

Hoofdstuk 3

Variabelen en datatypes

3.1 Inleiding

In PHP wordt een variabele aangeduid door een dollarteken (\$) gevolgd door een aantal geldige symbolen. Geldige symbolen omvatten letters, hoofdletters, cijfers en de underscore (_). Een variabele mag echter niet met een cijfer beginnen. Variabelen zijn 'case-sensitive' (hoofdlettergevoelig): \$getal en \$GETAL zijn twee verschillende variabelen.

In dit hoofdstuk worden variabelen van het type boolean, integer, double, string en NULL besproken. Booleans bevatten logische waarden, integers gehele getallen, doubles kommagetallen en strings tekstwaarden. NULL bevat de null-referentie.

PHP is een flexibele programmeertaal: variabelen worden niet gedeclareerd en elke variabele kan waarden van eender welk type bevatten. Het type van de variabele wordt bepaald door de context. In het onderstaande voorbeeld (**vb1.php**) wordt de waarde \$a afgedrukt voor hij een waarde gekregen heeft, hij is dan van het type NULL. Vervolgens wordt eerst de integer 1 aan de \$a toegewezen, dan de double met waarde 3.1415, de string met waarde 'tekst' en de boolean met waarde true. Tenslotte wordt via het commando unset de variabele 'leeggemaakt' en is hij opnieuw van het type NULL.

vb1.php

```
1 <?php
2 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");
3 $a = 1;
4 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");
5 $a = 3.1415;
6 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");
7 $a = "tekst";
8 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");
9 $a = TRUE;
10 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");
11 unset($a);
12 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");
13 $a=null;
14 print ("a heeft waarde " . $a . " en type " . gettype($a) . "<br/>\n");?>
```

Het is ook mogelijk omzettingen tussen verschillende types te maken via cast-operaties. In het onderstaande voorbeeld (`vb2.php`) krijgt de variabele `$a` een lege string als waarde. De inhoud van de variabele `$a` wordt vervolgens via een cast-operatie omgezet naar het type boolean en het resultaat wordt in de variabele `$b` geplaatst.

`vb2.php`

```
1 <?php
2 $a = "";
3 print ("\$a is van het type ". gettype($a));
4 $b = (boolean) $a;
5 if ($b) {
6     print ("<br>waar <br>");
7 }
8 else {
9     print ("<br>niet waar <br>");
10 }
11 print ("\$b is van het type ". gettype($b));
12 ?>
```

De types boolean, integer, double, string en NULL en omzettingen tussen deze types worden in de volgende secties apart besproken.

3.2 Boolean

Een variabele van het type boolean is een logische variabele die de waarde true of false kan bevatten. In `vb3.php` wordt het gebruik van het datatype boolean geïllustreerd.

`vb3.php`

```
1 <?php
2 $a = true; // ook toegelaten: TRUE, True, TrUe, trUe, ...
3 if ($a)
4     echo "\$a is waar<br>\n";
5 else
6     echo "\$a is niet waar<br>\n";
7
8 $b = 1;
9 $c = $b < 10;
10 if ($c)
11     echo "\$c is waar<br>\n";
12 else
13     echo "\$c is niet waar<br>\n";
14 ?>
```

De cast naar boolean gebeurt via de operaties (boolean) of (bool). De volgende waarden worden naar false gecast:

- de integer met waarde 0
- de double met waarde 0.0
- de lege string en de string met waarde "0"
- Null

Alle andere waarden van het type integer, double of string worden naar true gecast.

Opdracht 1 Zoek het type boolean op in de documentatie ([langref.html](#)). Bekijk de volledige conversie-regels bij omzettingen naar boolean. Bekijk ook de voorbeelden.

Opdracht 2 Schrijf een eenvoudig testprogramma om de conversie naar boolean te testen. Maak de volgende variabelen:

```
$a met waarde 12,
$b met waarde 0,
$c met waarde 1.23,
$d met waarde 0.0,
$e met waarde "0",
$f met waarde "test",
$g met waarde NULL.
```

Een per een cast je elk van deze variabelen naar boolean en plaats het resultaat in de variabele \$test. Druk \$test telkens af.

3.3 Integer

Een geheel getal gelegen tussen -2147483648 ($= -2^{31}$) en 2147483648 ($= 2^{31}$) kan bewaard worden in een variabele van het type integer. Elk getal dat buiten dit interval ligt wordt automatisch beschouwd als een double.

Zoals getoond in `vb4.php` kunnen integers ingevoerd worden in decimale, hexadecimale en octale notatie.

vb4.php

```
1 <?php
2 $a = 1234;           // decimaal
3 var_dump($a);
4 $a = 0123;           // octaal
5 var_dump($a);
6 $a = 0x1A;           // hexadecimaal
7 var_dump($a);
8 ?>
```

Opdracht 3 *Wat is het nut van de functie `var_dump`? Zoek op in de documentatie (`indexes.html`).*

De cast naar integer gebeurt via de operaties (integer) of (int). De volgende regels gelden voor deze operatie:

- een boolean met waarde true wordt naar 1 gecast, een boolean met waarde false naar 0
- een double wordt naar beneden afgerond (12.34 wordt 12)
- een string die niet begint met een cijfer wordt naar 0 omgezet (" `test 123`" wordt 0). Een string die wel begint met een aantal cijfers wordt omgezet naar het overeenkomende getal. Spaties in het begin van de string worden hierbij genegeerd. Ook worden symbolen na de reeks cijfers niet in rekening gebracht. (" `123aad`" wordt 123).
- Null wordt 0

Opdracht 4 *Schrijf een eenvoudig testprogramma om de conversie naar integer te testen.*

3.4 Double (= float)

Algemeen worden double's gebruikt om kommagetallen te bewaren. Deze getallen moeten tussen (plus minus) -1.810^{308} en 1.810^{308} liggen. Getallen die buiten dit interval worden als oneindig beschouwd ($-\text{INF}$ of $+\text{INF}$). De precisie van double's is ongeveer 14 getallen na de komma. In PHP wordt er geen onderscheid gemaakt tussen de types float en double zoals in de programmeertaal Java.

Enkele voorbeelden worden gegeven in `vb5.php` (de letter 'e' slaat op de wetenschappelijke notatie van machten van 10: $12.3\text{E}22 = 12.310^{22}$).

vb5.php

```

1 <?php
2 $a = 1.223;                                // double
3 echo "<br/><br/>$a<br/>\n";
4 var_dump($a);
5
6 $a = -1.23E122;                             // double
7 echo "<br/><br/>$a<br/>\n";
8 var_dump($a);
9
10 $a = 2147483647 ;                          // integer
11 echo "<br/><br/>$a<br/>\n";
12 var_dump($a);
13
14 $a = 2147483648 ;                          // double
15 echo "<br/><br/>$a<br/>\n";
16 var_dump($a);
17
18 $a = 1.22e310 ;                             // double , oneindig
19 echo "<br/><br/>$a<br/>\n";
20 var_dump($a);
21 ?>
```

Conversies gebeuren via de operaties (float) en (double). De conversie-regels zijn analoog aan die van integers.

