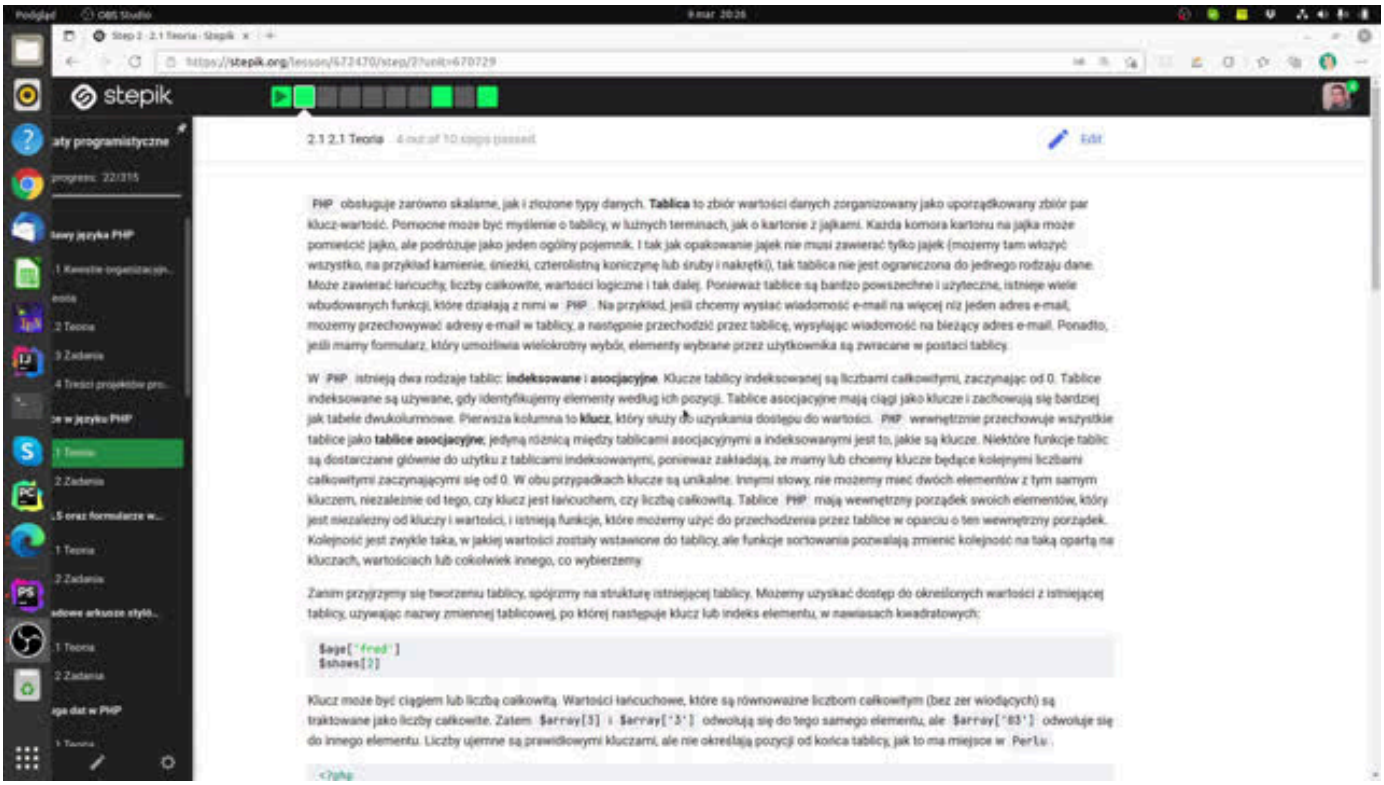
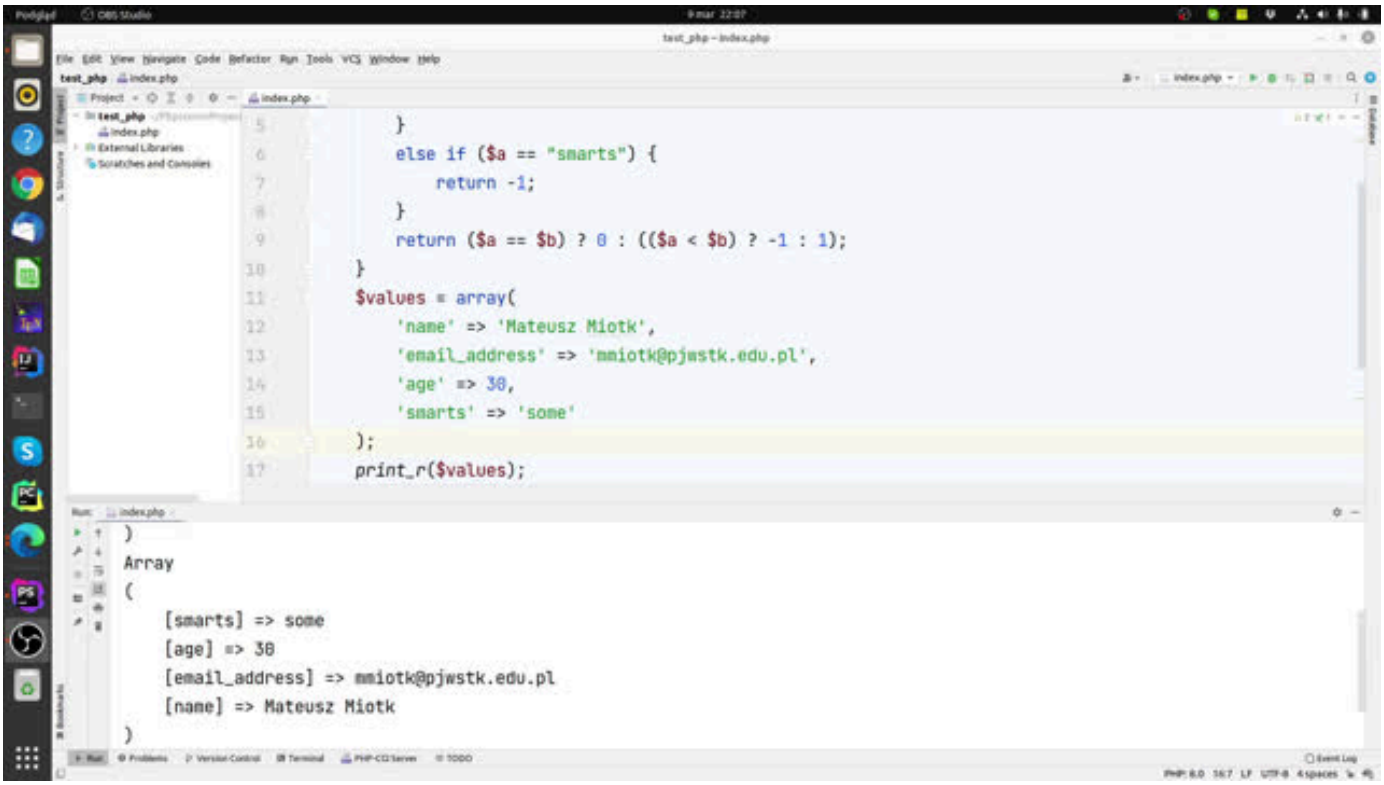
# 8.1 8.1 Teoria

