PHP5. Wprowadzenie

Autor: David Sklar
Tłumaczenie: Robert Górczyński
ISBN: 83-246-0288-7
Tytuł oryginału: Learning PHP 5
Format: B5, stron: 352

Poznaj możliwości języka PHP5

- Zbuduj komponenty dynamicznych witryn WWW
- Wykorzystaj funkcje z biblioteki PEAR
- Przetestuj kod i usuń błędy

Czas, kiedy statyczne witryny internetowe przyciągały rzesze odwiedzających, dawno już minął. Dziś trzeba znacznie więcej, aby zainteresować użytkowników internetu. Doskonale przygotowany projekt graficzny to tylko jeden z elementów. Najistotniejsza jest jednak zawartość witryny – aktualne treści, formularze umożliwiające kontakt z twórcami oraz mechanizmy pozwalające na personalizację witryny i przechowywanie artykułów w bazie danych. Doskonałym narzędziem do tworzenia takich elementów jest PHP5 – prosty język programowania o potężnych możliwościach i, co najważniejsze, dostępny nieodpłatnie.

Jeśli chcesz nauczyć się programowania w tym języku, sięgnij po książkę „PHP5. Wprowadzenie". Znajdziesz w niej wszystkie informacje niezbędne do tego, aby samodzielnie stworzyć dynamiczną witrynę WWW. Poznasz elementy języka PHP i nauczysz się stosować je, tworząc elementy strony WWW. Dowiesz się, jak łączyć witrynę WWW z bazą danych i w jaki sposób identyfikować jej użytkowników. Przeczytasz tu również o wykrywaniu i usuwaniu błędów ze skryptów oraz o tym, do czego możesz zastosować język PHP.

- Operacje na tekstach i liczbach
- Sterowanie przebiegiem programu
- Tworzenie interaktywnych formularzy
- Korzystanie z baz danych
- Mechanizmy sesji i obsługa plików cookie
- Przetwarzanie dat
- Operacje na plikach
- Generowanie i przetwarzanie dokumentów XML
- Testowanie kodu i usuwanie błędów

Tchnij życie w statyczne witryny internetowe – wykorzystaj PHP5!

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

ROZDZIAŁ 4.

Praca z tablicami

Tablice są zbiorami powiązanych wartości, na przykład danych wysłanych z formularza, nazwisk studentów w grupie lub wielkości populacji na liście miast. W rozdziale 2. dowiedziałeś się, że zmienne są nazwanymi stałymi, które przechowują wartość. Tablica jest pojemnikiem przechowującym wiele wartości różniących się między sobą.

Dzięki temu rozdziałowi dowiesz się, w jaki sposób pracować z tablicami. Podrozdział „Podstawy tablic" zawiera podstawowe informacje na temat tworzenia tablic oraz manipulowania jej elementami. Niejednokrotnie będziesz musiał wykonywać pewne czynności w odniesieniu do każdego elementu w tablicy, np. wyświetlić element bądź zbadać, czy zachodzą określone warunki. Podrozdział „Wykonywanie pętli na tablicach" wyjaśni kwestie używania pętli z tablicami za pomocą konstrukcji `foreach()` oraz `for()`. W podrozdziale „Modyfikacja tablic" wprowadzimy funkcje `implode()` oraz `explode()` zamieniające tablice w łańcuchy oraz łańcuchy na tablice. Innym rodzajem modyfikacji tablicy jest sortowanie, które zostanie przeanalizowane w podrozdziale „Sortowanie tablic". Na końcu rozdziału, w podrozdziale „Używanie tablic wielowymiarowych" zostanie poruszony temat tablic zawierających w sobie inne tablice.

Rozdział 6. z kolei pozwoli Ci zrozumieć, w jaki sposób przetwarzać dane formularzy, które interpreter PHP automatycznie umieszcza w tablicy. Kiedy otrzymujesz informacje z bazy danych, jak to zostało opisane w rozdziale 7., dane zostają często umieszczone w tablicy.

Podstawy tablic

Tablica składa się z *elementów*, zaś każdy element posiada *klucz* oraz *wartość*. Tablica przechowująca informacje na temat kolorów warzyw posiada nazwę warzywa jako klucz oraz kolor jako wartość klucza, co zostało pokazane na rysunku 4.1.

Tablica może posiadać tylko jeden element dla podanego klucza. W tablicy kolorów warzyw nie może być innego elementu z kluczem `kukurydza`, nawet jeśli jego wartością byłoby `niebieski`. Jednakże ta sama wartość może pojawiać się wielokrotnie w tej samej tablicy. Możesz więc posiadać pomarańczowe marchewki, pomarańczowe mandarynki oraz pomarańczowe pomarańcze.

Rysunek 4.1. Klucze i wartości tablicy

Dowolny łańcuch lub wartość liczbowa może być elementem klucza tablicy, na przykład `kukurydza`, `4`, `-36` lub `Solona kałamarnica`. Tablice i inne wartości nieskalarne[1] nie mogą być kluczami, ale mogą być elementami wartości. Element wartości może być łańcuchem, liczbą, wartością `prawda` lub `fałsz`, może być również inną tablicą.

Tworzenie tablicy

Aby utworzyć tablicę, przypisz wartości poszczególnym kluczom tablicy. Klucze tablicy są oznaczane za pomocą nawiasów kwadratowych, jak to zostało przedstawione na listingu 4.1.

Listing 4.1. Tworzenie tablic

```
// Tablica o nazwie $warzywa z kluczami łańcuchami.
$warzywa['kukurydza'] = 'żółty';
$warzywa['burak'] = 'czerwony';
$warzywa['marchewka'] = 'pomarańczowy';

// Tablica o nazwie $obiad z kluczami liczbowymi.
$obiad[0] = 'Słodka kukurydza ze szparagami';
$obiad[1] = 'Kurczak cytrynowy';
$obiad[2] = 'Duszone grzyby z bambusem';

// Tablica o nazwie $komputery z kluczami liczbowymi oraz łańcuchami.
$komputery['trs-80'] = 'Radio Shack';
$komputery[2600] = 'Atari';
$komputery['Adam'] = 'Coleco';
```

Klucze i wartości tablicy przedstawionej na listingu 4.1 są łańcuchami (na przykład `kukurydza`, `Duszone grzyby bambusa` czy `Coleco`) i liczbami (na przykład `0`, `1` czy `2600`). Zostały napisane podobnie jak inne łańcuchy i liczby w programach PHP: ze znakami cudzysłowów otaczającymi łańcuchy, ale nie liczby.

Tablicę możesz również utworzyć za pomocą służącej temu konstrukcji języka `array()`. Przykład z listingu 4.2 przedstawia tę samą tablicę, która została już zaprezentowana w przykładzie z listingu 4.1.