3.5 String

Variabelen van het type string bevatten een aaneenschakeling van symbolen. Er zijn twee manieren waarop een string-waarde toegekend kan worden, namelijk met enkele of met dubbele aanhalingstekens.

Indien een variabele een waarde krijgt via een reeks karakters tussen enkele aanhalings-tekens dan wordt de reeks karakters zo goed als ongewijzigd in de variabele geplaatst. De enige uitzonderingen hierop zijn de combinaties `\\` en `\'`. Deze combinaties worden in de string geplaatst als het symbool `\` en het symbool `'`.

In het onderstaande voorbeeld wordt het gebruik van strings met enkele aanhalingstekens getoond.

vb6.php

```

1 <?php
2 $a = 'abc'; // inhoud: [abc]
3 $b = 'de$a f'; // inhoud: [de$a f]
4 $c = 'a\'b'; // inhoud: [a'b]
5 $d = '(\)'; // inhoud: [(\)]
6 $e = 'a\nb'; // inhoud: [a\nb]
7 $f = 'a
8 b
9 c';
10
11 echo $a.'<br/>'.$b.'<br/>'.$c.'<br/>'.$d.'<br/>'.$e.'<br/>'.$f.'<br/>';
12 ?>

```

Binnen een string met dubbele aanhalingstekens worden variabelen geëxpandeerd: elke variabele in de string wordt vervangen door zijn inhoud. Een voorbeeld wordt gegeven in onderstaande code:

vb7.php

```

1 <?php
2 $a = "abc"; // inhoud: [abc]
3 $b = "de$a f"; // inhoud: [deabc f]
4 $c = "de${a}f"; // inhoud: [deabcf]
5 $d = "de$a f"; // inhoud: [de]
6
7 echo "$a<br/>$b<br/>$c<br/>";
8 ?>

```

Let in dit voorbeeld op het gebruik van accolades bij de toekenning van de variabele `$c`, `$d = "de$a f"` leidt tot inhoud `[de]` aangezien de variabele `$a` niet gedefinieerd is.

Ook binnen een string met dubbele aanhalingstekens gelden een aantal escape-characters. De belangrijkste escape-characters zijn `\\`, `\"`, `\n`, `\t`, `\$` en `\"`.

```

\\  wordt  \
\"  wordt  "
\n  wordt  een nieuwe lijn
\t  wordt  een tab
\$  wordt  $

```

Een voorbeeld van het gebruik van escape-characters bij strings met dubbele aanhalings-tekenen wordt gegeven in `vb8.php`.

vb8.php

```

1 <?php
2 $a = "a\'b"; // inhoud: [a\'b]
3 // \' is hier geen escape-char
4 $b = "a\"b"; // inhoud: [a"b]
5 $c = "a\tb"; // inhoud: [a b]
6 $d = "a$b "; // inhoud: [a$b ]
7 echo "$a<br/>$b<br/>$c<br/>$d<br/>";
8 ?>

```

De omzetting naar string gebeurt via de cast-operatie (string). Van numerische waarden wordt eenvoudig een string representatie gegeven. De boolean true wordt omgezet naar "1" en de boolean false naar de lege string. Ook Null wordt omgezet naar de lege string. Het is belangrijk op te merken dat het type char niet bestaat in PHP. Enkele symbolen worden bewaard in een string met lengte 1.

Opdracht 5 *Bekijk de code in vb9.php.*

vb9.php

```
1 <?php
2 $a = 'abc';
3 $eerste = $a[0];
4 $laatste = $a[strlen($a)-1];
5 echo "$a heeft lengte " . strlen($a). "<br/>";
6 echo "<br/>het eerste symbool is $eerste<br/>";
7 echo "\$eerste is van het type ". gettype($eerste) . "<br/>";
8 echo "<br/>het laatste symbool is $laatste<br/>";
9 echo "\$laatste is van het type ". gettype($laatste) . "<br/>";
10 ?>
```

Waarom dient strlen? Zoek op in de documentatie.

Opdracht 6 *Lees de informatie over de executie operator (backtick) op de pagina <http://php.net/manual/en/language.operators.execution.php>.*

3.6 Variabele variabelen

Variabelen kunnen ook aangemaakt en gebruikt worden met een variabele naam. De syntax hiervoor is

```
${ string_waarde }
```

In onderstaand voorbeeld wordt de string variabele \$a met inhoud test gebruikt om de variabele \$test aan te maken.

vb10.php

```
1 <?php
2 $a = "test";
3 ${$a} = 12;
4
5 echo ${$a}."<br/>";
6 echo $$a."<br/>";
7 echo $test."<br/>";
8
9 ${$a. "Twee"} = 123;
10 echo $testTwee."<br/>";
11 ?>
```

Aangezien in \${\$a} tussen de accolades één opdracht staat mag dit ook geschreven worden als \$\$a.

In een laatste voorbeeld worden de variabelen \$var_0 t.e.m. \$var_9 aangemaakt via een for lus.

vb11.php

```
1 <?php
2 for ($i = 0; $i < 10; $i++){
3     ${"var_$i"} = $i;
4 }
5
6 for ($i = 0; $i < 10; $i++){
7     echo ${"var_$i"} . "<br/>\n" ;
8 }
9
10 echo $var_5 . "<br/>\n" ;
11 ?>
```


Hoofdstuk 4

Operaties

4.1 De toekenningsoperatie

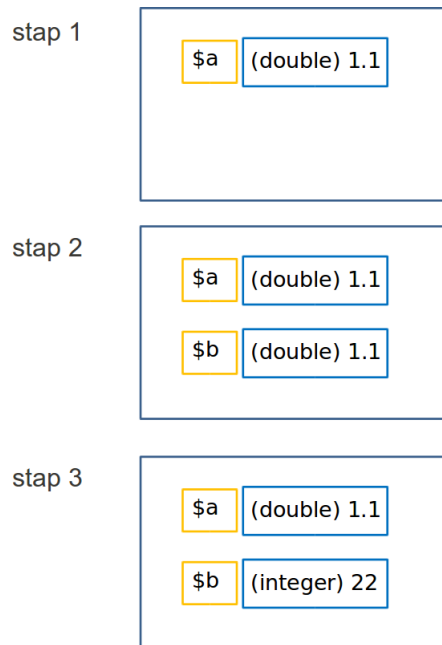
Via de toekenningsoperatie ('=') krijgt een variabele een waarde en wordt ook het type van de variabele bepaald. In het onderstaande voorbeeld worden twee variabelen \$a en \$b gemaakt. \$a krijgt eerst de waarde 1.1. De waarde van \$a wordt vervolgens in de variabele \$b gekopieerd en \$b krijgt tenslotte de waarde 22. Dit voorbeeld toont aan dat \$a en \$b onafhankelijke variabelen blijven ondanks de toekenning \$b = \$a;. Wijzigingen in \$b worden niet gemerkt in \$a.

vb1.php

```
1 <?php
2 $a = 1.1;           // stap 1
3 $b = $a;           // stap 2
4 $b = 22;           // stap 3
5 echo "\$a = $a, \$b = $b "; // resultaat: $a = 1.1, $b = 22
6 ?>
```

In figuur 4.1 wordt een grafische voorstelling van dit voorbeeld gegeven. Let wel dit is een vereenvoudigd beeld van het werkelijke gebruik van geheugen in PHP. Voor een exacte uitleg wordt de lezer verwezen naar de tekst 'References in PHP' ¹.