## Step 1




**To watch this video please visit https://stepik.org/lesson/672470/step/1**

## Step 2




**To watch this video please visit https://stepik.org/lesson/672470/step/2**

## Step 3

`PHP` obsługuje zarówno skalarne, jak i złożone typy danych. **Tablica** to zbiór wartości danych zorganizowany jako uporządkowany zbiór par klucz-wartość. Pomocne może być myślenie o tablicy, w luźnych terminach, jak o kartonie z jajkami. Każda komora kartonu na jajka może pomieścić jajko, ale podróżuje jako jeden ogólny pojemnik. I tak jak opakowanie jajek nie musi zawierać tylko jajek (możemy tam włożyć wszystko, na przykład kamienie, śnieżki, czterolistną koniczynę lub śruby i nakrętki), tak tablica nie jest ograniczona do jednego rodzaju dane. Może zawierać łańcuchy, liczby całkowite, wartości logiczne i tak dalej. Ponieważ tablice są bardzo powszechne i użyteczne, istnieje wiele wbudowanych funkcji, które działają z nimi w `PHP`. Na przykład, jeśli chcemy wysłać wiadomość e-mail na więcej niż jeden adres e-mail, możemy przechowywać adresy e-mail w tablicy, a następnie przechodzić przez tablicę, wysyłając wiadomość na bieżący adres e-mail. Ponadto, jeśli mamy formularz, który umożliwia wielokrotny wybór, elementy wybrane przez użytkownika są zwracane w postaci tablicy.

W `PHP` istnieją dwa rodzaje tablic: **indeksowane** i **asocjacyjne**. Klucze tablicy indeksowanej są liczbami całkowitymi, zaczynając od 0. Tablice indeksowane są używane, gdy identyfikujemy elementy według ich pozycji. Tablice asocjacyjne mają ciągi jako klucze i zachowują się bardziej jak tabele dwukolumnowe. Pierwsza kolumna to **klucz**, który służy do uzyskania dostępu do wartości. `PHP` wewnętrznie przechowuje wszystkie tablice jako **tablice asocjacyjne**; jedyną różnicą między tablicami asocjacyjnymi a indeksowanymi jest to, jakie są klucze. Niektóre funkcje tablic są dostarczane głównie do użytku z tablicami indeksowanymi, ponieważ zakładają, że mamy lub chcemy klucze będące kolejnymi liczbami całkowitymi zaczynającymi się od 0. W obu przypadkach klucze są unikalne. Innymi słowy, nie możemy mieć dwóch elementów z tym samym kluczem, niezależnie od tego, czy klucz jest łańcuchem, czy liczbą całkowitą. Tablice `PHP` mają wewnętrzny porządek swoich elementów, który jest niezależny od kluczy i wartości, i istnieją funkcje, które możemy użyć do przechodzenia przez tablice w oparciu o ten wewnętrzny porządek. Kolejność jest zwykle taka, w jakiej wartości zostały wstawione do tablicy, ale funkcje sortowania pozwalają zmienić kolejność na taką opartą na kluczach, wartościach lub cokolwiek innego, co wybierzemy.

Zanim przyjrzymy się tworzeniu tablicy, spójrzmy na strukturę istniejącej tablicy. Możemy uzyskać dostęp do określonych wartości z istniejącej tablicy, używając nazwy zmiennej tablicowej, po której następuje klucz lub indeks elementu, w nawiasach kwadratowych:

```
$age['fred']
$shows[2]
```

Klucz może być ciągiem lub liczbą całkowitą. Wartości łańcuchowe, które są równoważne liczbom całkowitym (bez zer wiodących) są traktowane jako liczby całkowite. Zatem `$array[3]` i `$array['3']` odwołują się do tego samego elementu, ale `$array['03']` odwołuje się do innego elementu. Liczby ujemne są prawidłowymi kluczami, ale nie określają pozycji od końca tablicy, jak to ma miejsce w `Perlu`.

```
<?php
    $person = array("name" => 'Peter');
    print "Hello, {$person['name']}";
?>
```

Przechowywanie wartości w tablicy spowoduje utworzenie tablicy, jeśli jeszcze nie istnieje, ale próba pobrania wartości z tablicy, która nie została zdefiniowana, nie spowoduje utworzenia tablicy. Na przykład:

```
<?php
    echo $addresses[0];
    echo $addresses;
    $addresses[0] = "spam@cyberpromo.net";
    echo $addresses;
?>
```

Użycie prostego przypisania do zainicjowania tablicy w programie może prowadzić do następującego kodu:

```
<?php
    $addresses[0] = "spam@cyberpromo.net";
    $addresses[1] = "abuse@example.com";
    $addresses[2] = "root@example.com";
?>
```

To jest tablica indeksowana, z indeksami całkowitymi zaczynającymi się od 0. Natomiast poniżej znajduje się przykład tablicy asocjacyjnej:

```
<?php
    $price['gasket'] = 15.29;
    $price['wheel'] = 75.25;
    $price['tire'] = 50.00;
?>
```

Łatwiejszym sposobem na zainicjowanie tablicy jest użycie konstrukcji `array()`, która buduje tablicę z jej argumentów. Tworzy to tablicę indeksowaną, a wartości indeksu (począwszy od 0) są tworzone automatycznie:

```
<?php
    $addresses = array("spam@cyberpromo.net", "abuse@example.com", "root@example.com");
?>
```

Aby utworzyć tablicę asocjacyjną za pomocą `array()`, należy użyć symbolu `=>` do oddzielenia indeksów (kluczy) od wartości:

```
<?php
    $price = array(
        'gasket' => 15.29,
        'wheel' => 75.25,
        'tire' => 50.00
    );
?>
```

Użycie spacji i wyrównania jest opcjonalne. Moglibyśmy zebrać kod, ale nie byłby tak łatwy do odczytania (jest to odpowiednik poprzedniego przykładowego kodu) ani tak łatwy do dodania lub usunięcia wartości:

```
<?php
    $price = array('gasket' => 15.29, 'wheel' => 75.25, 'tire' => 50.00);
?>
```