Listing 4.2. Tworzenie tablicy za pomocą konstrukcji array()

```
$warzywa = array('kukurydza' => 'żółty',
                 'burak' => 'czerwony',
                 'marchewka' => 'pomarańczowy');
```

[1] Wartości *skalarne* opisują dane, które posiadają pojedynczą wartość: liczbę, fragment tekstu, wartość `prawda` lub `fałsz`. Złożone typy danych, na przykład tablice przechowujące wiele wartości, nie są skalarne.

```
$obiad = array(0 => 'Słodka kukurydza ze szparagami',
               1 => 'Kurczak cytrynowy',
               2 => 'Duszone grzyby z bambusem');

$komputery = array('trs-80' => 'Radio Shack',
                   2600 => 'Atari',
                   'Adam' => 'Coleco');
```

Za pomocą konstrukcji `array()` określasz listę par klucz-wartość oddzieloną przecinkami. Natomiast klucz i wartość są rozdzielone przez `=>`. Składnia `array()` jest zwięzła, dlatego jest bardziej odpowiednia, gdy w danym momencie dodajesz do tablicy więcej niż jeden element. Składnia nawiasów kwadratowych jest lepsza, kiedy dodajesz elementy pojedynczo.

Wybór dobrej nazwy tablicy

W stosunku do nazw tablic obowiązują te same reguły, które mają zastosowanie do nazw zmiennych. Pierwszym znakiem tablicy musi być litera bądź cyfra, a pozostałe znaki w nazwie muszą być literami, cyframi lub znakami podkreślenia. Nazwy tablic i zmiennych skalarnych pochodzą z tej samej puli nazw, tak więc nie możesz posiadać jednocześnie tablicy `$warzywa` oraz zmiennej skalarnej o nazwie `$warzywa`. Jeżeli przypiszesz wartość tablicy do zmiennej skalarnej lub na odwrót, wówczas stara wartość zostanie usunięta, a zmienna otrzyma nowy typ. W przykładzie na listingu 4.3 zmienna `$warzywa` staje się skalarna, a zmienna `$owoce` staje się tablicą.

Listing 4.3. Tablice a kolizje skalarne

```
// Poniższy wiersz przekształca zmienną $warzywa w tablicę.
$warzywa['kukurydza'] = 'żółty';

// Usunięcie wszelkich śladów po "kukurydza" i "żółty" powoduje, że
// tablica $warzywa staje się skalarna.
$warzywa = 'wyborne';

// Poniższy wiersz powoduje, że tablica $owoce staje się skalarna.
$owoce = 283;

// Zmienia zmienną $owoce w tablicę i usuwa poprzednią wartość skalarną.
$owoce['potas'] = 'banan';
```

Na listingu 4.1 tablice `$warzywa` oraz `$komputery` przechowywały listę powiązań. Tablica `$warzywa` wiązała warzywa z kolorami, podczas gdy tablica `$komputery` wiązała nazwy komputerów i producentów. Jednakże w tablicy `$obiad` zatroszczyliśmy się o nazwy posiłków, które były wartościami tablicy. Klucze tablicy są po prostu liczbami, które odróżniają jeden element od drugiego.

Tworzenie tablic liczbowych

PHP daje możliwość wykorzystania pewnych skrótów do pracy z tablicami, których klucze to wyłącznie liczby. Jeżeli utworzysz tablicę za pomocą konstrukcji `array()`, określając jedynie listę wartości zamiast par klucz-wartość, interpreter PHP automatycznie przypisze każdej wartości klucz liczbowy. Klucze rozpoczynają się od `0` i zwiększają o `1` w przypadku każdego elementu. Kod przedstawiony na listingu 4.4 wykorzystuje tę technikę do utworzenia tablicy `$obiad`.

Listing 4.4. Tworzenie tablic liczbowych za pomocą konstrukcji array()

```
$obiad = array('Słodka kukurydza ze szparagami',
               'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
print "Chciałbym $obiad[0] oraz $obiad[1].";
```

Wykonanie kodu przedstawionego na listingu 4.4 spowoduje wyświetlenie:

```
Chciałbym Słodka kukurydza ze szparagami oraz Kurczak cytrynowy.
```

Wewnętrznie interpreter PHP traktuje jednakowo tablice z kluczami liczbowymi oraz tablice z kluczami łańcuchami (jak i również tablice z mieszanymi kluczami liczbowymi i łańcuchami). Z powodu podobieństwa do funkcji w innych językach programowania programiści często odwołują się do tablic z kluczami wyłącznie liczbowymi jako do tablic „liczbowych", „indeksowanych" lub „uporządkowanych" oraz do tablic z łańcuchami jako do tablic „asocjacyjnych". Innymi słowy, z *tablicą asocjacyjną* mamy do czynienia wtedy, gdy jej klucze są wyraźnie czymś innym niż pozycjami wartości wewnątrz tablicy.

W trakcie tworzenia przez Ciebie tablicy PHP automatycznie zwiększa o jednostkę liczbę dla kluczy tablicy oraz dodaje do niej elementy z pustymi nawiasami kwadratowymi, co zostało przedstawione na listingu 4.5.

Listing 4.5. Dodawanie elementów za pomocą składni []

```
// Utwórz tablicę $lunch zawierającą dwa elementy.
// Powoduje to ustawienie $lunch[0]
$lunch[] = 'Suszone grzyby w przyrumienionym sosie';
// Powoduje to ustawienie $lunch[1]
$lunch[] = 'Ananas i grzyb';

// Tworzy tablicę $obiad zawierającą trzy elementy.
$obiad = array('Słodka kukurydza ze szparagami', 'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
// Dodaj element na końcu tablicy $obiad.
// Powoduje to ustawienie $obiad[3]
$obiad[] = 'Skrzydełko na ostro';
```

Puste nawiasy kwadratowe dodają element do tablicy. Element posiada klucz liczbowy, który ma wartość o jeden większą niż dotychczasowy największy klucz obecny w tablicy. Jeżeli tablica jeszcze nie istnieje, puste nawiasy kwadratowe dodadzą element z kluczem `0`.

Tworzenie pierwszego elementu posiadającego klucz `0`, a nie `1` odbiega od sposobu myślenia zwykłego człowieka (różni się od logiki przyjętej przez programistów komputerowych), tak więc wymaga przypomnienia. Pierwszym elementem tablicy z kluczem liczbowym jest `0`, a nie `1`.

Poznawanie rozmiaru tablicy

Funkcja `count()` informuje o liczbie elementów w tablicy. Przykład przedstawiony na listingu 4.6 prezentuje użycie funkcji `count()`.

Listing 4.6. Poznawanie rozmiaru tablicy

```
$obiad = array('Słodka kukurydza ze szparagami',
               'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
```

```
$posilki = count($obiad);

print "Mamy $posilki potrawy na obiad.";
```

Wykonanie kodu z listingu 4.6 spowoduje wyświetlenie:

```
Mamy 3 potrawy na obiad.
```

Kiedy przekażesz pustą tablicę (to znaczy tablicę nie zawierającą elementów), funkcja zwróci `0`. W wyrażeniach testowych konstrukcji `if()` pusta tablica przyjmie wartość `fałsz`.

Wykonywanie pętli na tablicach

Jedną z najczęstszych czynności wykonywanych za pomocą tablic jest indywidualne rozpatrywanie każdego elementu w tablicy i jego przetwarzanie. Działanie to może obejmować włączenie elementu do wiersza tabeli HTML lub dodanie jego wartości do funkcjonującej całości.

Najłatwiejszym sposobem przechodzenia przez kolejne elementy tablicy jest zastosowanie funkcji `foreach()`. Konstrukcja `foreach()` pozwala na jednokrotne uruchomienie bloku kodu dla każdego elementu tablicy. Przykład z listingu 4.7 wykorzystuje funkcję `foreach()` do wyświetlenia tabeli HTML zawierającej każdy element tablicy.

