If builders built buildings the way programmers wrote programs, the first woodpecker that came along would destroy civilization.

Unknown

2 Arrays

2.1 What is an array?

An array is a collection of elements, an ordered map. A map is a type that associates values to keys. It is optimized for several different uses. It can be treated as an array, list (vector), hash table, and more. As array values can be other arrays, trees and multidimensional arrays are also possible.

An array holds more then one value, ordered in key-value mappings. Each value of an array is called an element. Every element has a key. When you know the key, you can get the value.

You can think of an array as a cabinet with drawers. The array is the cabinet and every drawer holds a value. It is possible to create a new drawer by adding a new value to the array. Every drawer has a label. This label is the key that 'points' to the value. So the labels on a cabinet should be unique. You can get the whole cabinet or you can just get the value of one drawer by giving the key.

2.2 Arraytypes

In PHP there are two types of arrays: indexed arrays and associative arrays.

2.2.1 Indexed arrays

When using indexed arrays, you rely on PHP to label the drawers or you can choose labelnames by yourself, but only integer ones.

Listing 2.1 indexed array example

```php
<?php

$myArray = array('first value', 'second value', 'third value');

// or

$myArray = array(
        0 => 'first value',
        1 => 'second value',
        2 => 'third value'
        );

// or

$myArray = array(
        0 => 'first value',
        3 => 'second value',
        1009 => 'third value'
        );

?>
```

If you add three values (like the first line in the previous example), 'first value' automatically gets a key of `0` and so on.

If you skip some numbers (like the last example). PHP will add them in an indexed way and take the next integer as a key for the next value added, like in the example below.

Listing 2.2 adding an extra element to an array

```php
<?php

$myArray = array(
        0 => 'first value',
        3 => 'second value',
        1009 => 'third value'
        );

// another way to add an extra element to an array
$myArray[] = 'fourth value';

/*
$myArray now contains four elements.
The index of the fourth element is 1010, this is the next integer based on the highest index.
*/

?>
```

The => hasn't got a real name, sometimes called the 'double arrow'. But is is an operator to link a key, value pair.

2.2.2 Associative arrays

When using associative arrays, you explicitly give a label for each drawer. You can freely choose the keyname for each drawer using more then only numeric characters.

Listing 2.3 associative array example

```
<?php

$myArray = array(
        'name' => 'Pilot',
        'surname' => 'Test',
        'address' => 'Somestreet 1'
        );

?>
```

There is no need to explicitly declare an array as an indexed or associative one. Just like variables, PHP manages this by itself.

2.2.3 Multidimensional arrays

The value you give to an element in an array can be an array by itself, thus creating multidimensional arrays. You can mix indexed and associative arrays.

Listing 2.4 multidimensional array example

```
<?php

$myArray = array(
        'title' => 'my example',
        'data' => array(
                        'id' => 44,
                        'name' => 'Jos',
                )
        );

?>
```

2.3 Assigning and getting the values of an array

As said before, an array is a *collection* of values, so how to get only one value? If you use an `echo()` call, you will get `Array();` because `echo()` only can print scalar values (values with one dimension), no arrays, no objects.

Listing 2.5 echo of an Array doesn't work

```
<?php

$myArray = array(
        0 => 'first value',
        3 => 'second value',
        1009 => 'third value'
        );

// the next line will just print Array(), not the actual keys and values. Try it!
echo $myAarray;

?>
```

Use the square brackets `[]` for this. Between the square brackets, put the key of the value you want to use. Make sure to use quotes if the key is a string value.

Listing 2.6 addressing values in an array example

```
<?php

$myArray = array(
        'name' => 'Pilot',
        'surname' => 'Test',
        'address' => 'Somestreet 1'
        );

// Getting the value for key 'address'
echo $myArray['address'];

// Assigning a new value to the key 'address'
$myArray['address'] = 'Another street 2';

// Because there is no key 'city' yet, this will add a new key=>value pair to the array
$myArray['city'] = 'Somecity';

// Or add a value to the next numeric value available
$myNumericArray = array('one','two','three');
$myNumericArray[] = 'four';

// this will print four
echo $myNumericArray[3];

?>
```

Note that when adding a key=>value to the array, this will create a new key at the end of the array.

2.3.1 Multidimensional arrays

If you have a multidimensional array, every dimension has its own brackets `[]`. Arrays can be n-dimensional. Look at the example below.