¹<http://derickrethans.nl/files/phparch-php-variables-article.pdf>



Figuur 4.1: Variabelen en geheugengebruik in `vb1.php`.

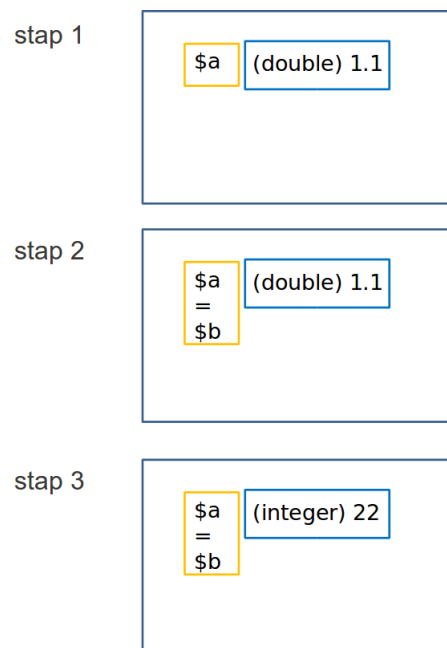
Variabelen kunnen ook een waarde krijgen door toekenning via referentie. Dit gebeurt via de referentie-operator `&`. Zie `vb2.php`.

`vb2.php`

```

1 <?php
2
3 $a = 1.1;                               // stap 1
4 $b =& $a;                               // stap 2
5 $b = 22;                                // stap 3
6 echo "\$a = $a, \$b = $b ";            // resultaat: $a = 22, $b = 22
7
8 ?>
```

Dankzij de regel `$b = &$a;` zijn `$a` en `$b` in weze dezelfde variabele (aliases). De wijziging in `$b` (`$b=22;`) wordt dus ook gemerkt in `$a`. Zie ook figuur 4.2.



Figuur 4.2: Variabelen en geheugengebruik in *vb2.php*.

4.2 Wiskundige operaties

De belangrijkste wiskundige operaties zijn

- + plus
- min
- * maal
- / gedeeld door
- % modulo (rest na deling)

Deze operaties verwachten altijd twee numerische argumenten en het resultaat is steeds numerisch.

- Als bij de bewerkingen `+`, `-` en `*` een van de argumenten een float is (of een string die kan omgezet worden naar een double (bv `"1.33aa"` \rightarrow `1.33`)) dan worden beiden beschouwd als float en is het resultaat een float. Anders worden beide geëvalueerd als integer en is het resultaat een integer.
- Bij een deling worden beide argumenten als float beschouwd en is het resultaat een float. De gehele deling is niet voorzien in PHP maar kan wel via de cast naar integer geïmplementeerd worden.
- Bij de modulo-bewerking worden de argumenten steeds als integer beschouwd en is het resultaat een integer.

Enkele voorbeelden van wiskundige bewerkingen worden in `vb3.php` gegeven.

vb3.php

```

1 <?php
2
3 $a = 1 + 2;           // int (3)
4 // int + int
5
6 $b = 3.1 * 4;         //float(12.4)
7 // float * int       (float) 4 = 4.0
8
9 $c = $b + "a";       // float(12.4)
10 // float + string    string wordt gecast naar float
11 //                  (float)"a" = 0.0
12
13 $d = "1.23aaz" * 12;  //float(14.76)
14 // string * int      "1.23aaz" kan omgezet worden naar float
15 //                  (float) "1.23aaz" = 1.23
16 //                  (float) 12 =12.0
17
18 $e = true + 2.1;      // float(3.1)
19 // boolean + float   (float) true = 1.0
20
21 $f = 1/2;            // float(0.5)
22 // int / int         (float) 1 = 1.0
23 //                  (float) 2 = 2.0
24
25 $g = 3.99 % 1.1;      // int(3)
26 // float % float     (int)3.99 = 3
27 //                  (int) 1.1 = 1
28
29 $h = 1 / "12aa";      // float(0.0833333333333333)
30 // int / string      // (float) 1 =1.0
31 //                  // (float) "12aa" = 12.0
32 ?>

```

Net zoals in Java en C++ bestaan er ook verkorte notaties voor de combinatie van een toekenning en een operatie:

```
+=, -=, *=, /=, %=
++, --
```

Via `$a += $b`; wordt de waarde `$b` opgeteld bij `$a`, via `$a -= 5`; wordt 5 afgetrokken van `$a` etc. Via de increment en decrement operatie (`++`, `--`) wordt 1 opgeteld of afgetrokken van een waarde. Er bestaat zowel een pre-increment en pre-decrement vorm (`++$a` en `-$a`) als een post-increment en post-decrement vorm (`$a++` en `$a--`). De pre-increment of pre-decrement wordt altijd helemaal eerst uitgevoerd, dan volgt de eigenlijke bewerking (waarbij elke `++` en `--` weggelaten worden) en tenslotte wordt de post-increment of post-decrement uitgevoerd. Bijvoorbeeld de bewerking

```
1 $c = ++$a / $b--;
```

komt neer op

```
1 $a = $a + 1;           // EERST pre-increment of decrement
2 $c = $a / $b;          // DAN de eigenlijke bewerking zonder ++ en --
3 $b = $b - 1;           // TENSLOTTE de post-increment of decrement
```

4.3 Tekst-operaties

De belangrijkste tekst-operatie is de concatenatie (`.`) die twee strings aan elkaar verbindt. De argumenten links en rechts van deze operatie worden beide als string beschouwd en indien nodig naar string gecast. Zie ook `vb4.php`:

`vb4.php`

```
1 <?php
2 $a = 'aa';
3 $b = 'bb';
4 $c = 12;
5 $d = true;
6 $e = $a.$b;           // $e = 'aabb'
7 $f = $a.$c;           // $f = 'aa12'
8                       // (string) 12 = '12'
9 $g = $a.$q;           // $g = 'aa'
10                      // (string) NULL = ''
11 $h = $a.$d;           // $h = 'aa1'
12                      // (string) true = '1'
13 echo $e . '<br/>';
14 echo $f . '<br/>';
15 echo $g . '<br/>';
16 echo $h . '<br/>';
17 ?>
```