Listing 4.7. Wykonywanie pętli za pomocą polecenia foreach()

```
$posilek = array('sniadanie' => 'Włoskie orzechy',
                 'lunch' => 'Orzechy i białe grzyby',
                 'przekaska' => 'Suszone morwy',
                 'obiad' => 'Bakłażan z sosem chili');
print "<table>\n";
foreach ($posilek as $klucz => $wartosc) {
    print "<tr><td>$klucz</td><td>$wartosc</td></tr>\n";
}
print '</table>';
```

Wykonanie kodu z listingu 4.7 spowoduje wyświetlenie następujących wierszy:

```
<table>
<tr><td>sniadanie</td><td>Włoskie orzechy </td></tr>
<tr><td>lunch</td><td>Orzechy i białe grzyby</td></tr>
<tr><td>przekaska</td><td>Suszone morwy </td></tr>
<tr><td>obiad</td><td>Bakłażan z sosem chili</td></tr>
</table>
```

Dla każdego elementu w tablicy `$posilek` konstrukcja `foreach()` kopiuje klucz elementu do zmiennej `$klucz` oraz wartość do zmiennej `$wartosc`. Następnie uruchamia blok kodu zawarty wewnątrz nawiasów klamrowych. Program z listingu 4.7 powoduje wyświetlenie zmiennych `$klucz` i `$wartosc` wraz z pewnym kodem HTML w wierszu tabeli. Dla klucza i jego wartości wewnątrz bloku kodu możesz użyć jakichkolwiek nazw zmiennych. Jednak jeśli nazwy zmiennych zostały użyte przed konstrukcją `foreach()`, zostaną nadpisane wartościami z tablicy.

Kiedy używasz konstrukcji `foreach()` do wyświetlenia danych w tabeli HTML, często chcesz zastosować naprzemienne kolory lub style dla każdego wiersza tabeli. Jest to bardzo łatwe do osiągnięcia, jeżeli przechowujesz wartości naprzemiennych kolorów w oddzielnej tablicy. Następnie przełącz zmienną pomiędzy `0` a `1` za każdym razem, gdy konstrukcja `foreach()` ma wyświetlić odpowiedni kolor. W przykładzie z listingu 4.8 wyborowi podlegają dwie wartości koloru w tablicy `$kolor_wiersza`.

Listing 4.8. Naprzemienne kolory wiersza tabeli

```
$kolor_wiersza = array('red','green');
$index_koloru = 0;
$posilek = array('sniadanie' => 'Włoskie orzechy',
                 'lunch' => 'Orzechy i białe grzyby',
                 'przekaska' => 'Suszone morwy',
                 'obiad' => 'Bakłażan z sosem chili');
print "<table>\n";
foreach ($posilek as $klucz => $wartosc) {
    print '<tr bgcolor="' . $kolor_wiersza[$index_koloru] . '">';
    print "<td>$klucz</td><td>$wartosc</td></tr>\n";
    // Poniższy wiersz przełącza $index_koloru między 0 a 1.
    $index_koloru = 1 - $index_koloru;
}
print '</table>';
```

Kod przedstawiony na listingu 4.8 spowoduje wyświetlenie następujących wierszy:

```
<table>
<tr bgcolor="red"><td>sniadanie</td><td>Włoskie orzechy</td></tr>
<tr bgcolor="green"><td>lunch</td><td>Orzechy i białe grzyby</td></tr>
<tr bgcolor="red"><td>przekaska</td><td>Suszone morwy</td></tr>
<tr bgcolor="green"><td>obiad</td><td>Bakłażan z sosem chili</td></tr>
</table>
```

Zmiana zmiennych pętli takich jak `$klucz` i `$wartosc` wewnątrz bloku kodu `foreach()` nie wpływa na rzeczywistą tablicę. Jeżeli chcesz zmienić tablicę, użyj zmiennej `$klucz` jako indeksu w tablicy. Technika ta została przedstawiona na listingu 4.9 do podwojenia ceny każdego elementu tablicy.

Listing 4.9. Modyfikacja tablicy za pomocą konstrukcji foreach()

```
$posilek = array('Włoskie orzechy' => 1,
                 'Orzechy i białe grzyby' => 4.95,
                 'Suszone morwy' => 3.00,
                 'Bakłażan z sosem chili' => 6.50);

foreach ($posilek as $potrawa => $cena) {
    // zapis $cena = $cena * 2 NIE działa.
    $posilek[$potrawa] = $posilek[$potrawa] * 2;
}

// Przejdź przez kolejne elementy tablicy i wyświetl zmienione wartości.
foreach ($posilek as $potrawa => $cena) {
    printf("Nowa cena potrawy %s to %.2f.\n",$potrawa,$cena);
}
```

Wykonanie kodu przedstawionego powyżej spowoduje wyświetlenie:

```
Nowa cena potrawy Włoskie orzechy to 2.00 zł.
Nowa cena potrawy Orzechy i białe grzyby to 9.90 zł.
Nowa cena potrawy Suszone morwy to 6.00 zł.
Nowa cena potrawy Bakłażan z sosem chili to 13.00 zł.
```

W przypadku konstrukcji `foreach()` występuje też bardziej zwięzła forma do zastosowania z tablicami liczbowymi. Jej użycie zostało przedstawione na listingu 4.10.

Listing 4.10. Użycie konstrukcji foreach() z tablicami liczbowymi

```
$obiad = array('Słodka kukurydza ze szparagami',
               'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
```

```
foreach ($posilek as $potrawa) {
    print "Możesz zjeść: $posilek\n";
}
```

Wykonanie kodu z listingu 4.10 spowoduje wyświetlenie:

```
Możesz zjeść: Słodka kukurydza ze szparagami
Możesz zjeść: Kurczak cytrynowy
Możesz zjeść: Duszone grzyby z bambusem
```

W tej formie konstrukcji `foreach()` po prostu określiłeś jedną nazwę zmiennej, a każdy element został skopiowany do tej zmiennej wewnątrz bloku kodu. Jednakże wewnątrz bloku kodu nie możesz uzyskać dostępu do klucza elementu.

Aby monitorować za pomocą konstrukcji `foreach()` swoją pozycję w tablicy, musisz użyć oddzielnej zmiennej, której wartość będziesz zwiększał o jeden przy każdym wykonaniu bloku kodu konstrukcji `foreach()`. Wykorzystując polecenie `for()`, otrzymasz dokładną pozycję w zmiennej pętli. Pętla `foreach()` wskaże Ci wartość każdego elementu tablicy, natomiast pętla `for()` poda Ci pozycję każdego elementu tablicy. Nie ma struktury pętli, która podawałaby jednocześnie i wartość każdego elementu, i jego położenie.

Zatem jeśli chcesz wiedzieć, na którym zatrzymałeś się elemencie w trakcie kolejnego przechodzenia przez tablicę liczbową, zamiast konstrukcji `foreach()` użyj polecenia `for()`. Twoja pętla `for()` powinna polegać na zmiennej pętli rozpoczynającej się od `0` i kończącej na wartości o jeden mniejszej od całkowitej liczby elementów tablicy. Ten mechanizm został przedstawiony na listingu 4.11.