Możemy również określić tablicę, używając krótszej, alternatywnej składni:

```
<?php
    $price = ['gasket' => 15.29, 'wheel' => 75.25, 'tire' => 50.0];
?>
```

Aby skonstruować pustą tablicę, nie przekazujemy żadnych argumentów do `array()`:

```
<?php
    $addresses = array();
?>
```

Możemy określić początkowy klucz za pomocą `=>`, a następnie listę wartości. Wartości są wstawiane do tablicy zaczynając od tego klucza, a kolejne wartości mają klucze sekwencyjne:

```
<?php
    $days = array(1 => "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun");
?>
```

Jeśli początkowy indeks jest ciągiem nienumerycznym, kolejne indeksy są liczbami całkowitymi zaczynającymi się od 0. Zatem poniższy kod może prowadzić do nieporozumień:

```
<?php
    $whoops = array('Fri' => "Black", "Brown", "Green");
    echo $whoops[1];
?>
```

## Step 4

Aby dodać więcej wartości na końcu istniejącej tablicy indeksowanej, należy użyć składni `[]`:

```
<?php
    $family = array("Fred", "Wilma");
    $family[] = "Pebbles";
    echo $family[2];
?>
```

Ta konstrukcja zakłada, że indeksy tablicy są liczbami i przypisuje elementy do następnego dostępnego indeksu numerycznego, zaczynając od 0. Próba dołączenia do tablicy asocjacyjnej bez odpowiednich kluczy jest prawie zawsze błędem programisty, ale `PHP` przydzieli nowym elementom indeksy numeryczne bez wydawania ostrzeżenia:

```
<?php
    $person = array('name' => "Fred");
    $person[] = "Wilma";
    echo $person[0];
?>
```

Funkcja `range()` tworzy tablicę kolejnych wartości całkowitych lub znakowych między dwiema wartościami, które przekazujemy jako argumenty, i włączając w to dwie wartości. Na przykład:

```
<?php
    $numbers = range(2, 5);
    $letters = range('a', 'z');
    $reversedNumbers = range(5, 2);
    var_dump($numbers);
    var_dump($letters);
    var_dump($reversedNumbers);
?>
```

Do zbudowania zakresu używana jest tylko pierwsza litera argumentu ciągu znaków:

```
<?php
    $numbers = range(2, 5);
    $letters = range('aaa', 'zzz');
    $reversedNumbers = range(5, 2);
    var_dump($numbers);
    var_dump($letters);
    var_dump($reversedNumbers);
?>
```

Funkcje `count()` i `sizeof()` są identyczne w użyciu i działaniu. Zwracają liczbę elementów w tablicy. Nie ma preferencji stylistycznych dotyczących używanej funkcji. Oto przykład:

```
<?php
    $family = array("Fred", "Wilma", "Pebbles");
    $size = count($family);
    $size_of = sizeof($family);
    echo "$size\n";
    echo $size_of;
?>
```

Ta funkcja zlicza tylko wartości tablicy, które są faktycznie ustawione:

```
<?php
    $confusion = array( 10 => "ten", 11 => "eleven", 12 => "twelve");
    $size = count($confusion);
    echo $size;
?>
```

Aby utworzyć tablicę z wartościami zainicjowanymi do tej samej zawartości, należy użyć funkcji `array_pad()`. Pierwszym argumentem `array_pad()` jest tablica, drugim argumentem jest minimalna liczba elementów, jakie ma mieć tablica, a trzecim argumentem jest wartość, która ma dać wszelkim utworzonym elementom. Funkcja `array_pad()` zwraca nową tablicę uzupełnioną elementami tablicy z pierwszego argumentu, gdzie pozostałe utworzone elementy (podane w drugim argumencie) mają wartość argumentu trzeciego:

```
<?php
    $scores = array(5, 10);
    $padded = array_pad($scores, 5, 0);
    var_dump($scores);
    var_dump($padded);
?>
```

Nowe wartości są dołączane do tablicy. Jeśli chcemy dodać nowe wartości na początku tablicy, należy użyć ujemnego drugiego argumentu:

```
<?php
    $scores = array(5, 10);
    $padded = array_pad($scores, -5, 0);
    var_dump($scores);
    var_dump($padded);
?>
```

Jeśli dopełnimy tablicę asocjacyjną, istniejące klucze zostaną zachowane. Nowe elementy będą miały klawisze numeryczne zaczynające się od 0.

## Step 5

Wartości w tablicy mogą same być tablicami. Pozwala to łatwo tworzyć wielowymiarowe tablice:

```
<?php
    $row0 = array(1, 2, 3);
    $row1 = array(4, 5, 6);
    $row2 = array(7, 8, 9);
    $multi = array($row0, $row1, $row2);
    var_dump($multi);
?>
```

Możemy odwoływać się do elementów tablic wielowymiarowych, dodając więcej nawiasów kwadratowych, `[]`:

```
<?php
    $row0 = array(1, 2, 3);
    $row1 = array(4, 5, 6);
    $row2 = array(7, 8, 9);
    $multi = array($row0, $row1, $row2);
    $value = $multi[2][0];
    echo $value;
?>
```

Aby wyświetlić wartość podczas przeszukiwania tablicy wielowymiarowej, należy ująć całe wyszukiwanie tablicy w nawiasy klamrowe:

```
<?php
    $row0 = array(1, 2, 3);
    $row1 = array(4, 5, 6);
    $row2 = array(7, 8, 9);
    $multi = array($row0, $row1, $row2);
    echo("The value at row 2, column 0 is {$multi[2][0]}\n");
?>
```

Nieużycie nawiasów klamrowych skutkuje następuje outputem: `The value at row 2, column 0 is Array[0]`.