Via vierkante haken kan een symbool dat zich op een bepaalde index bevindt uit een string geselecteerd worden. Het resultaat is een string met daarin het symbool. Via de functie `strlen` kan de lengte bepaald worden. (zie `vb5.php`).

vb5.php

```

1 <?php
2 $a = 'abc';
3 $eersteSymbool = $a[0];
4 $tweedeSymbool = $a[1];
5 $laatsteSymbool = $a[strlen($a)-1];
6 var_dump($eersteSymbool);
7 ?>%$

```

4.4 Vergelijkingsoperaties

Met behulp van vergelijkingsoperaties worden twee waarden vergeleken. Het resultaat van zo een operatie is steeds een boolean: `true` of `false` naargelang de waarden al dan niet voldoen aan de voorwaarde. De belangrijkste vergelijkingsoperaties worden gegeven in de volgende tabel:

<code>==</code>	is gelijk
<code>===</code>	is gelijk en van hetzelfde type
<code>!=</code>	niet gelijk
<code>!==</code>	is niet gelijk en van hetzelfde type
<code><</code>	kleiner dan
<code>></code>	groter dan
<code><=</code>	kleiner dan of gelijk aan
<code>>=</code>	groter dan of gelijk aan

Er wordt een onderscheid gemaakt tussen de vergelijkingen `==` en `===`. De operatie `==` controleert of twee waarden gelijk zijn, `===` controleert of twee waarden gelijk zijn en hetzelfde type hebben. In het onderstaande voorbeeld is de variabele `$a` een string met inhoud "1" en `$b` een integer met inhoud 1. `$a` en `$b` zijn wel gelijk voor `==` maar niet gelijk voor de vergelijking `===`.

vb6.php

```

1 <?php
2
3 $a = "1";           // string
4 $b = 1;             // integer
5
6 if ($a == $b) echo '$a == $b <br/>';
7 if ($a === $b) echo '$a === $b <br/>';
8 ?>

```

De operatie `!=` controleert of twee waarden verschillend zijn, `!==` controleert of twee waarden verschillen of van een ander type zijn.

De operatoren `<`, `>`, `<=` en `>=` worden meestal gebruikt om twee getallen met elkaar te vergelijken. Ze kunnen ook gebruikt worden om twee tekstvariabelen alfabetisch te vergelijken. Indien een tekstvariabele met een numerische variabele vergeleken wordt dan wordt de tekstvariabele naar een getal gecast. Zie `vb7.php`

vb7.php

```

1 <?php
2 $a = 1;
3 $b = 2.1;
4 $c = "aap";
5 $d = "noot";
6
7 if ($a < $b)      echo '$a &lt; $b <br/>';
8 if ($c < $d)      echo '$c &lt; $d <br/>';
9 if ($a < $c)      echo '$a &lt; $c <br/>'; // (int) "aap" = 0
10 ?>

```

4.5 Logische operaties

De volgende operaties kunnen worden toegepast op variabelen van het type boolean:

<code>!</code>	niet
<code>&</code>	en
<code>&&</code>	kortsluiting-en (short-circuit and)
<code> </code>	of
<code> </code>	kortsluiting-of (short-circuit or)

`!` is de negatie-operatie:

```

! true  = false
! false = true

```

`&` is de en-operatie en `|` de of-operatie:

```

true  & true   = true
true  & false  = false
false & true   = false
false & false  = false

true  | true   = true
true  | false  = true
false | true   = true
false | false  = false

```

Bij de kortsluitingsvorm van de en-operatie en de of-operatie wordt soms het tweede argument (rechts van `&&` of `||`) niet uitgevoerd.

Bij de `&&`-operatie wordt eerst gekeken of het eerste argument true is en enkel als dit zo is wordt het tweede argument gecontroleerd. Als het eerste argument false is heeft het geen zin om het tweede te controleren: ongeacht het tweede argument is het resultaat steeds false. Hetzelfde geldt voor `||`: er wordt eerst gekeken naar het eerste argument. Enkel indien dit argument false is wordt er gekeken naar het tweede argument. Als het eerste argument true is heeft het geen zin om naar het tweede argument te kijken: (true of false is true, true of true = true).

4.6 Oefeningen

Oefening 1 *Verklaar de uitvoer van het onderstaande programma.*

```

1 <?php
2
3 $a = 'a' + 'b';
4 echo '$a = ' . $a . '<br/>';
5
6 $b = 7 % 3.5;
7 echo '$b = ' . $b . '<br/>';
8
9 $c = 0;
10 $d = 2;
11 $c -= 1 - $d;
12
13 echo '$c = ' . $c . '<br/>';
14 echo '$d = ' . $d . '<br/>';
15
16 $e = 1;
17 $f = 10;
18 $g = $e++ * --$f;
19
20 echo '$e = ' . $e . '<br/>';
21 echo '$f = ' . $f . '<br/>';
22 echo '$g = ' . $g . '<br/>';
23
24 $h = 1;
25 if (++$h == 2 && $h++ == 2 )
26     echo 'a<br/>';
27 else
28     echo 'b<br/>';
29 echo '$h = ' . $h . '<br/>';
30
31 ?>

```


Hoofdstuk 5

Controle structuren

In dit hoofdstuk worden de selectie, de while-lus en de for-lus besproken.

5.1 De selectie

5.1.1 Algemeen

De meest algemene vorm van een selectie heeft de volgende syntax:

```
if (voorwaarde){
    sequentie1
}
else {
    sequentie2
}
```

Via deze structuur wordt gecontroleerd of er aan een bepaalde voorwaarde voldaan is. Indien dit zo is dan wordt sequentie1 uitgevoerd anders wordt sequentie2 uitgevoerd. Zoals hieronder getoond wordt is het else-deel optioneel:

```
if (voorwaarde){
    sequentie1
}
```

In `vb1.php` worden twee getallen vergeleken en een bijhorende opdracht afgedrukt.

`vb1.php`

```
1 <?php
2 $a = 1;
3 $b = 2;
4 if ($a < $b){
5     $c = "$a is kleiner dan $b";
6 }
7 else {
8     $c = "$a is groter dan $b";
9 }
10 echo $c. '<br/>';
11 ?>
```

In dit voorbeeld bestaan de sequenties tussen de accolades uit één opdracht (de toekenning `$c = ...`). De accolades mogen bijgevolg weggelaten worden.