Listing 4.11. Kolejne przechodzenie przez elementy tablicy liczbowej za pomocą pętli for()

```
$obiad = array('Słodka kukurydza ze szparagami',
               'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
for ($i = 0, $liczba_potraw = count($obiad); $i < $liczba_potraw; $i++) {
  print "Potrawą numer $i jest $obiad[$i]\n";
}
```

Wynikiem działania powyższego listingu będzie wyświetlenie następujących wierszy:

```
Potrawą numer 1 jest Słodka kukurydza ze szparagami
Potrawą numer 2 jest Kurczak cytrynowy
Potrawą numer 3 jest Duszone grzyby z bambusem
```

Podczas przechodzenia przez kolejne elementy tablicy za pomocą funkcji `for()` masz do dyspozycji działający licznik, dostępny dla elementu tablicy, w którym jesteś. Użyj tego licznika w połączeniu z operatorem współczynnika do uzyskania naprzemiennych kolorów wiersza tabeli. Odpowiedni kod został przedstawiony na listingu 4.12.

Listing 4.12. Naprzemienne kolory wiersza tabeli uzyskane za pomocą funkcji for()

```
$kolor_wiersza = array('red','green');
$obiad = array('Słodka kukurydza ze szparagami',
               'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
print "<table>\n";

for ($i = 0, $liczba_potraw = count($obiad); $i < $liczba_potraw; $i++) {
    print '<tr bgcolor="' . $kolor_wiersza[$i % 2] . '">';
    print "<td>Element $i</td><td>$obiad[$i]</td></tr>\n";
}
print '</table>';
```

Przykład z listingu 4.12 określa prawidłowy kolor wiersza tabeli za pomocą `$i % 2`. Wartość wynosi na przemian: `0` i `2`, a `$i` jest parzyste i nieparzyste. Nie ma potrzeby wykorzystania oddzielnej zmiennej, jak na przykład `$index_koloru` w listingu 4.8, do przechowywania odpowiedniego koloru wiersza. Wynikiem działania listingu 4.12 będzie:

```
<table>
<tr bgcolor="red"><td>Element 1</td><td>Słodka kukurydza ze szparagami</td></tr>
<tr bgcolor="green"><td>Element 2</td><td>Kurczak cytrynowy</td></tr>
<tr bgcolor="red"><td>Element 3</td><td>Duszone grzyby z bambusem</td></tr>
</table>
```

Kiedy przechodzisz przez kolejne elementy tablicy za pomocą konstrukcji `foreach()`, elementy stają się dostępne w kolejności, w jakiej zostały dodane do tablicy. Pierwszym dostępnym jest element dodany jako pierwszy, drugi dodany element jest kolejnym i tak dalej. Jeżeli posiadasz tablicę liczbową, której elementy zostały dodane w innej kolejności niż ich klucze są zwykle uporządkowane, może to spowodować powstanie nieoczekiwanych wyników. Program z listingu 4.13 nie wyświetli elementów tablicy w kolejności alfabetycznej czy liczbowej.

Listing 4.13. Kolejność elementów tablicy a funkcja foreach()

```
$litery[0] = 'A';
$litery[1] = 'B';
$litery[3] = 'D';
$litery[2] = 'C';

foreach ($litery as $litera) {
    print $litera;
}
```

Wykonanie powyższego programu spowoduje wyświetlenie:

```
ABDC
```

Aby zagwarantować dostęp do elementów w porządku liczbowym klucza, użyj polecenia `for()` do przejścia przez pętlę:

```
for ($i = 0, $liczba_liter = count($litery); $i < $liczba_liter; $i++) {
   print $litery[$i];
}
```

Wynikiem działania powyższego kodu będzie:

```
ABCD
```

Jeśli szukasz określonego elementu w tablicy, nie musisz przechodzić kolejno przez całą tablicę, aby go znaleźć. Są bardziej efektywne sposoby na odszukanie poszczególnego elementu. Aby sprawdzić element z określonym kluczem, użyj funkcji `array_key_exists()`, która została przedstawiona na listingu 4.14. Funkcja zwraca wartość `prawda`, jeżeli element z dostarczonym kluczem istnieje w tablicy.

Listing 4.14. Sprawdzanie elementu z określonym kluczem

```
$posilek = array('Włoskie orzechy' => 1,
                 'Orzechy i białe grzyby' => 4.95,
                 'Suszone morwy' => 3.00,
                 'Bakłażan z sosem chili' => 6.50,
                 'Krewetki' => 0); // Krewetki są za darmo!
$ksiazki = array("Przewodnik po chińskich znakach dla łasuchów",
                  'Jak gotować i jeść chińskie potrawy');
```

```
// To jest prawda.
if (array_key_exists('Krewetki',$posilek)) {
    print "Tak, posiadamy krewetki.";
}
// To jest falsz.
if (array_key_exists('Kanapka ze stekiem',$posilek)) {
    print "Posiadamy kanapkę ze stekiem.";
}
// To jest prawda.
if (array_key_exists(1, $ksiazki)) {
    print "Element 1 to książka Jak gotować i jeść chińskie potrawy";
}
```

Aby sprawdzić element o określonej wartości, użyj funkcji `in_array()`, jak to zostało przedstawione na listingu 4.15.

Listing 4.15. Sprawdzenie elementu z określoną wartością

```
$posilek = array('Włoskie orzechy' => 1,
              'Orzechy i białe grzyby' => 4.95,
              'Suszone morwy' => 3.00,
              'Bakłażan z sosem chili' => 6.50,
              'Krewetki' => 0);
$ksiazki = array("Przewodnik po chińskich znakach dla łasuchów",
               'Jak gotować i jeść chińskie potrawy');

// To jest prawda: klucz Suszone morwy posiada wartość 3.00.
if (in_array(3, $posilek)) {
  print 'Cena wynosi 3 zł za sztuke.';
}
// To jest prawda.
if (in_array('Jak gotować i jeść chińskie potrawy', $ksiazki)) {
  print "Posiadamy książkę Jak gotować i jeść chińskie potrawy.";
}
// To jest falsz: funkcja in_array() jest wrażliwa na wielkość liter.
if (in_array("przewodnik po chińskich znakach dla łasuchów", $books)) {
  print "Posiadamy książkę Przewodnik po chińskich znakach dla łasuchów.";
}
```

Funkcja `in_array()` zwraca wartość `prawda`, jeśli znajdzie element o zadanej wartości. W przypadku tej funkcji wielkość liter w trakcie porównywania łańcuchów ma znaczenie. Funkcja `array_search()` jest podobna do funkcji `in_array()`, różni się jednak od niej tym, że jeśli znajdzie szukany element, to zwraca klucz elementu zamiast wartości `prawda`. W przykładzie na listingu 4.16 funkcja `array_search()` zwraca nazwę potrawy, która kosztuje 6,50 zł.

Listing 4.16. Znajdowanie elementu o określonej wartości

```
$posilek = array('Włoskie orzechy' => 1,
              'Orzechy i białe grzyby' => 4.95,
              'Suszone morwy' => 3.00,
              'Bakłażan z sosem chili' => 6.50,
              'Krewetki' => 0);

$potrawa = array_search(6.50, $posilek);
if ($potrawa) {
    print "$potrawa kosztuje 6.50 zł.";
}
```

Wynikiem działania kodu z listingu 4.16 będzie wyświetlenie wiersza:

```
Bakłażan z sosem chili kosztuje 6.50 zł.
```

Modyfikacja tablic

Możesz operować na indywidualnych elementach tablicy, podobnie jak na zmiennych skalarnych, używając do tego operatorów arytmetycznych, logicznych i innych. Kod na listingu 4.17 przedstawia pewne operacje na elementach tablicy.