Aby skopiować wszystkie wartości tablicy do zmiennych, należy użyć konstrukcji `list()`:

```
list ($variable, ...) = $array;
```

Wartości tablicy są kopiowane do wymienionych zmiennych w porządku wewnętrznym tablicy. Domyślnie jest to kolejność, w jakiej zostały wstawione, ale można je zmienić za pomocą funkcji sortowania.

```
<?php
    $person = array("Fred", 35, "Betty");
    list($name, $age, $wife) = $person;
    echo "$name\n";
    echo "$age\n";
    echo "$wife\n";
?>
```

Jeśli mamy więcej wartości w tablicy niż w `list()`, dodatkowe wartości są ignorowane:

```
<?php
    $person = array("Fred", 35, "Betty");
    list($name, $age) = $person;
    echo "$name\n";
    echo "$age\n";
?>
```

Jeśli mamy więcej wartości w `list()` niż w tablicy, dodatkowe wartości są ustawiane na wartość `NULL`:

```
<?php
    $values = array("hello", "world");
    list($a, $b, $c) = $values;
    echo "$a\n";
    echo "$b\n";
    var_dump($c);
?>
```

Co najmniej dwa kolejne przecinki w `list()` pomijają wartości w tablicy:

```
<?php
    $values = range('a', 'e');
    list($m, , $n, , $o) = $values;
    echo "$m\n";
    echo "$n\n";
    echo "$o\n";
?>
```

Aby wyodrębnić tylko podzbiór tablicy, należy użyć funkcji `array_slice()`:

```
$subset = array_slice(array, offset, length);
```

Funkcja `array_slice()` zwraca nową tablicę składającą się z kolejnych serii wartości z oryginalnej tablicy. Parametr `offset` identyfikuje początkowy element do skopiowania ( `0` oznacza pierwszy element w tablicy), a parametr `length` określa liczbę wartości do skopiowania. Nowa tablica ma kolejne klucze numeryczne zaczynające się od `0`.

```
<?php
    $people = array("Tom", "Dick", "Harriet", "Brenda", "Jo");
    $middle = array_slice($people, 2, 2);
    var_dump($middle);
?>
```

Sensowne jest używanie `array_slice()` w tablicach indeksowanych (tj. tych z kolejnymi indeksami całkowitymi zaczynającymi się od `0`):

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Betty");
    $subset = array_slice($person, 1, 2);
    var_dump($subset);
?>
```

Połączenie `array_slice()` z `list()`, wyodrębnia tylko niektóre wartości do zmiennych:

```
<?php
    $order = array("Tom", "Dick", "Harriet", "Brenda", "Jo");
    list($second, $third) = array_slice($order, 1, 2);
    echo "$second\n";
    echo "$third\n";
?>
```

## Step 6

Aby podzielić tablicę na mniejsze tablice o równych rozmiarach, należy użyć funkcji `array_chunk()`:

```
$chunks = array_chunk(array, size [, preserve_keys]);
```

Funkcja zwraca tablicę mniejszych tablic. Trzeci argument, `preserve_keys`, jest wartością logiczną, która określa, czy elementy nowych tablic mają te same klucze, co w oryginale (przydatne w przypadku tablic asocjacyjnych), czy też nowe klucze numeryczne zaczynające się od `0` (przydatne w przypadku tablic indeksowanych). Domyślnym ustawieniem jest przypisanie nowych kluczy.

```
<?php
    $nums = range(1, 7);
    $rows = array_chunk($nums, 3);
    print_r($rows);
?>
```

Funkcja `array_keys()` zwraca tablicę składającą się tylko z kluczy w tablicy w kolejności tak jak zostały przypisane:

```
$arrayOfKeys = array_keys(array);
```

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    $keys = array_keys($person);
    print_r($keys);
?>
```

`PHP` zapewnia również (ogólnie mniej użyteczną) funkcję do pobierania tablicy tylko wartości z tablicy, `array_values()`:

```
$arrayOfValues = array_values(array);
```

Podobnie jak w przypadku `array_keys()`, wartości są zwracane w podanej kolejności tablicy:

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    $values = array_values($person);
    print_r($values);
?>
```

Aby sprawdzić, czy element istnieje w tablicy, należy użyć funkcji `array_key_exists()`:

```
if (array_key_exists(key, array)) { ... }
```

Funkcja zwraca wartość boolowską wskazującą, czy pierwszy argument jest prawidłowym kluczem w tablicy podanej jako drugi argument. Następujący kod może wprowadzać w błąd:

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    $values = array_values($person);
    if ($person['name']) {
        echo "Is value of key name";
    }
?>
```

Nawet jeśli w tablicy znajduje się element z nazwą klucza, odpowiadająca mu wartość może być fałszem (tj. `0`, `NULL` lub pusty ciąg). Zamiast tego należy użyć `array_key_exists()` w następujący sposób:

```
<?php
    $person['age'] = 0;
    if ($person['age']) {
        echo "true!\n";
    }
    if (array_key_exists('age', $person)) {
        echo "exists!\n";
    }
?>
```

Wiele osób zamiast `array_key_exists()` używa funkcji `isset()`, która zwraca `true`, jeśli element istnieje i nie ma wartości `NULL`:

```
<?php
    $a = array(0, NULL, '');
    function tf($v)
    {
        return $v ? 'T' : 'F';
    }
    for ($i=0; $i < 4; $i++) {
        printf("%d: %s %s\n", $i, tf(isset($a[$i])), tf(array_key_exists($i,$a)));
    }
?>
```

## Step 7

Funkcja `array_splice()` może usuwać lub wstawiać elementy do tablicy i opcjonalnie tworzyć inną tablicę z usuniętych elementów:

```
$removed = array_splice(array, start [, length [, replacement ] ]);
```

Do przykładów rozważmy następującą tablicę:

```
$subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
```

Możemy usunąć elementy `„math”`, `„bio”` i `„cs”`, nakazując `array_splice()` rozpoczęcie od pozycji `2` i usunięcie `3` elementów:

```
<?php
    $subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
    print_r($subjects);
    $removed = array_splice($subjects, 2, 3);
    print_r($subjects);
    print_r($removed);
?>
```

Jeśli pominiemy długość (argument `length`), `array_splice()` zacznie usuwać elementy aż do końca tablicy:

```
<?php
    $subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
    print_r($subjects);
    $removed = array_splice($subjects, 2);
    print_r($subjects);
    print_r($removed);
?>
```

Jeśli chcemy po prostu usunąć elementy z tablicy źródłowej i nie zależy nam na zachowaniu ich wartości, nie musimy przechowywać wyników funkcji `array_splice()`:

```
<?php
    $subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
    print_r($subjects);
    array_splice($subjects, 2);
    print_r($subjects);
?>
```

Aby wstawić elementy, zamiast tych usuniętych, należy użyć czwartego argumentu:

```
<?php
    $subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
    $new = array("law", "business", "IS");
    print_r($subjects);
    array_splice($subjects, 4, 3, $new);
    print_r($subjects);
?>
```

Rozmiar tablicy podanej jako czwarty argument funkcji `array_splice()` nie musi być taki sam, jak liczba usuwanych elementów. Tablica rośnie lub kurczy się zgodnie z potrzebami:

```
<?php
    $subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
    $new = array("law", "business", "IS");
    print_r($subjects);
    array_splice($subjects, 3, 4, $new);
    print_r($subjects);
?>
```