`vb2.php`

```
1 <?php
2 $a = 1;
3 $b = 2;
4 if ($a < $b)
5     $c = "$a is kleiner dan $b";
6 else
7     $c = "$a is groter dan $b";
8 echo $c. '<br/>';
9 ?>
```

Het voorbeeld kan ook zonder `else` geschreven worden:

`vb3.php`

```
1 <?php
2 $a = 1;
3 $b = 2;
4 $c = "$a is groter dan $b";
5 if ($a < $b)
6     $c = "$a is kleiner dan $b";
7 echo $c. '<br/>';
8 ?>
```

5.1.2 Verkorte notatie

Er bestaat ook een verkorte notatie voor de selectie. Deze wordt meestal gebruikt om een waarde toe te kennen aan een variabele:

```
$res = (voorwaarde) ? waarde1 : waarde2 ;
```

Indien voldaan is aan de voorwaarde krijgt de variabele **\$res** de waarde **waarde1** anders krijgt **\$res** de waarde **waarde2**. **vb1.php** kan verkort geschreven worden als **vb4.php**.

vb4.php

```
1 <?php
2 $a = 1;
3 $b = 2;
4 $c = ($a<$b) ? "$a is kleiner dan $b" : "$a is groter dan $b";
5 echo $c. '<br/>';
6 ?>
```

5.1.3 De meervoudige selectie

Via een meervoudige selectie kunnen meerdere voorwaarden gecontroleerd worden. De syntax van deze structuur wordt gegeven door:

```
if (voorwaarde1){
    sequentie1
}
elseif (voorwaarde2) {
    sequentie2
}

elseif (voorwaarde3) {
    sequentie3
}

...

else{
    sequentieN
}
```

In deze structuur wordt eerst voorwaarde1 gecontroleerd. Als aan deze voorwaarde voldaan is dan wordt sequentie1 uitgevoerd. Als niet voldaan is aan voorwaarde1 dan wordt voorwaarde2 gecontroleerd. Als hieraan voldaan is wordt sequentie2 uitgevoerd anders wordt voorwaarde3 gecontroleerd etc. Als aan geen van de van de voorwaarden voldaan is wordt sequentieN uitgevoerd. Zie **vb5.php**.

vb5.php

```
1 <?php
2 $a = 1;
3 $b = 2;
4 if ($a < $b) {
5     $c = '$a is kleiner dan $b';
6 }
7 elseif ($a == $b) {
8     $c = '$a is gelijk aan $b';
9 }
10 else {
11     $c = '$a is groter dan $b';
12 }
13 echo $c;
14 ?>
```

Opdracht 1 *Herschrijf het bovenstaande voorbeeld als een geneste selectie.*

Opdracht 2 *Zoek in de documentatie op hoe een switch werkt ([langref.html](#)).*

5.2 De while-lus

Via een while-lus kan een sequentie meerdere keren uitgevoerd worden. Een eerste vorm van de while-lus wordt gegeven door de onderstaande syntax:

```
while (voorwaarde){
    sequentie;
}
```

De sequentie wordt uitgevoerd zolang aan de voorwaarde voldaan is. De voorwaarde wordt telkens getest voor de sequentie uitgevoerd wordt. Als bij de eerste iteratie niet aan de voorwaarde voldaan is, wordt de lus niet uitgevoerd.

Er bestaat ook een while-lus waarbij de controle achteraan gebeurt:

```
do{
    sequentie;
}
while (voorwaarde);
```

De sequentie wordt opnieuw uitgevoerd zolang er aan de voorwaarde voldaan is. De voorwaarde wordt telkens getest nadat de sequentie uitgevoerd wordt. De lus wordt dus minstens 1 maal uitgevoerd.

In het onderstaande voorbeeld wordt het kwadraat van de getallen gelegen tussen 0 en 5 berekend en afgedrukt

vb6.php

```
1 <?php
2 $i =0;
3 while ($i < 5){
4     echo $i * $i . '<br/>';
5     $i++;
6 }
7 ?>
```

Opdracht 3 *Herschrijf bovenstaand voorbeeld m.b.v. een do-while.*

5.3 De for-lus

Ook via een for-lus kan een sequentie meerdere keren herhaald worden. Algemeen ziet de syntax van een for-lus er als volgt uit:

```
for (initialisatie; voorwaarde; incrementatie){
    sequentie
}
```

Het initialisatiegedeelte wordt 1 keer uitgevoerd voor de lus start, meestal krijgt een getal hier een initiële waarde (e.g. `$i=0`). De sequentie wordt uitgevoerd zolang aan de voorwaarde voldaan is. De voorwaarde wordt telkens voor de sequentie gecontroleerd (e.g. `$i < 10`). Na de uitvoering van elke sequentie wordt de incrementatie uitgevoerd (e.g. `$i++`, `$i+=2`).

Een voorbeeld van het gebruik van de for-lus wordt in `vb7.php` gegeven.

vb7.php

```
1 <?php
2 for ($i =0;$i <5;$i++) {
3     echo $i * $i . '<br/>';
4     $i++;
5 }
6 ?>
```

5.4 Oefeningen

Oefening 1 *Stel de vermenigvuldigingstabel op die getoond wordt in figuur 5.1. Zorg er voor dat de grootte van de tabel eenvoudig veranderd kan worden (voorzie een variabele `$grootte`. In figuur 5.1 werd regel code `$grootte = 5`; gebruikt).*

Vermenigvuldigingstabel

1 * 1 = 1	1 * 2 = 2	1 * 3 = 3	1 * 4 = 4	1 * 5 = 5
2 * 1 = 2	2 * 2 = 4	2 * 3 = 6	2 * 4 = 8	2 * 5 = 10
3 * 1 = 3	3 * 2 = 6	3 * 3 = 9	3 * 4 = 12	3 * 5 = 15
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12	4 * 4 = 16	4 * 5 = 20
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15	5 * 4 = 20	5 * 5 = 25

Figuur 5.1: De vermenigvuldigingstabel voor `$grootte = 5`.

Oefening 2 *Voorzie een variabele `getal` (`$getal = ...;`) Controleer of `getal`*

- *negatief en even is*
- *negatief en oneven is*
- *tussen 0 en 10 ligt*
- *groter dan 10 is*

en druk een corresponderende boodschap af. Maak hierbij gebruik van genestelde selecties.

Oefening 3 *Bereken de faculteit van een `getal` (bv `$getal = 10;`) en druk het resultaat af. Gebruik hiervoor eerst een `for`-lus en in een tweede versie een `while`-lus.*

Oefening 4 *Lees de teksten*

- <http://www.php.net/manual/en/control-structures.alternative-syntax.php>
- <http://www.php.net/manual/en/control-structures.while.php>
- <http://www.php.net/manual/en/control-structures.for.php>

Probeer de voorbeelden i.v.m. alternatieve syntax uit.

Hoofdstuk 6

Rijen

6.1 Creatie

In PHP is een array een datatype waarin een aantal waarden (values) bewaard kunnen worden. Bij elke waarde in de array is er een sleutel (key) voorzien waarmee de waarde opgevraagd kan worden. (In veel programmeertalen wordt deze datastructuur een 'map' genoemd.)