Listing 4.17. Operacje na elementach tablicy

```
$potrawy['Wołowina Chow Foon'] = 12;
$potrawy['Wołowina Chow Foon']++;
$potrawy['Pieczona kaczka'] = 3;

$potrawy['total'] = $potrawy['Wołowina Chow Foon'] + $potrawy['Pieczona kaczka'];

if ($potrawy['total'] > 15) {
    print "Zjadłeś za dużo: ";
}

print 'Zjadłeś ' . $potrawy['Wołowina Chow Foon'] . ' dania Wołowina Chow Foon.';
```

Kod z listingu 4.17 wyświetli:

```
Zjadłeś za dużo: Zjadłeś 13 dania Wołowina Chow Foon.
```

Wstawianie wartości elementu tablicy w łańcuch otoczony znakami podwójnego cudzysłowu lub dokumenty typu here document jest podobne do wstawiania liczb lub znaków. Najłatwiejszym sposobem jest zawarcie elementu tablicy w łańcuchu bez umieszczania znaków cudzysłowu wokół klucza elementu. Zostało to przedstawione na listingu 4.18.

Listing 4.18. Wstawianie wartości elementu tablicy w łańcuch otoczony znakami podwójnego cudzysłowu

```
$posilki['sniadanie'] = 'Włoskie orzechy';
$posilki['lunch'] = 'Bakłażan z sosem chili';
$ilosci = array(3, 6);

print "Na śniadanie chciałbym zjeść $posilki[sniadanie], a na lunch ";
print "chciałbym $posilki[lunch]. Poproszę $ilosci[0] na śniadanie oraz ";
print "$ilosci[1] na lunch.";
```

Wykonanie programu z listingu 4.18 spowoduje wyświetlenie:

```
Na śniadanie chciałbym zjeść Włoskie orzechy, a na lunch
chciałbym Bakłażan z sosem chili. Poproszę 3 na śniadanie oraz
6 na lunch.
```

Mechanizm wstawiania widoczny na listingu 4.18 funkcjonuje jedynie z kluczami tablicy, które składają się wyłącznie z liter, liczb oraz znaków podkreślenia. Jeżeli posiadasz klucz tablicy zawierający znaki odstępu bądź inne znaki interpunkcyjne, wstaw je w nawiasy klamrowe, jak to zostało przedstawione na listingu 4.19.

Listing 4.19. Wstawianie wartości elementu tablicy za pomocą nawiasów klamrowych

```
$posilki['Włoskie orzechy'] = '3.95 zł';
$wezly['www.przyklad.com'] = 'witryna internetowa';

print "Włoskie orzechy kosztują {$posilki['Włoskie orzechy']}.";
print "www.przyklad.com to jest {$wezly['www.przyklad.com']}.";
```

W wyniku działania powyższego programu otrzymamy:

```
Włoskie orzechy kosztują 3.95 zł.
www.przyklad.com to jest witryna internetowa.
```

W łańcuchach otoczonych znakami podwójnego cudzysłowu lub dokumentach typu here document jest obliczane wyrażenie zawarte wewnątrz nawiasów klamrowych, a następnie jego wartość zostaje umieszczona w łańcuchu. W listingu 4.19 użytymi wyrażeniami są samodzielne elementy tablicy, tak więc wartości elementów zostają włączone do łańcuchów.

Do usunięcia elementu z tablicy służy funkcja `unset()`:

```
unset($posilki['Pieczona kaczka']);
```

Usunięcie elementu za pomocą funkcji `unset()` nie jest po prostu ustawieniem elementowi wartości `0` lub pustego łańcucha. Kiedy używasz `unset()` element nie jest dłużej brany pod uwagę w trakcie przechodzenia przez tablicę lub obliczania liczby elementów w tablicy. Użycie funkcji `unset()` na tablicy przedstawiającej spis produktów sprzedawanych w sklepie jest podobne do stwierdzenia, że sklep nie będzie dłużej oferował tego produktu. Ustawienie wartości elementu na `0` lub jako pusty łańcuch oznacza, że danego produktu tymczasowo nie ma w sklepie.

Jeśli będziesz chciał od razu wyświetlić wszystkie wartości w tablicy, najszybciej osiągniesz to, używając funkcji `implode()`. Tworzy ona łańcuch przez połączenie wszystkich wartości w tablicy, które rozdziela separatorem łańcucha. Kod z listingu 4.20 wyświetla listę potraw, których nazwy są oddzielone przecinkami.

Listing 4.20. Utworzenie łańcucha z tablicy za pomocą funkcji implode()

```
$potrawy = array('Pieczony kurczak','Faszerowana kaczka','Ciasto dyniowe');
$menu = implode(', ', $potrawy);
print $menu;
```

Wykonanie programu z listingu 4.20 spowoduje wyświetlenie:

```
Pieczony kurczak, Faszerowana kaczka, Ciasto dyniowe
```

Aby rozbić tablicę bez znaków rozdzielających, jako pierwszego argumentu funkcji `implode()` użyj pustego łańcucha:

```
$litery = array('A','B','C','D');
print implode('',$litery);
```

Powyższy kod spowoduje wyświetlenie:

```
ABCD
```

Użyj funkcji `implode()`, aby uprościć wyświetlanie wierszy tabeli HTML, jak to zostało przedstawione na listingu 4.21.

Listing 4.21. Wyświetlanie tabeli HTML za pomocą funkcji implode()

```
$potrawy = array('Pieczony kurczak','Faszerowana kaczka','Ciasto dyniowe');
print '<tr><td>' . implode('</td><td>',$potrawy) . '</td></tr>';
```

Uruchomiony program z listingu 4.21 wyświetli:

```
<tr><td>Pieczony kurczak</td><td>Faszerowana kaczka</td><td>Ciasto dyniowe</td></tr>
```

Funkcja `implode()` umieszcza swoje separatory między każdą wartością, a więc do utworzenia kompletnego wiersza tabeli będziesz musiał również wyświetlić przed pierwszym elementem znacznik otwierający wiersz oraz za ostatnim elementem znacznik zamykający.

Odwrotnością funkcji `implode()` jest funkcja `explode()`. Powoduje ona rozbicie łańcucha i przekształcenie go do postaci tablicy. Argumentem separatora dla funkcji `explode()` jest łańcuch, który powinien szukać elementów rozdzielających tablicę. Przykład użycia funkcji `explode()` został zamieszczony na listingu 4.22.

Listing 4.22. Zamiana łańcucha na tablicę za pomocą funkcji explode()

```
$ryby = 'Okoń, Karp, Szczupak, Flądra';
$lista_ryb = explode(', ', $ryby);
print "Drugą rybą z listy jest $lista_ryb[1]";
```

Wykonanie programu z listingu 4.22 spowoduje wyświetlenie:

```
Drugą rybą z listy jest Karp.
```

Sortowanie tablic

Mamy kilka sposobów na sortowanie tablic. Użycie konkretnej funkcji zależy od tego, jak chcesz posortować swoją tablicę oraz jakiego jest ona rodzaju.