Aby wstawić nowe elementy do tablicy, jednocześnie przesuwając istniejące elementy w prawo, jako trzeci argument należy wpisać liczbę zero:

```
<?php
    $subjects = array("physics", "chem", "math");
    $new = array("law", "business");
    print_r($subjects);
    array_splice($subjects, 2, 0 ,$new);
    print_r($subjects);
?>
```

Chociaż dotychczasowe przykłady wykorzystywały tablicę indeksowaną, `array_splice()` działa również na tablicach asocjacyjnych:

```
<?php
    $capitals = array(
        'USA' => "Washington",
        'Great Britain' => "London",
        'New Zealand' => "Wellington",
        'Australia' => "Canberra",
        'Italy' => "Rome",
        'Canada' => "Ottawa"
    );
    print_r($capitals);
    $downUnder = array_splice($capitals, 2, 2);
    $france = array('France' => "Paris");
    print_r($capitals);
    array_splice($capitals, 1, 0, $france);
    print_r($capitals);
?>
```

## Step 8

`PHP` udostępnia dwie funkcje, `extract()` i `compact()`, które konwertują między tablicami i zmiennymi. Nazwy zmiennych odpowiadają kluczom w tablicy, a wartości zmiennych stają się wartościami w tablicy. Na przykład tablica

```
$person = array('name' => "Fred", 'age' => 35, 'wife' => "Betty");
```

może być przekonwertowana lub zbudowana ze zmiennych:

```
$name = "Fred";
$age = 35;
$wife = "Betty";
```

Funkcja `extract()` automatycznie tworzy zmienne lokalne z tablicy. Indeksy elementów tablicy stają się nazwami zmiennych:

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Betty");
    extract($person);
    echo "$name\n";
    echo "$wife\n";
    echo "$age\n";
?>
```

Jeśli zmienna utworzona przez funkcję `extract()` ma taką samą nazwę jak już istniejąca, wartość istniejącej zmiennej jest zastępowana wartością z tablicy. Możemy zmodyfikować zachowanie funkcji `extract()`, przekazując drugi argument. opisuje. Najbardziej użyteczną wartością jest `EXTR_PREFIX_ALL`, która wskazuje, że trzecim argumentem funkcji `extract()` jest prefiks dla tworzonych nazw zmiennych. Pomaga to zapewnić tworzenie unikalnych nazw zmiennych podczas korzystania z metody `extract()`. Dobrym stylem `PHP` jest zawsze używać `EXTR_PREFIX_ALL`:

```
<?php
    $shape = "round";
    $array = array('cover' => "bird", 'shape' => "rectangular");
    extract($array, EXTR_PREFIX_ALL, "book");
    echo "Cover: {$book_cover}, Book Shape: {$book_shape}, Shape: {$shape}";
?>
```

Funkcja `compact()` jest odwrotnością funkcji `extract();` przekazujemy mu nazwy zmiennych do załadowania jako oddzielne parametry lub w tablicy. Funkcja `compact()` tworzy tablicę asocjacyjną, której klucze są nazwami zmiennych, a których wartościami są wartości tych zmiennych. Wszelkie nazwy w tablicy, które nie odpowiadają rzeczywistym zmiennym, są pomijane.

```
<?php
    $color = "indigo";
    $shape = "curvy";
    $floppy = "none";
    $a = compact("color", "shape", "floppy");
    print_r($a);
    $names = array("color", "shape", "floppy");
    $a = compact($names);
    print_r($a);
?>
```

## Step 9

Najczęstszym sposobem na zapętlenie elementów tablicy jest użycie konstrukcji `foreach`:

```
<?php
    $addresses = array("spam@cyberpromo.net", "abuse@example.com");
    foreach ($addresses as $value) {
        echo "Processing {$value}\n";
    }
?>
```

`PHP` wykonuje ciało pętli (instrukcję `echo`) raz dla każdego elementu `$addresses` po kolei, z `$value` ustawioną na bieżący element. Elementy są przetwarzane według ich kolejności.

Alternatywna do `foreach` jest dostęp do aktualnego klucza:

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    foreach ($person as $key => $value) {
        echo "Fred's {$key} is {$value}\n";
    }
?>
```

W tym przypadku klucz dla każdego elementu jest umieszczany w `$key`, a odpowiadająca mu wartość jest umieszczana w `$value`. Konstrukcja `foreach` nie działa na samej tablicy, ale raczej na jej kopii. Możemy wstawiać lub usuwać elementy w ciele pętli `foreach`, wiedząc, że pętla nie będzie próbowała przetworzyć usuniętych lub wstawionych elementów.

Każda tablica `PHP` śledzi bieżący element, z którym pracujemy; jest to wskaźnik do bieżącego elementu, który jest nazywany **iteratorem**. `PHP` posiada funkcje do ustawiania, przenoszenia i resetowania iteratora. Funkcje iteratorów przedstawia poniższa tabela:

| Funkcja iteratora | Opis funkcji |
|---|---|
| `current()` | Zwraca element aktualnie wskazywany przez iterator. |
| `reset()` | Przenosi iterator do pierwszego elementu w tablicy i zwraca go |
| `next()` | Przenosi iterator do następnego elementu w tablicy i zwraca go. |
| `prev()` | Przenosi iterator do poprzedniego elementu w tablicy i zwraca go. |
| `end()` | Przenosi iterator do ostatniego elementu w tablicy i zwraca go. |
| `each()` - Depreceated | Zwraca klucz i wartość bieżącego elementu jako tablicę i przesuwa iterator do następnego elementu w tablicy. |
| `key()` | Zwraca klucz bieżącego elementu. |

```
<?php
    $ages = array(
        'Person' => "Age",
        'Fred' => 35,
        'Barney' => 30,
        'Tigger' => 8,
        'Pooh' => 40
    );
    $element = current($ages);
    echo "$element\n";
    $element = next($ages);
    echo "$element\n";
    $element = prev($ages);
    $key = key($ages);
    echo "ages[$key] => $element\n";
    $element = end($ages);
    echo "$element\n";
    reset($ages);
    $element = current($ages);
    echo "$element\n"
?>
```

Jeśli wiemy, że mamy do czynienia z tablicą indeksowaną, w której klucze są kolejnymi liczbami całkowitymi zaczynającymi się od `0`, możemy użyć pętli `for` do przeliczania indeksów. Pętla `for` działa na samej tablicy, a nie na kopii tablicy, i przetwarza elementy w kolejności kluczy, niezależnie od ich kolejności wewnętrznej. Oto jak wydrukować tablicę za pomocą `for`:

```
<?php
    $addresses = array("spam@cyberpromo.net", "abuse@example.com");
    $addressCount = count($addresses);
    for ($i = 0; $i < $addressCount; $i++) {
        $value = $addresses[$i];
        echo "{$value}\n";
    }
?>
```