Een array kan aangemaakt worden via de functie array:

```
$a = array(element_1, element_2, ... ,element_N );
```

```
    waarbij      element_i = value_i
    of           element_i = key_i => value_i
```

In deze definitie wordt de rij \$a bestaande uit N elementen aangemaakt. Per element wordt ofwel enkel de waarde (value_i) gespecificeerd ofwel wordt de combinatie van sleutel (key_i) en waarde (value_i) opgegeven.

De waarden (values) in een array hoeven niet allemaal van hetzelfde type te zijn: in eenzelfde rij kunnen waarden van verschillende types bewaard worden. De sleutel (key) moet van het datatype integer of string zijn. Als er geen key gespecificeerd wordt bij het aanroepen van de functie array dan wordt de eerstvolgende beschikbare integer genomen als key.

Enkele voorbeelden van de creatie van arrays worden gegeven in `vb1.php`.

`vb1.php`

```

1 <?php
2 $a = array(1, 2, 3, 4);
3 $b = array(1, 2.1, true, "ja");
4 $c = array(1 => 12, "Juist" => true);
5 $d = array (2 => 1, 10);
6
7 /* onderstaande code toont de creatie en de var_dump van elke rij
8    het resultaat wordt getoond in een tabel */
9 echo '<table border="1">';
10 echo '<tr><td>$a = array(1,2,3,4);</td><td> ';
11 var_dump($a);
12 echo '</td></tr>';
13 echo '<tr><td>$b = array(1,2.1,true,"ja");</td><td> ';
14 var_dump($b);
15 echo '</td></tr>';
16 echo '<tr><td>$c = array(1=> 12, "Juist" => true);</td><td> ';
17 var_dump($c);
18 echo '</td></tr>';
19 echo '<tr><td>$d = array (2 => 1, 10);</td><td> ';
20 var_dump($d);
21 echo '</td></tr>';
22 echo '</table>';
23 ?>

```

Het resultaat van `vb1.php` wordt getoond in figuur 6.1.

<code>\$a = array(1,2,3,4);</code>	<pre> array 0 => int 1 1 => int 2 2 => int 3 3 => int 4 </pre>
<code>\$b = array(1,2.1,true,"ja");</code>	<pre> array 0 => int 1 1 => float 2.1 2 => boolean true 3 => string 'ja' (length=2) </pre>
<code>\$c = array(1=> 12, "Juist" => true);</code>	<pre> array 1 => int 12 'Juist' => boolean true </pre>
<code>\$d = array (2 => 1, 10);</code>	<pre> array 2 => int 1 3 => int 10 </pre>

Figuur 6.1: `vb1.php` getoond in Mozilla Firefox.

Bij de creatie van de rijen `$a` en `$b` werden enkel waarden (values) ingegeven. De key die bij elke value hoort wordt automatisch gegenereerd, deze keys gaan in beide gevallen van 0 tot 3. Voor de rij `$c` werd er bij elk element zowel een key als een waarde gespecificeerd.

De key 1 bij de waarde 12 en de key "Juist" bij de waarde true. Bij de definitie van de rij \$d werd zowel een key,value-paar (key 2 en value1) als een value (10) ingegeven. De value 10 krijgt key 3 omdat 3 de eerstvolgende beschikbare integer key is (na de key 2).

Opdracht 1 *Maak de rij \$a. Deze rij heeft de volgende key, value paren:*

key	value
"Ja"	true
"Juist"	true
1	true
"Mis"	false
"Nee"	false
0	false

Bekijk het resultaat met behulp van de functie var_dump.

Opdracht 2 *Maak de rij \$b. Deze rij heeft de volgende key, value paren:*

key	value
0	"nul"
1	"een"
2	"twee"
...	...
9	"negen"

Bekijk het resultaat met behulp van de functie print_r (zoek deze functie op in de documentatie).

6.2 Gebruik

Na de creatie van een array kunnen waarden (values) opgehaald en gewijzigd worden aan de hand van hun sleutels (keys). Een waarde uit de rij wordt geselecteerd via vierkante haken:

```
$rij[ key_i ]
```

selecteert de value die overeenkomt met key_i. Een voorbeeld wordt gegeven in `vb2.php`.

vb2.php

```

1 <?php
2 $a = array(1, 2, 3, 4);
3 echo "$a[0]<br/>";
4 $a[0] = "ja";
5
6 $b = array("een" => 1, "twee" => 3 );
7 echo $b["een"].'<br/>';
8 echo "$b[een]<br/>";
9 $b["twee"] = 2;
10 ?>

```

In dit voorbeeld wordt de waarde met key 0 van de rij \$a eerst afgedrukt en daarna vervangen door de string "ja"; Van de rij \$b wordt de waarde met key 'een' afgedrukt (let op het gebruik van "). De waarde met key 'twee' wordt vervolgens vervangen door 2.

Zoals getoond in vb3.php kunnen na de creatie van de rij nog steeds elementen bij de rij toegevoegd worden.

vb3.php

```

1 <?php
2 $a = array(1, 2, 3, 4);
3 $a[7] = 11;
4 var_dump($a); echo '<br/>';
5
6 $b = array("een" => 1, "twee" => 3 );
7 $b["drie"] = 3;
8 var_dump($b); echo '<br/>';
9 ?>

```

Bij de rij \$a wordt de value 11 met als key 7 toegevoegd, bij de rij \$b wordt de value 3 met als key "drie" geplaatst.

Ook via lege vierkante haken kan een waarde toegevoegd worden:

```
$rij [ ] = value_i;
```

De waarde value_i wordt in de rij geplaatst met als key de eerstvolgende vrije index. Zie vb4.php.

vb4.php

```

1 <?php
2 $c = array(1, "a" =>2);
3 $c[] ="nieuw";
4 var_dump($c);
5 ?>

```

In dit voorbeeld wordt de waarde "nieuw" toegevoegd met 1 als key (de key 0 was al ingenomen door de waarde 1).

Elementen kunnen ook verwijderd worden uit de rij via de functie `unset`. Zie ook `vb5.php`.

`vb5.php`

```
1 <?php
2 $d = array(1, "a" =>2, "b");
3 unset($d["a"]);
4 var_dump($d); echo '<br/>';
5 ?>
```

Opdracht 3 *Gegeven de rij \$a uit opdracht 1. Voeg na de creatie de volgende key, value paren toe aan \$a*

```
key      value
"correct" true
"verkeerd" false
```

Bekijk het resultaat met behulp van de functie `var_dump`.

Opdracht 4 *Gegeven de rij \$b uit opdracht 2. Voeg na de creatie de volgende key, value paren toe aan \$b*

```
key      value
10       "tien"
11       "11"
```

Bekijk het resultaat met behulp van de functie `print_r`.