Funkcja `sort()` sortuje tablice według wartości elementów. Powinna być używana jedynie na tablicach liczbowych, ponieważ w trakcie sortowania resetuje klucze. Przykład z listingu 4.23 przedstawia pewne tablice przed sortowaniem i po nim.

Listing 4.23. Sortowanie za pomocą funkcji sort()

```
$obiad = array('Słodka kukurydza ze szparagami',
               'Kurczak cytrynowy',
               'Duszone grzyby z bambusem');
$posilek = array('sniadanie' => 'Włoskie orzechy',
              'lunch' => 'Orzechy i białe grzyby',
              'przekaska' => 'Suszone morwy',
              'obiad' => 'Bakłażan z sosem chili');

print "Przed sortowaniem:\n";
foreach ($obiad as $klucz => $wartosc) {
    print " \$obiad: $klucz $wartosc\n";
}
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}

sort($obiad);
sort($posilek);

print "Po sortowaniu:\n";
foreach ($obiad as $klucz => $wartosc) {
    print " \$obiad: $klucz $wartosc\n";
}
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}
```

Uruchomiony program z listingu 4.23 wyświetli nam:

```
Przed sortowaniem:
$obiad: 0 Słodka kukurydza ze szparagami
$obiad: 1 Kurczak cytrynowy
$obiad: 2 Duszone grzyby z bambusem
$posilek: sniadanie Włoskie orzechy
```

```
$posilek: lunch Orzechy i białe grzyby
$posilek: przekaska Suszone morwy
$posilek: obiad Bakłażan z sosem chili
Po sortowaniu:
$obiad: 0 Duszone grzyby z bambusem
$obiad: 1 Kurczak cytrynowy
$obiad: 2 Słodka kukurydza ze szparagami
$posilek: 0 Bakłażan z sosem chili
$posilek: 1 Orzechy i białe grzyby
$posilek: 2 Suszone morwy
$posilek: 3 Włoskie orzechy
```

Obie tablice zostały przeorganizowane w kolejności rosnącej wartości elementu. Pierwszą wartością w tablicy `$obiad` jest teraz `Duszone grzyby z bambusem`, natomiast w tablicy `$posilek` pierwszą wartością jest `Bakłażan z sosem chili`. Klucze w tablicy `$obiad` nie zostały zmienione, ponieważ była ona tablicą liczbową, zanim została posortowana. Jednakże klucze w tablicy `$posilek` zostały zastąpione przez liczby od `0` do `3`.

W celu posortowania tablicy asocjacyjnej użyj funkcji `asort()`, która trzyma klucze razem z ich wartościami. Kod z listingu 4.24 przedstawia tablicę `$posilek` z listingu 4.23 posortowaną za pomocą funkcji `asort()`.

Listing 4.24. Sortowanie za pomocą funkcji asort()

```
$posilek = array('sniadanie' => 'Włoskie orzechy',
                 'lunch' => 'Orzechy i białe grzyby',
                 'przekaska' => 'Suszone morwy',
                 'obiad' => 'Bakłażan z sosem chili');

print "Przed sortowaniem:\n";
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}

asort($posilek);

print "Po sortowaniu:\n";
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}
```

Wykonanie powyższego programu spowoduje wyświetlenie:

```
Przed sortowaniem:
$posilek: sniadanie Włoskie orzechy
$posilek: lunch Orzechy i białe grzyby
$posilek: przekaska Suszone morwy
$posilek: obiad Bakłażan z sosem chili
Po sortowaniu:
$posilek: obiad Bakłażan z sosem chili
$posilek: lunch Orzechy i białe grzyby
$posilek: przekaska Suszone morwy
$posilek: sniadanie Włoskie orzechy
```

Przy zastosowaniu funkcji `asort()` wartości zostały posortowane w ten sam sposób, jak to miało miejsce w przypadku użycia funkcji `sort()` ale tym razem klucze znajdują się obok siebie.

Podczas gdy funkcje `sort()` oraz `asort()` sortują tablice według wartości elementu, to dzięki funkcji `ksort()` możesz również posortować tablicę według klucza. Utrzymuje ona razem pary klucz-wartość, ale ustawia kolejność par według klucza. Na listingu 4.25 tablica `$posilek` została posortowana za pomocą funkcji `ksort()`.

Listing 4.25. Sortowanie za pomocą funkcji ksort()

```
$posilek = array('sniadanie' => 'Włoskie orzechy',
              'lunch' => 'Orzechy i białe grzyby',
              'przekaska' => 'Suszone morwy',
              'obiad' => 'Bakłażan z sosem chili');

print "Przed sortowaniem:\n";
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}

ksort($posilek);

print "Po sortowaniu:\n";
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}
```

W wyniku działania kodu z listingu 4.25 otrzymamy:

```
Przed sortowaniem:
$posilek: sniadanie Włoskie orzechy
$posilek: lunch Orzechy i białe grzyby
$posilek: przekaska Suszone morwy
$posilek: obiad Bakłażan z sosem chili
Po sortowaniu:
$posilek: lunch Orzechy i białe grzyby
$posilek: obiad Bakłażan z sosem chili
$posilek: przekaska Suszone morwy
$posilek: sniadanie Włoskie orzechy
```

Tablica zostaje przeorganizowana, a klucze są teraz uporządkowane rosnąco w kolejności alfabetycznej. Każdy element pozostaje niezmieniony; także wartość przypisana do klucza przed sortowaniem jest po sortowaniu taka sama dla każdego klucza. Jeżeli korzystając z funkcji `ksort()`, sortujesz tablicę liczbową, wówczas elementy przyjmują taki układ, że klucze zostają uporządkowane w rosnącej kolejności liczbowej. Jest to ta sama kolejność, od której rozpoczynasz podczas tworzenia tablicy za pomocą konstrukcji `array()` bądź `[]`.

Funkcje sortujące tablice `sort()`, `asort()` oraz `ksort()` mają swoje odpowiedniki sortujące w kolejności malejącej. Tymi funkcjami są odpowiednio `rsort()`, `arsort()` oraz `krsort()`. Mechanizm działania tych funkcji jest identyczny z zasadą działania wspomnianych wcześniej funkcji `sort()`, `asort()` i `ksort()`, z tą tylko różnicą, że sortują tablice od największego (lub alfabetycznie ostatniego) klucza lub wartości, a ich kolejne elementy są układane w kolejności malejącej. Przykład z listingu 4.26 przedstawia w działaniu funkcję `arsort()`.

Listing 4.26. Sortowanie za pomocą funkcji arsort()

```
$posilek = array('sniadanie' => 'Włoskie orzechy',
              'lunch' => 'Orzechy i białe grzyby',
              'przekaska' => 'Suszone morwy',
              'obiad' => 'Bakłażan z sosem chili');

print "Przed sortowaniem:\n";
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}

arsort($posilek);

print "Po sortowaniu:\n";
```

```
foreach ($posilek as $klucz => $wartosc) {
    print "   \$posilek: $klucz $wartosc\n";
}
```

Efektem działania powyższego kodu jest wyświetlenie następujących wierszy:

```
Przed sortowaniem:
$posilek: sniadanie Włoskie orzechy
$posilek: lunch Orzechy i białe grzyby
$posilek: przekaska Suszone morwy
$posilek: obiad Bakłażan z sosem chili
Po sortowaniu:
$posilek: sniadanie Włoskie orzechy
$posilek: przekaska Suszone morwy
$posilek: lunch Orzechy i białe grzyby
$posilek: obiad Bakłażan z sosem chili
```

Używanie tablic wielowymiarowych

Jak już zostało to wspomniane we wcześniejszym podrozdziale, „Podstawy tablic", wartość elementu tablicy może być inną tablicą. Taka możliwość jest użyteczna, gdy zachodzi potrzeba przechowywania danych, które będą bardziej skomplikowaną strukturą, a nie po prostu kluczem i pojedynczą wartością. Standardowa para klucz-wartość jest wystarczająca do porównania nazwy posiłku (na przykład `sniadanie` lub `lunch`) z pojedynczą potrawą (na przykład `Włoskie orzechy` lub `Kurczak z orzechami`). Co się jednak stanie, kiedy każdy posiłek będzie się składał z niejednej potrawy? Wówczas wartości elementów powinny być tablicami, a nie łańcuchami.