`PHP` udostępnia mechanizm `array_walk()`, służący do wywoływania funkcji zdefiniowanej przez użytkownika raz na element tablicy:

```
array_walk(array, callable);
```

Zdefiniowana funkcja przyjmuje dwa lub opcjonalnie trzy argumenty: pierwszy to wartość elementu, drugi to klucz elementu, a trzeci to wartość dostarczana do `array_walk()` podczas jej wywołania. Na przykład, oto inny sposób drukowania kolumn tabeli utworzonych z wartości z tablicy:

```
<?php
    $printRow = function ($value, $key)
    {
        print("<tr><td>{$key}</td><td>{$value}</td></tr>\n");
    };
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    echo "<table border=1>";
    array_walk($person, $printRow);
    echo "</table>";
?>
```

Kolejny przykład określa kolor tła za pomocą opcjonalnego trzeciego argumentu funkcji `array_walk()`. Ten parametr daje nam elastyczność, której możemy potrzebować, aby wydrukować wiele tabel, z wieloma kolorami tła:

```
<?php
    function printRow($value, $key, $color)
    {
        echo "<tr>\n<td bgcolor=\"{$color}\">{$value}</td>";
        echo "<td bgcolor=\"{$color}\">{$key}</td>\n</tr>\n";
    }
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    echo "<table border=\"1\">";
    array_walk($person, "printRow", "lightblue");
    echo "</table>";
?>
```

Jeśli mamy wiele opcji, które chcemy przekazać do wywoływanej funkcji, po prostu przekazujemy tablicę jako trzeci parametr:

```
<?php
    $extraData = array('border' => 2, 'color' => "red");
    $baseArray = array("Ford", "Chrysler", "Volkswagen", "Honda", "Toyota");
    array_walk($baseArray, "walkFunction", $extraData);
    function walkFunction($item, $index, $data)
    {
        echo "{$item} <- item, then border: {$data['border']}";
        echo " color->{$data['color']}<br />" ;
    }
?>
```

Funkcja `array_walk()` przetwarza elementy w kolejności, jaką są zapisane w tablicy.

Kuzyn `array_walk()`, `array_reduce()`, stosuje funkcję kolejno do każdego elementu tablicy, aby zbudować pojedynczą wartość:

```
$result = array_reduce(array, callable [, default ]);
```

```
<?php
    $addItUp = function ($runningTotal, $currentValue)
    {
        $runningTotal += $currentValue * $currentValue;
        return $runningTotal;
    };
    $numbers = array(2, 3, 5, 7);
    $total = array_reduce($numbers, $addItUp);
    echo $total;
?>
```

Funkcja przyjmuje dwa argumenty: `runningTotal` i `currentValue`. Program zwraca sumę kwadratów tablicy. Funkcja `array_reduce()` wykonuje następujące wywołania funkcji:

```
addItUp(0, 2);
addItUp(4, 3);
addItUp(13, 5);
addItUp(38, 7);
```

Argument `default`, jeśli został podany, jest wartością inicjalizacyjną. Na przykład, jeśli zmienimy wywołanie `array_reduce()` w poprzednim przykładzie na:

```
<?php
    $addItUp = function ($runningTotal, $currentValue)
    {
        $runningTotal += $currentValue * $currentValue;
        return $runningTotal;
    };
    $numbers = array(2, 3, 5, 7);
    $total = array_reduce($numbers, $addItUp, 11);
    echo $total;
?>
```

Wywołaniami funkcji są:

```
addItUp(11, 2);
addItUp(15, 3);
addItUp(24, 5);
addItUp(49, 7);
```

## Step 10

Funkcja `in_array()` zwraca prawdę ( `true` ) lub fałsz ( `false` ), w zależności od tego, czy pierwszy argument jest elementem tablicy podanej jako drugi argument:

```
if (in_array(to_find, array [, strict])) { ... }
```

Jeśli opcjonalny trzeci argument jest `true` , typ używany w argumencie `to_find` i wartość w tablicy `array` muszą być zgodne. Domyślnie nie sprawdza się typów danych.

```
<?php
    $addresses = array("spam@cyberpromo.net", "abuse@example.com","root@example.com");
    $gotSpam = in_array("spam@cyberpromo.net", $addresses);
    $gotMilk = in_array("milk@tucows.com", $addresses);
    var_dump($gotSpam);
    var_dump($gotMilk);
?>
```

`PHP` automatycznie indeksuje wartości w tablicach, więc `in_array()` jest generalnie znacznie szybsze niż pętla sprawdzająca każdą wartość w tablicy, aby znaleźć tę, którą chcemy.

Odmianą `in_array()` jest funkcja `array_search()` . Podczas gdy `in_array()` zwraca prawdę ( `true` ), jeśli wartość zostanie znaleziona, `array_search()` zwraca klucz elementu, jeśli został znaleziony:

```
<?php
    $person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
    $k = array_search("Wilma", $person);
    echo("Fred's $k is Wilma\n");
?>
```

Funkcja `array_search()` pobiera również opcjonalny trzeci ścisły argument, który wymaga, aby typy szukanej wartości i wartości w tablicy były zgodne (podobnie jak w przypadku funkcji `in_array` ).

**Sortowanie** zmienia wewnętrzną kolejność elementów w tablicy i opcjonalnie przepisuje klucze, aby odzwierciedlić tę nową kolejność. Możemy na przykład użyć sortowania, aby uporządkować listę wyników od największej do najmniejszej, ułożyć alfabetycznie listę nazwisk lub uporządkować zestaw użytkowników na podstawie liczby opublikowanych przez nich wiadomości. `PHP` udostępnia trzy sposoby sortowania tablic — sortowanie według kluczy, sortowanie według wartości bez zmiany kluczy lub sortowanie według wartości, a następnie zmienianie kluczy. Każdy rodzaj sortowania można wykonać w kolejności rosnącej, malejącej lub określonej przez funkcję zdefiniowaną przez użytkownika. Funkcje dostarczane przez `PHP` do sortowania tablicy są pokazane w tabeli:

| Użycie | Rosnąco | Malejąco | Zdefiniowane przez użytkownika |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Sortuj tablicę według wartości, a następnie ponownie przypisz indeksy, zaczynając od 0 | `sort()` | `rsort()` | `usort()` |
| Sortuj tablicę według wartości | `asort()` | `arsort()` | `uasort()` |
| Sortuj tablicę według kluczy | `ksort()` | `krsort()` | `uksort()` |

Funkcje `sort()`, `rsort()` i `usort()` są zaprojektowane do pracy na tablicach indeksowanych, ponieważ przypisują nowe klucze numeryczne do reprezentowania kolejności. Są przydatne, gdy musimy odpowiedzieć na pytania, takie jak „Jakie jest 10 najlepszych wyników?" i „Kto jest trzecią osobą w kolejności alfabetycznej?" Inne funkcje sortowania mogą być używane na tablicach indeksowanych, ale dostęp do posortowanej kolejności można uzyskać tylko za pomocą konstrukcji przechodzenia, takich jak `foreach` i `next()`.