6.3 Nuttige functies

Enkele nuttige functies die kunnen toegepast worden op een array worden hieronder opgesomd:

<code>count(\$a)</code>	geef het aantal waarden in de rij \$a
<code>array_keys(\$a)</code>	geef een rij met alle keys van \$a
<code>array_keys(\$a, val)</code>	geef een rij met de keys die overeenkomen met de waarde val (zoekfunctie)
<code>array_values(\$a)</code>	geef een rij met alle values van \$a
<code>print_r(\$a)</code>	print de rij \$a
<code>sort(\$a)</code>	sorteer de rij \$a
<code>shuffle(\$a)</code>	shud de rij \$a
<code>min(\$a)</code>	minimum van de rij \$a
<code>max(\$a)</code>	maximum van de rij \$a

Een voorbeeld van het gebruik van deze functies wordt in `vb6.php` gegeven.

`vb6.php`

```

1 <?php
2 $a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");
3 $keys = array_keys($a);
4 $keysMa = array_keys($a, "ma");
5 $vals = array_values($a);
6 $grootte = count($a);
7
8 print_r($a); echo '<br/>';
9 print_r($keys); echo '<br/>';
10 print_r($keysMa); echo '<br/>';
11 print_r($vals); echo '<br/>';
12 echo $grootte; echo '<br/>';
13
14 sort ($a);
15 print_r($a); echo '<br/>';
16
17 shuffle($a);
18 print_r($a); echo '<br/>';
19 ?>

```

Op het einde van het programma hebben de variabelen `$keys`, `$keysMa`, `$vals` en `$grootte` de volgende waarden:

```

$keys      Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )
$vals      Array ( [0] => ma [1] => di [2] => wo [3] => ma )
$keysMa    Array ( [0] => 1 [1] => 4 )
$grootte   4

```

Opdracht 5 *Gegeven de rij \$a*

```
$a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");
```

Schrijf een programma dat de volgende uitvoer heeft:

```

sleutel = 1, waarde = ma
sleutel = 2, waarde = di
sleutel = 3, waarde = wo
sleutel = 4, waarde = ma

```

Maak hierbij gebruik van een for-lus en de functies `count`, `array_vals` en `array_keys`.

6.4 Foreach

foreach een handige structuur die gebruikt kan worden om door de waarden (en de sleutels) van een rij te gaan. Via een eerste versie van de foreach-lus wordt elke waarde van de rij `$rij` in de variabele `$v` gekopieerd.

```
foreach ($rij as $v) {
    sequentie
}
```

In `vb7.php` wordt elke waarde uit de rij in de variabele `$v` gekopieerd en deze variabele wordt afgedrukt:

`vb7.php`

```
1 <?php
2 $a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");
3 foreach ($a as $v) {
4     echo "value: $v <br/>\n";
5 }
6 ?>
```

Via een tweede vorm van de foreach-lus is zowel de waarde (`$v`) als de sleutel (`$k`) van elk element uit de rij beschikbaar in de sequentie.

```
foreach ($rij as $k => $v) {
    sequentie
}
```

Opdracht 6 *Gegeven de rij \$a*

```
$a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");
```

Schrijf een programma dat elke waarde en elke sleutel van deze rij afdrukt.

Omdat de variabele `$v` een copy van de waarde in de rij bevat worden wijzigingen `$v` niet gemerkt in de rij zelf. Wijzigingen laten doorvoeren in de rij wordt gedaan via de `&`-operatie.

```
foreach ($rij as &$amp;$v) {
    sequentie
}
```

of

```
foreach ($rij as $k => &$amp;$v) {
    sequentie
}
```

Zie ook `vb8.php`.

vb7.php

```
1 <?php
2 $a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");
3
4 foreach ($a as $v) {
5     $v = -1;
6 }
7 var_dump ($a); echo '<br/>';
8
9 foreach ($a as &$v) {
10     $v = -1;
11 }
12 var_dump ($a); echo '<br/>';
13 ?>
```

In de eerste foreach-lus worden de wijziging in `$v` niet doorgevoerd in de rij `$a`, in de tweede foreach-lus wel.

Hoofdstuk 7

Functies

7.1 Algemeen

Een functie is een sequentie van opdrachten die voorzien is van een naam. Deze sequentie wordt uitgevoerd telkens de functie aangeroepen wordt. Verder kan een functie een reeks argumenten meekrijgen, dit zijn variabelen die in de sequentie beschikbaar zijn. Ook kan een functie een waarde terugkeren.

Een functie wordt gedeclareerd als

```
function naam (argumenten){  
    sequentie  
}
```

via het keyword 'return' wordt aangegeven of een waarde moet teruggekeerd worden. Indien geen return-waarde gespecificeerd wordt, wordt NULL teruggegeven.

In het onderstaande voorbeeld heeft de functie printBoodschap geen terugkeerwaarde en wordt er in de functie som wel een waarde teruggekeerd.

vb1.php

```
1 <?php  
2 function printBoodschap($boodschap){  
3     echo "$boodschap<br/>\n";  
4 }  
5  
6 function som ($a, $b){  
7     return $a + $b;  
8 }  
9  
10 printBoodschap ("test");  
11 $a = 1;  
12 $b = som($a,2);  
13 echo $b;  
14 ?>
```

Elke naam kan slechts één keer gebruikt worden om een functie te definiëren. In het onderstaande voorbeeld leidt het hergebruik van de naam som (een keer met twee parameters, een keer met drie parameters) tot de foutmelding: "Cannot redeclare som() ..."

vb2.php

```
1 <?php
2 function som ($a, $b){
3     return $a + $b;
4 }
5
6 // onderstaande regel leidt tot een foutmelding !
7 function som ($a, $b, $c){
8     return $a + $b+$c;
9 }
10
11 $a = 1;
12 $b = som($a, 2);
13 $c = som(1, 2, 3);
14 echo "$b<br/>$c";
15 ?>
```

Er zijn twee manieren hoe men dit probleem kan oplossen: de eerste methode bestaat erin een **default argument** voor de functie som te voorzien. Een default waarde is een argument waarvan een waarde wordt voorzien. In het onderstaand voorbeeld heeft het argument \$c default waarde 0. De functie kan worden aangeroepen met twee argumenten (\$c is dan gelijk aan 0), maar hij kan ook aangeroepen met drie argumenten (\$c krijgt dan de waarde van het argument)

vb3.php

```
1 <?php
2 function som ($a, $b, $c = 0){
3     return $a + $b + $c;
4 }
5
6 $a = 1;
7 $b = som($a, 2);
8 $c = som(1, 2, 3);
9 echo "$b<br/>$c";
10 ?>
```


Een tweede oplossing bestaat er in geen argument in te geven bij de declaratie en gebruik te maken van de functies `func_get_args`, `func_get_arg` en `func_num_args`.

vb4.php

```

1 <?php
2 function som (){
3     $args = func_get_args();
4     $s =0;
5     foreach ($args as $v){
6         $s += $v;
7     }
8     return $s;
9 }
10
11 $a = 1;
12 $b = som($a,2);
13 $c = som(1,2,3);
14 echo "$b<br/>$c";
15 ?>

```

Opdracht 1 Zoek de functies `func_get_arg`, `func_get_args` en `func_num_args` op in de documentatie.