Użycie konstrukcji `array()` do utworzenia tablic, które zawierają więcej tablic jako wartości elementów, zostało przedstawione na listingu 4.27.

Listing 4.27. Utworzenie tablic wielowymiarowych za pomocą konstrukcji array()

```
$posilki = array('sniadanie' => array('Włoskie orzechy','Kawa'),
                 'lunch' => array('Orzechy','Białe grzyby'),
                 'przekaska' => array('Suszone morwy','Solony krab z sezamem'));

$lunche = array( array('Kurczak','Bakłażan','Ryż'),
                  array('Wołowina','Szarlotka','Makaron'),
                  array('Bakłażan','Tofu'));

$przyprawy = array('Japonska' => array('gorące' => 'wasabi',
                                       'słone' => 'sos sojowy'),
                   'Chinska'  => array('gorące' => 'musztarda',
                                       'pikantne' => 'jesion'));
```

Dostęp do elementów w tych tablicach tablic jest uzyskiwany przez użycie większej ilości nawiasów kwadratowych identyfikujących elementy. Każdy zbiór nawiasów kwadratowych określa jeden poziom w całej tablicy. Kod na listingu 4.28 prezentuje, w jaki sposób uzyskać dostęp do elementów tablic zdefiniowanych na listingu 4.27.

Listing 4.28. Dostęp do elementów tablic wielowymiarowych

```
print $posilki['lunch'][1];                 // Białe grzyby
print $posilki['przekaska'][0];             // Suszone morwy
print $lunche[0][0];                        // Kurczak
print $lunche[2][1];                        // Tofu
print $przyprawy['Japonska']['słone'];      // sos sojowy
print $przyprawy['Chinska']['gorące'];      // musztarda
```

Każdy poziom tablicy jest nazwany *wymiarem*. Wszystkie tablice umieszczone w rozdziale przed tym podrozdziałem były *tablicami jednowymiarowymi*. Każda z nich posiadała jeden poziom kluczy. Tablice takie jak `$posilki`, `$lunche` i `$przekaski` przedstawione w listingu 4.28 są nazwane *tablicami wielowymiarowymi*, ponieważ każda z nich posiada więcej niż jeden wymiar.

Tablice wielowymiarowe możesz również utworzyć i modyfikować za pomocą składni nawiasów kwadratowych. Na listingu 4.29 zostały przedstawione pewne modyfikacje przeprowadzane na tablicach wielowymiarowych.

Listing 4.29. Manipulacje tablicami wielowymiarowymi

```
$ceny['obiad']['Słodka kukurydza ze szparagami'] = 12.50;
$ceny['lunch']['Orzechy i białe grzyby'] = 4.95;
$ceny['obiad']['Duszony bambus z grzybami'] = 8.95;

$ceny['obiad']['ogolem'] = $ceny['obiad']['Słodka kukurydza ze szparagami'] +
                           $ceny['obiad']['Duszony bambus z grzybami'];

$specjalnosci[0][0] = 'Kasztanowe bułeczki';
$specjalnosci[0][1] = 'Włoskie orzechy';
$specjalnosci[0][2] = 'Orzeszki zmiemne';
$specjalnosci[1][0] = 'Sałatka';
$specjalnosci[1][1] = 'Włoska sałatka';
// Pomijając dodanie indeksu na końcu tablicy,
// tworzymy $specjalnosci[1][2]
$specjalnosci[1][] = 'Sałatka z orzeszków ziemnych';
```

Do kolejnego przechodzenia przez każdy wymiar tablicy wielowymiarowej użyj zagnieżdżonej pętli `foreach()` lub `for()`. W kodzie z listingu 4.30 do kolejnego przechodzenia przez wielowymiarową tablicę asocjacyjną została wykorzystana konstrukcja `foreach()`.

Listing 4.30. Przejście za pomocą foreach() przez kolejne wymiary tablicy wielowymiarowej

```
$przyprawy = array('Japonska' => array('gorąca' => 'wasabi',
                                       'słona' => 'sos sojowy'),
                   'Chinska'  => array('gorąca' => 'musztarda',
                                       'pikantna' => 'jesion'));

// Zmienna $kultura jest kluczem, a zmienna $przyprawy_kultury to wartość (tablicy).
foreach ($przyprawy as $kultura => $przyprawy_kultury) {

    // $przyprawa jest kluczem, a $przyklad jest wartością.
    foreach ($przyprawy_kultury as $przyprawa => $przyklad) {
        print "$culture przyprawą $flavor jest $example.\n";
    }
}
```

Uruchomienie programu z listingu 4.30 spowoduje wyświetlenie:

```
Japonska gorąca przyprawą jest wasabi.
Japonska słona przyprawą jest sos sojowy.
Chinska gorąca przyprawą jest musztarda.
Chinska pikantna przyprawą jest jesion.
```

Pierwsza pętla `foreach()` na listingu 4.30 powoduje przejście przez pierwszy wymiar tablicy `$przyprawy`. Klucze przechowywane w zmiennej `$kultura` są łańcuchami `Japonska` i `Chinska`. Natomiast wartości przechowywane przez zmienną `$przyprawy_kultury` są tablicami będącymi wartością elementów tego wymiaru. Kolejne przejście pętli `foreach()` przez te tablice

wartości elementów kopiuje klucze takie jak `gorąca` i `słona` do tablicy `$przyprawy`, a wartości, na przykład `wasabi` i `sos sojowy`, do zmiennej `$przyklad`. Blok kodu należący do drugiej pętli `foreach()` używa zmiennych z obu poleceń `foreach()` do wyświetlenia kompletnego komunikatu.

Podobnie jak zagnieżdżone pętle `foreach()` przechodzące kolejno przez wielowymiarową tablicę asocjacyjną, zagnieżdżone pętle `for()` przechodzą kolejno przez wielowymiarowe tablice liczbowe, jak to zostało przedstawione na listingu 4.31.

