshift($tab, 9, 10);` powoduje natomiast dodanie na początku tablicy dwóch komórek, pierwszej o wartości `9` i drugiej o wartości `10`. Ostatecznie tablica zawiera zatem ciąg wartości `9`, `10`, `3`, `4`, `5`, `6`, `7` `8`, `1`, `2`, tak jak jest to widoczne na rysunku 4.16.

Liczba elementów tablicy

W wielu przypadkach bardzo przydaje się ustalenie rozmiaru tablicy, czyli stwierdzenie, ile zawiera ona elementów. W PHP wykorzystywana jest w tym celu funkcja `count` (zamiast `count` można również użyć `sizeof`, która jest aliasem dla `count`). Jeśli zatem zostanie wykonany kod:

```
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$rozmiar = count($tab);
```

w zmiennej `$rozmiar` zostanie zapisana wartość `10`.

Nieco bardziej złożoną czynnością jest stwierdzenie, ile razy dana wartość występuje w tablicy. W PHP istnieje specjalna funkcja, która wykonuje to zadanie: `array_count_values`. Zwraca ona tablicę asocjacyjną, której kluczami są wartości tablicy oryginalnej, natomiast wartościami kluczy jest liczba wystąpień tych wartości w tablicy oryginalnej. Najlepiej zobaczyć, jak to działa, zapoznając się z konkretnym przykładem przedstawionym na listingu 4.20. Efekt jego działania został zaprezentowany na rysunku 4.17.

Listing 4.20. *Ilustracja działania funkcji array_count_values*

```
<?php
$tab = array(1, 2, 5, 1, 5, 1, 5, 1, 8, 2);
$values = array_count_values($tab);
foreach($values as $key => $val){
  echo "[$key] => $val <br />";
}
?>
```

Rysunek 4.17.
Efekt działania funkcji array_count_values

W skrypcie została utworzona tablica `$tab` zawierająca zbiór liczb. Wykonanie funkcji `array_count_values` spowodowało zwrócenie tablicy asocjacyjnej, której zawartość została wyświetlona w przeglądarce. Widać wyraźnie, że wartość `1` w oryginalnej tablicy występuje cztery razy, wartość `2` — dwa razy, wartość `5` — trzy razy, a wartość `8` — jeden raz.

Skorowidz

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

PHP i MySQL. Dla każdego. Wydanie II

PHP i MySQL są w świecie nowoczesnych, dynamicznych serwisów nierozłączne niemal jak Bonnie i Clyde, jednak w przeciwieństwie do pary słynnych przestępców mają przed sobą wspaniałą przyszłość. To właśnie dzięki tym dwóm technologiom funkcjonuje większość rozbudowanych witryn internetowych, na nich też opiera się działanie wielu najbardziej popularnych systemów zarządzania treścią. Nic w tym dziwnego, bo duet składający się z PHP i MySQL-a zapewnia ogromne możliwości, dużą elastyczność i wysoką stabilność pracy. Ponadto wystarczy znaleźć odpowiednie źródło wiedzy, by łatwo i szybko poznać obydwa te rozwiązania oraz opanować sposoby stosowania ich w praktyce.

Takim źródłem jest książka *PHP i MySQL. Dla każdego. Wydanie II*. Dzięki niej początkujący twórcy serwisów mogą bez kompleksów wkroczyć w świat nowoczesnych technologii internetowych i nauczyć się sprawnie używać bezpłatnych narzędzi do pisania, testowania czy wdrażania aplikacji WWW, a bardziej zaawansowani — usystematyzować i rozszerzyć posiadaną wiedzę. Podręcznik krok po kroku prezentuje sposoby instalacji i konfiguracji środowiska pracy, omawia składnię i możliwości języka PHP, a także wskazuje, jak poprawnie projektować i tworzyć bazy danych oparte na mechanizmie MySQL. Z książki dowiesz się, jak posługiwać się językiem SQL oraz jak połączyć możliwości PHP i MySQL-a, a ponadto jak za ich pomocą tworzyć praktyczne serwisy WWW.

Obowiązkowa pozycja w biblioteczce każdego początkującego twórcy serwisów WWW.

helion.pl
księgarnia internetowa
Helion

Nr katalogowy: 10306

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

- Instalacja i konfiguracja niezbędnych narzędzi w różnych systemach operacyjnych
- Przegląd typów, stałych i operatorów języka PHP, tworzenie zmiennych i ich używanie
- Instrukcje sterujące i pętle, budowa funkcji i korzystanie z nich
- Tworzenie tablic i ich stosowanie, elementy programowania obiektowego w PHP
- Nawigowanie w systemie plików, korzystanie z cookies i obsługa sesji
- Zarządzanie bazami danych oraz kontami korzystających z nich użytkowników
- Praktyczne przykłady serwisów WWW opartych na języku PHP i bazie MySQL

sięgnij po WIĘCEJ

ISBN 978-83-246-4797-2

9 788324 647972

Cena 89,00 zł

Informatyka w najlepszym wydaniu