Aby posortować nazwy w rosnącej kolejności alfabetycznej, można wykonać coś takiego:

```
<?php
    $names = array("Cath", "Angela", "Brad", "Mira");
    print_r($names);
    sort($names);
    print_r($names);
?>
```

Aby uzyskać je w odwrotnej kolejności alfabetycznej, po prostu należy wywołać `rsort()` zamiast `sort()`.

```
<?php
    $names = array("Cath", "Angela", "Brad", "Mira");
    print_r($names);
    rsort($names);
    print_r($names);
?>
```

Jeśli mamy tablicę asocjacyjną, która odwzorowuje nazwy użytkowników wraz z minutami czasu logowania, możemy użyć `arsort()`, aby wyświetlić tabelę trzech pierwszych użytkowników:

```
<?php
    $logins = array(
        'njt' => 415,
        'kt' => 492,
        'rl' => 652,
        'jht' => 441,
        'jj' => 441,
        'wt' => 402,
        'hut' => 309,
    );
    arsort($logins);
    $numPrinted = 0;
    echo "<table>\n";
    foreach ($logins as $user => $time) {
        echo("<tr><td>{$user}</td><td>{$time}</td></tr>\n");
        if (++$numPrinted == 3) {
            break; // stop after three
        }
    }
    echo "</table>";
?>
```

Jeśli chcesz, aby ta tabela była wyświetlana w porządku rosnącym według nazwy użytkownika, należy użyć funkcji `ksort()` zamiast `arsort()`.

```php
<?php
    $logins = array(
        'njt' => 415,
        'kt' => 492,
        'rl' => 652,
        'jht' => 441,
        'jj' => 441,
        'wt' => 402,
        'hut' => 309,
    );
    ksort($logins);
    $numPrinted = 0;
    echo "<table>\n";
    foreach ($logins as $user => $time) {
        echo("<tr><td>{$user}</td><td>{$time}</td></tr>\n");
        if (++$numPrinted == 3) {
            break; // stop after three
        }
    }
    echo "</table>";
?>
```

Użycie sortowania zdefiniowanego przez użytkownika wymaga podania funkcji, która przyjmuje dwie wartości i zwraca wartość, która określa kolejność dwóch wartości w posortowanej tablicy. Funkcja powinna zwrócić `1`, jeśli pierwsza wartość jest większa od drugiej, `-1`, jeśli pierwsza wartość jest mniejsza od drugiej, i `0`, jeśli wartości są takie same na potrzeby niestandardowego porządku sortowania.

```php
<?php
function userSort($a, $b)
    {
        if ($b == "smarts") {
            return 1;
        }
        else if ($a == "smarts") {
            return -1;
        }
        return ($a == $b) ? 0 : (($a < $b) ? -1 : 1);
    }
    $values = array(
        'name' => "Buzz Lightyear",
        'email_address' => "buzz@starcommand.gal",
        'age' => 32,
        'smarts' => "some"
    );
    print_r($values);
    uksort($values,"userSort");
    print_r($values);
?>
```

Wbudowane funkcje sortowania `PHP` poprawnie sortują ciągi i liczby, ale nie sortują poprawnie ciągów zawierających liczby. Na przykład, jeśli mamy nazwy plików `ex10.php`, `ex5.php` i `ex1.php`, normalne funkcje sortowania zmienią je w tej kolejności: `ex1.php, ex10.php, ex5.php`. Aby poprawnie posortować ciągi zawierające liczby, należy użyć funkcji `natsort()` i `natcasesort()`:

```php
<?php
    $values = array(
        "ex10.php",
        "ex5.php",
        "ex1.php"
    );
    print_r($values);
    natsort($values);
    print_r($values);
?>
```

```
<?php
    $values = array(
        "ex10.php",
        "EX5.php",
        "ex1.php"
    );
    print_r($values);
    natcasesort($values);
    print_r($values);
?>
```

Funkcja array_multisort() sortuje tablice wielowymiarowe jednocześnie:

```
array_multisort(array1 [, array2, ... ]);
```

Funkcja ta jako argumenty przyjmuje serię tablic i porządków sortowania (identyfikowanych przez stałe `SORT_ASC` lub `SORT_DESC`) i zmienia kolejność elementów wszystkich tablic, przypisując nowe indeksy. Jest to podobne do operacji łączenia w relacyjnej bazie danych. Weźmy dla przykładu kilka fragmentów danych dotyczących każdej osoby:

```
$names = array("Tom", "Dick", "Harriet", "Brenda", "Joe");
$ages = array(25, 35, 29, 35, 35);
$zips = array(80522, '02140', 90210, 64141, 80522);
```

Pierwszy element każdej tablicy reprezentuje pojedynczy rekord — wszystkie znane informacje o `Tomie`. Podobnie drugi element stanowi kolejny zapis – wszystkie znane informacje o `Dicku`. Funkcja `array_multisort()` zmienia kolejność elementów tablic, zachowując rekordy. Oznacza to, że jeśli `"Dick"` znajdzie się jako pierwszy w tablicy `$names` po sortowaniu, reszta informacji `Dicka` będzie również pierwsza w innych tablicach. (Zauważmy, że musieliśmy zacytować kod pocztowy `Dicka`, aby zapobiec interpretowaniu go jako stałej ósemkowej.) Oto jak posortować rekordy najpierw rosnąco według wieku, a następnie malejąco według kodu pocztowego:

```
<?php
    $names = array("Tom", "Dick", "Harriet", "Brenda", "Joe");
    $ages = array(25, 35, 29, 35, 35);
    $zips = array(80522, '02140', 90210, 64141, 80522);
    for ($i = 0; $i < count($names); $i++) {
        echo "{$names[$i]}, {$ages[$i]}, {$zips[$i]}\n";
    }
    echo "----------------------\n";
    array_multisort($ages, SORT_ASC, $zips, SORT_DESC, $names, SORT_ASC);
    for ($i = 0; $i < count($names); $i++) {
        echo "{$names[$i]}, {$ages[$i]}, {$zips[$i]}\n";
    }
?>
```

Funkcja `array_reverse()` odwraca wewnętrzną kolejność elementów w tablicy:

```
<?php
    $names = array("Tom", "Dick", "Harriet", "Brenda", "Joe");
    print_r($names);
    $reversed = array_reverse($names);
    print_r($reversed);
    $associate = array("Men" => "Fred", "Child" => "Pebbles", "Wife" => "Wilma");
    print_r($associate);
    $reversed_associate = array_reverse($associate);
    print_r($reversed_associate);
?>
```

Indeksy zaczynają się od 0, podczas gdy klucze w tablicach asocjacyjnych pozostają niezmienione. Ogólnie rzecz biorąc, lepiej jest używać funkcji sortowania w odwrotnej kolejności zamiast sortowania, a następnie odwracania kolejności w tablicy.