Listing 4.31. Przejście za pomocą for() przez kolejne wymiary tablicy wielowymiarowej

```
$specjalnosci = array( array('Kasztanowe bułeczki', 'Włoskie orzechy', 'Orzeszki
ziemne'),
                array('Sałatka','Włoska sałatka', 'Sałatka z orzeszków ziemnych') );

// Zmienna $liczba_specjalnosci wynosi 2: jest to liczba elementów w pierwszym wymiarze
// tablicy $specjalnosci.
for ($i = 0, $liczba_specjalnosci = count($specjalnosci); $i < $liczba_specjalnosci;
$i++) {
    // Zmienna $liczba_podelementow wynosi 3: jest to liczba elementów w każdej podtablicy.
    for ($m = 0, $liczba_podelementow = count($specjalnosci[$i]); $m <
$liczba_podelementow; $m++) {
        print "Element [$i][$m] to " . $specjalnosci[$i][$m] . "\n";
    }
}
```

Wykonanie powyższego programu spowoduje wyświetlenie:

```
Element [0][0] to Kasztanowe bułeczki
Element [0][1] to Włoskie orzechy
Element [0][2] to Orzeszki ziemne
Element [1][0] to Sałatka
Element [1][1] to Włoska sałatka
Element [1][2] to Sałatka z orzeszków ziemnych
```

W listingu 4.31 dane wyjściowe pętli `for()` przechodzą przez dwa elementy tablicy `$specjalnosci`. Wewnętrzna pętla `for()` przechodzi przez każdy element podtablic, które przechowują odmienne łańcuchy. W poleceniu wyświetlającym `print` argument `$i` jest indeksem w pierwszym wymiarze (elementy tablicy `$specjalnosci`), natomiast `$m` jest indeksem w drugim wymiarze (podtablicy).

W celu umieszczania wartości z tablicy wielowymiarowej w łańcuchu otoczonym znakami podwójnego cudzysłowu lub dokumencie typu here document użyj składni nawiasów klamrowych przedstawionej na listingu 4.19. W kodzie z listingu 4.32 zostały użyte nawiasy klamrowe do interpolacji w celu uzyskania takich samych danych wyjściowych, jak w przypadku listingu 4.31. W rzeczywistości jedynym innym wierszem w listingu 4.32 jest ten, zawierający polecenie `print`.

Listing 4.32. Interpolacja elementu tablicy wielowymiarowej

```
$specjalnosci = array( array('Kasztanowe bułeczki', 'Włoskie orzechy', 'Orzeszki
ziemne'),
                array('Sałatka','Włoska sałatka', 'Sałatka z orzeszków ziemnych') );

// Zmienna $liczba_specjalnosci wynosi 2: jest to liczba elementów w pierwszym wymiarze
// tablicy $specjalnosci.
for ($i = 0, $liczba_specjalnosci = count($specjalnosci); $i < $liczba_specjalnosci;
$i++) {
```

```
    // Zmienna $liczba_podelementow wynosi 3: jest to liczba elementów w każdej podtablicy.
    for ($m = 0, $liczba_podelementow = count($specjalnosci[$i]); $m <
$liczba_podelementow; $m++) {
        print "Element [$i][$m] to {$specjalnosci[$i][$m]}\n";
    }
}
```

Podsumowanie rozdziału

W rozdziale czwartym zostały przedstawione następujące zagadnienia:

- Podstawowe komponenty tablicy: elementy, klucze i wartości.
- Dwa sposoby definiowania tablicy w programach: za pomocą konstrukcji `array()` oraz przy użyciu nawiasów kwadratowych.
- Pojęcie skrótów dostarczanych przez PHP dla tablic z kluczami liczbowymi.
- Zliczanie liczby elementów w tablicy.
- Przejście przez każdy element tablicy za pomocą funkcji `foreach()`.
- Naprzemienne kolory wierszy tabeli — zastosowanie funkcji `foreach()` oraz tablicy wartości kolorów.
- Modyfikacja wartości elementu tablicy wewnątrz bloku kodu `foreach()`.
- Odwiedzanie każdego elementu tablicy liczbowej za pomocą funkcji `for()`.
- Naprzemienne kolory wierszy tabeli — zastosowanie funkcji `foreach()` oraz operatora współczynnika (`%`).
- Zrozumienie kolejności, w której konstrukcje `foreach()` i `for()` przechodzą przez elementy tablicy.
- Sprawdzanie elementu tablicy z określonym kluczem.
- Sprawdzanie elementu tablicy z określoną wartością.
- Interpolacja wartości elementu tablicy do łańcucha.
- Usuwanie elementu z tablicy.
- Generowanie łańcuchów z tablicy za pomocą funkcji `implode()`.
- Generowanie tablicy z łańcucha za pomocą funkcji `explode()`.
- Sortowanie tablicy przy użyciu funkcji `sort()`, `asort()` lub `ksort()`.
- Odwrotne sortowanie tablicy.
- Definiowanie tablicy wielowymiarowej.
- Dostęp do poszczególnych elementów tablicy wielowymiarowej.
- Przejście do każdego elementu w tablicy wielowymiarowej za pomocą funkcji `foreach()` bądź `for()`.
- Interpolacja elementów tablicy wielowymiarowej do łańcucha.

Ćwiczenia

1. Na podstawie danych zebranych przez amerykański urząd statystyczny (Biura ds. Spisu Ludności) 10 największych miast amerykańskich (według wielkości populacji) w 2000 roku to:
   - Nowy Jork, NY (8 008 278 osób)
   - Los Angeles, CA (3 694 820 osób)
   - Chicago, IL (2 896 016 osób)
   - Houston, TX (1 953 631 osób)
   - Filadelfia, PA (1 517 550 osób)
   - Phoenix, AZ (1 321 045 osób)
   - San Diego, CA (1 223 400 osób)
   - Dallas, TX (1 188 580 osób)
   - San Antonio, TX (1 144 646 osób)
   - Detroit, MI (951 270 osób)

   Zdefiniuj tablicę (lub tablice), które będą przechowywać te informacje o położeniu i wielkości populacji. Wyświetl tabelę położenia oraz informacji o wielkości populacji, która będzie zawierać ogólną wielkość populacji we wszystkich dziesięciu miastach.
2. Zmodyfikuj swój program z poprzedniego ćwiczenia tak, aby wiersze w tabeli wynikowej były ułożone według wielkości populacji. Następnie zmodyfikuj program w taki sposób, aby wiersze zostały ułożone w kolejności według nazw miast.
3. Ponownie dokonaj modyfikacji programu z ćwiczenia pierwszego w taki sposób, aby tabela zawierała również wiersze przechowujące ogólną wielkość populacji dla każdego stanu przedstawionego na liście miast.
4. Odnosząc się do każdego przedstawionego poniżej rodzaju informacji, określ, w jaki sposób przechowywałbyś te informacje w tablicy, a następnie podaj przykładowy kod, który tworzy taką tablicę z kilkoma elementami. Na przykład w przypadku pierwszego elementu mógłbyś powiedzieć: „Tablica asocjacyjna, której kluczem jest nazwisko studenta, natomiast wartością w tablicy asocjacyjnej jest ocena oraz numer identyfikacyjny” (patrz poniżej):

   ```
   $uczniowie = array('Jan Kowalski' => array('ocena' => '5+', 'id' => 271231),
                      ('Aneta Nowak' => array('ocena' => '5', 'id' => 818211));
   ```

   a. Oceny i numery identyfikacyjne uczniów w klasie.

   b. Jak wiele sztuk każdego towaru jest przechowywanych w magazynie.

   c. Spis obiadów w szkole na cały tydzień — różne rodzaje każdego z posiłków (przystawka, danie główne, napój itd.) oraz koszt na każdy dzień.

   d. Imiona osób w Twojej rodzinie.

   e. Imiona, wiek oraz powiązania z Tobą osób z Twojej rodziny.