Listing 2.7 multidimensional array example

```php
<?php

$myArray = array(
        'title' => 'my example',
        'data' => array(
                        'id' => 44,
                        'name' => 'Jos',
                )
        );

// We want the value for key 'name' within the key 'data'
// This will print 'Jos'
echo $myArray['data']['name'];

?>
```

Exercise 2.1

- Make sure you understand multidimensional arrays and how to address each key and value.
- Download the source file from `http://dynweb.webontwerp.khleuven.be/exercises/2.1-source.txt`
- Use assigning/addressing values in an array to obtain the result you can see at `http://dynweb.webontwerp.khleuven.be/exercises/2.1.php`. (Look at the source-code)
- Upload your file so that it is accessible at `http://<studentnr>.webontwerp.khleuven.be/exercises/2.1.php`

2.4 array structures: looping through an array

There are several ways to cycle through all values in an array. For indexed arrays the keys most of the time don't matter so you can cycly through each element and display its value. Two common used ways:

Listing 2.8 cycling through numeric arrays

```php
<?php

$myArray = array('first value','third value','second value');

// count() is a function to ... count the number of elements in an array
$count = count($myArray);

/*
The next loop will print

   first value
   third value
   second value

*/

for ($i=0;$i<$count;$i++) {
        echo $myArray[$i] . "\n";
}

// foreach will copy the value for each key in the array $myArray to a variable $value
// The loop below will print the same result as the for-loop above
foreach ($myArray as $value) {
        echo $value . "\n";
}

?>
```

If you want to loop through a associative array, you can still use the foreach structure. If you want to get the keys as well, use the foreach structure as shown in the example below.

Listing 2.9 cycling through associative arrays

```php
<?php

$myArray = array(
        'name' => 'Pilot',
        'surname' => 'Test',
        'address' => 'Somestreet 1'
        );

// foreach will copy the key in the array $myArray to a variable $key
// and each corresponding value to the variabel $value
// try it out and see for yourself

foreach ($myArray as $key => $value) {
        echo $key . ': ' . $value . "\n";
}

?>
```

An imported note: While looping through the array and changing the value of `$value` doesn't change the original value. If you want to do this, address the value using the original arrayname and key like in the example below.

Listing 2.10 changing values while cycling through an array

```
<?php

$myArray = array(
        'name' => 'Pilot',
        'surname' => 'Test',
        'address' => 'Somestreet 1'
        );

// cycling, printing and changing the value
foreach ($myArray as $key => $value) {
        echo $key . ': ' . $value . "\n";
        $myArray[$key] = strtoupper($value);
}

// printing the changed value
foreach ($myArray as $key => $value) {
        echo $key . ': ' . $value . "\n";
}

?>
```

2.5 array functions

2.5.1 Sorting

The standard order of elements in an array is the order in which the elements were added. There are three main sorting possibilities.

- `sort()`: sorts the values, erasing the keys, creating a new indexed array
- `asort()`: sorts the values, keeping the keys (indexed and associative array)
- `ksort()`: sorts the keys, keeping the values in place

Listing 2.11 sort() sorting an indexed array

```
<?php

$myArray = array(
        'name' => 'Pilot',
        'surname' => 'Test',
        'address' => 'Somestreet 1'
        );

// try out these three sorting functions to see what output they give.
ksort($myArray);
asort($myArray);
sort($myArray);

?>
```

You can also sort high to low by using `rsort()`, `arsort()` or `krsort()`. More info on sorting arrays can be found at `www.php.net/manual/en/array.sorting.php`.

2.5.2 Conversions between string and arrays

Instead of looping through all values of an array, it can be interested to export all values of an array in a string, with a delimiter. Or to split values from one string on a certain delimiter. You can use `implode()` and `explode()` for that.

Listing 2.12 explode example

```
1  <?php
2
3  $myFriends = array('Willy', 'Marcel', 'Jos');
4
5  // note that the comma is only placed in between and not at the end.
6  echo 'These are my friends: ' . implode(', ', $myFriends) . "\n";
7
8  $myDate = '20/01/2011';
9  $date_elements = explode('/',$myDate);
10
11 foreach($date_elements as $value) {
12         echo $value . "\n";
13 }
14
15 ?>
```

Exercise 2.2

- Download the source file from `http://dynweb.webontwerp.khleuven.be/exercises/2.2-source.txt`
- Combine sorting and looping to obtain the result you can see at `http://dynweb.webontwerp.khleuven.be/exercises/2.2.php`. (Look at the source-code)
- Upload your file so that it is accessible at `http://<studentnr>.webontwerp.khleuven.be/exercises/2.2.php`

2.5.3 array functions

Arrays are very powerful in PHP . These are some of common used functions to check if a key, a value or an array exists. A complete list is available on `/www.php.net/manual/en/ref.array.php`

Make sure to look up these:

- `array_key_exists()`
- `in_array()`
- `array_search()`

- `unset()`

Exercise 2.3

- Download the source file from `http://dynweb.webontwerp.khleuven.be/exercises/2.3-source.txt`
- Combine all stuff about arrays to obtain the result you can see at `http://dynweb.webontwerp.khleuven.be/exercises/2.3.php`. (Look at the source-code)
- Upload your file so that it is accessible at `http://<studentnr>.webontwerp.khleuven.be/exercises/2.3.php`