Funkcja `array_flip()` zwraca tablicę, która odwraca kolejność pary klucz-wartość każdego oryginalnego elementu. Oznacza to, że dla każdego elementu tablicy, którego wartość jest prawidłowym kluczem, wartość elementu staje się jego kluczem, a klucz elementu staje się jego wartością. Na przykład, jeśli mamy tablicę, która odwzorowuje nazwy użytkowników na ich katalogi domowe, możemy użyć `array_flip()`, aby utworzyć tablicę, która odwzorowuje katalogi domowe na nazwy użytkowników:

```php
<?php
    $u2h = array(
        'gnat' => "/home/staff/nathan",
        'frank' => "/home/action/frank",
        'petermac' => "/home/staff/petermac",
        'ktatroe' => "/home/staff/kevin"
    );
    $h2u = array_flip($u2h);
    print_r($u2h);
    print_r($h2u);
?>
```

Elementy, których oryginalne wartości nie są ani łańcuchami, ani liczbami całkowitymi, pozostają same w wynikowej tablicy. Nowa tablica pozwala odkryć klucz w oryginalnej tablicy na podstawie jego wartości, ale ta technika działa skutecznie tylko wtedy, gdy oryginalna tablica ma unikalne wartości.

Aby przemieszać elementy tablicy w kolejności losowej, należy użyć funkcji `shuffle()`. Zastępuje wszystkie istniejące klucze — łańcuchowe lub numeryczne — kolejnymi liczbami całkowitymi zaczynającymi się od 0.

```php
<?php
    $weekdays = array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday");
    shuffle($weekdays);
    print_r($weekdays);
?>
```

Oczywiście kolejność po `shuffle()` zawsze może być inne niż poprzednio ze względu na losowy charakter funkcji. Jeśli nie jesteśmy zainteresowani pobieraniem wielu losowych elementów z tablicy bez powtarzania żadnego konkretnego elementu, użycie funkcji `rand()` do wybrania indeksu jest bardziej wydajne.

## Step 11

`PHP` ma kilka przydatnych wbudowanych funkcji do modyfikowania lub stosowania operacji na wszystkich elementach tablicy. Możemy obliczyć sumę tablicy, scalić wiele tablic, znaleźć różnicę między dwiema tablicami i nie tylko.

Funkcja `array_sum()` sumuje wartości w tablicy indeksowanej lub asocjacyjnej:

```php
<?php
    $scores = array(98, 76, 56, 80);
    $users = array("Bob" => 45, "Wilma" => 55);
    $total = array_sum($scores);
    $total_users = array_sum($users);
    echo "Total variable = $total\n";
    echo "Total_users variable = $total_users\n";
?>
```

Funkcja `array_merge()` inteligentnie łączy dwie lub więcej tablic. Jeśli klucz jest numeryczny z wcześniejszej tablicy powtarza się, wartości, a z późniejszej tablicy przypisywany jest nowy klucz numeryczny:

```php
<?php
    $first = array("hello", "world");
    $second = array("exit", "here");
    $merged = array_merge($first, $second);
    print_r($merged);
?>
```

Jeśli powtarza się klucz ciągu z wcześniejszej tablicy, to wcześniejsza wartość jest zastępowana późniejszą wartością:

```php
<?php
    $first = array('bill' => "clinton", 'tony' => "danza");
    $second = array('bill' => "gates", 'adam' => "west");
    $merged = array_merge($first, $second);
    print_r($merged);
?>
```

Funkcja `array_diff()` oblicza różnicę między dwiema lub większą liczbą tablic, zwracając tablicę z wartościami z pierwszej tablicy, których nie ma w pozostałych:

```
<?php
    $a1 = array("bill", "claire", "ella", "simon", "judy");
    $a2 = array("jack", "claire", "toni");
    $a3 = array("ella", "simon", "garfunkel");
    $difference = array_diff($a1, $a2, $a3);
    print_r($difference);
?>
```

Wartości są porównywane przy użyciu operatora ścisłego porównania, więc `1` i `„1”` są uważane za różne wartości. Klucze pierwszej tablicy są zachowywane, więc w `$diff` klucz `„bill”` to `0`, a klucz `„judy”` to `4`. W kolejnym przykładzie poniższy kod zwraca różnicę dwóch tablic:

```
<?php
    $first = array(1, "two", 3);
    $second = array("two", "three", "four");
    $difference = array_diff($first, $second);
    print_r($difference);
?>
```

Aby zidentyfikować podzbiór tablicy na podstawie jego wartości, należy użyć funkcji `array_filter()`:

```
array_filter(array, callback);
```

Każda wartość tablicy jest przekazywana do funkcji o nazwie `callback`. Zwrócona tablica zawiera tylko te elementy oryginalnej tablicy, dla których funkcja zwraca wartość `true`.

```
<?php
    function isOdd ($element) {
        return $element % 2;
    }
    $numbers = array(9, 23, 24, 27);
    $odds = array_filter($numbers, "isOdd");
    print_r($odds);
?>
```

Jak widać, klucze są zachowane. Ta funkcja jest najbardziej użyteczna w przypadku tablic asocjacyjnych.

## Step 12

Funkcje `readline()` można łączyć z funkcją `explode()`. W poniższym przykładzie separatorem jest spacja. Drugim argumentem jest funkcja `readline()`. Tutaj również typem danych `$var1` i `$var2` będzie ciąg tekstowy. Musimy więc oddzielnie rzutować je dla innych typów danych. W poniższym przykładzie rzutowanie typu jest pokazane dla liczb całkowitych.

```
<?php
  $array = explode(' ', readline());
  $var1 = (int)$array[0];
  $var2 = (int)$array[1];
  echo "The sum of " . $var1 . " and " . $var2 . " is " . ($var1 + $var2);
  print_r($array);
?>
```