Opdracht 2 Schrijf een programma waarin de functie `printBoodschap` gedeclareerd wordt. Deze functie heeft een argument `$tekst` en een argument `$symbool` dat als default waarde `"` heeft. De functie wordt twee keer aangeroepen in het programma:

```

printBoodschap ("test zonder kadersymbool");
printBoodschap ("test met kadersymbool", "*");

```

Het resultaat wordt getoond in figuur 7.1.

```

test zonder kadersymbool

*****
*test met kadersymbool*
*****

```

Figuur 7.1: Het resultaat van het programma beschreven in opdracht 2.

Maak in de functie gebruik van de HTML-tag

```
<p style="font-size: 12px; font-family: monospace;">
```

Opdracht 3 Lees de tekst *Variable functions* op de pagina <http://www.php.net/manual/en/functions.variable-functions.php>.

7.2 Pass by reference

Variabelen worden normaal pass by value doorgegeven: een wijziging van een argument van de functie wordt niet gemerkt in het hoofdprogramma. In het onderstaande voorbeeld wordt het argument `$arg` gelijk gesteld aan `null`. Deze wijziging wordt niet gemerkt in de variabele `$a` van het hoofdprogramma.

vb5.php

```
1 <?php
2 function maakNull ($arg){
3     $arg= null;
4 }
5
6 $a =1;
7 maakNull($a);
8 var_dump($a);
9 ?>
```

Via de `&`-operator wordt een variabele pass by reference doorgegeven worden.

vb6.php

```
1 <?php
2 function maakNull (&$arg){
3     $arg= null;
4 }
5
6 $a =1;
7 maakNull($a);
8 var_dump($a);
9 ?>
```

De wijziging in het argument wordt nu wel gemerkt in de variabele in het hoofdprogramma (`$arg` en `$a` zijn aliases).

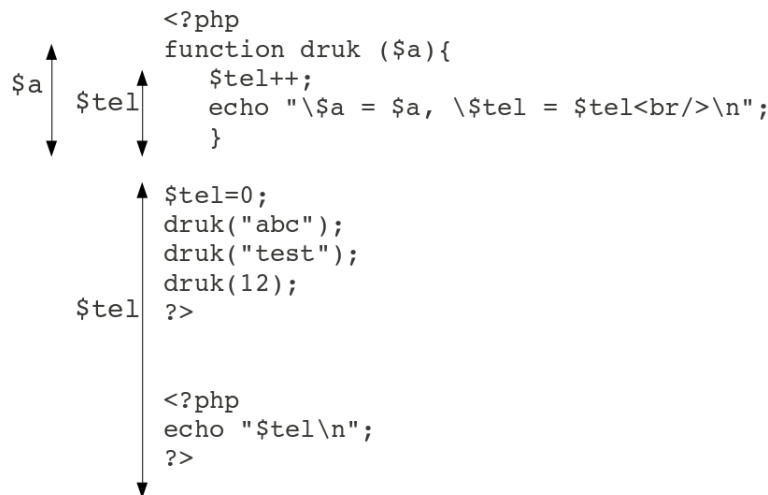
Opdracht 4 *Schrijf een programma waarin de functie `wisselOm` gedeclareerd wordt. Deze functie heeft twee argumenten `$a` en `$b`, deze worden pass by reference doorgegeven. Via de functie `wisselOm` worden de variabelen omgewisseld. Test de functie in het hoofdprogramma.*

Opdracht 5 *Een functie kan ook een referentie teruggeven. Lees de tekst 'Returning References' op de pagina <http://php.net/manual/en/language.references.return.php>*

7.3 De scope van variabelen

De scope beschrijft de levensduur van een variabele. In figuur 7.2 worden enkele voorbeelden gegeven (zie ook `vb7.php`). De scope van het argument `$a` omvat het bereik van heel de functie druk. De variabele `$tel` wordt enerzijds gebruikt in het hoofdprogramma

en anderzijds als lokale variabele in de functie druk. Zoals getoond in de figuur loopt de scope van de variabele \$tel uit het hoofdprogramma voorbij de sluitende PHP-tag ?> en kan deze variabele gebruikt worden in volgende PHP-tags. De scope van de variabele \$tel in de functie loopt van waar de variabele eerst gebruikt wordt tot het eind van de functie.



Figuur 7.2: Een voorbeeld van de scope van variabelen (vb7.php).

De twee variabelen \$tel hebben wel dezelfde naam maar voor de rest bestaat er geen verband tussen beide variabelen. De ene bestaat in de functie, de andere in het hoofdprogramma. Telkens de functie druk aangeroepen wordt krijgt \$tel de waarde NULL, bij deze waarde wordt 1 opgeteld ((int) NULL =0) en 1 wordt afgedrukt. De uitvoer wordt gegeven door

```

$a = abc, $tel = 1
$a = test, $tel = 1
$a = 12, $tel = 1
0

```

7.4 Global

Binnen een functie kan aangeduid worden dat een variabele als globaal beschouwd moet worden. Deze variabele wordt dan als dezelfde beschouwd als gelijk aan de variabele met dezelfde naam die in het hoofdprogramma gebruikt wordt. In het onderstaande voorbeeld wordt \$tel als globaal gedefinieerd.

vb8.php

```
1 <?php
2 function druk ($a){
3     global $tel;
4     $tel++;
5     echo "\$a = $a, \$tel = $tel<br/>\n";
6 }
7
8 $tel = 0;
9 druk("abc");
10 druk("test");
11 druk(12);
12 ?>
13
14 <?php
15 echo "$tel\n";
16 ?>
```

De uitvoer van vb8.php wordt gegeven door

```
$a = abc, $tel = 1
$a = test, $tel = 2
$a = 12, $tel = 3
3
```

7.5 Static

Een static variabele wordt gedefinieerd in een functie. De waarde van een static variabele blijft behouden en kan dus gebruikt worden telkens de functie aangeroepen wordt.

In `vb9.php` wordt de static variabele `$tel` gebruikt om te tellen hoeveel keer de functie druk aangeroepen wordt.

`vb9.php`

```

1 <?php
2 function druk ($a){
3     static $tel =0;
4     $tel++;
5     echo "\$a = $a, \$tel = $tel<br/>\n";
6 }
7
8 $tel = 0;
9 druk("abc");
10 druk("test");
11 druk(12);
12 ?>
13
14 <?php
15 echo "$tel\n";
16 ?>

```

Het resultaat van dit programma wordt gegeven door

```

$a = abc, $tel = 1
$a = test, $tel = 2
$a = 12, $tel = 3
0

```

7.6 Recursie

Een recursieve functie is een functie die zichzelf aanroept. In `vb10.php` is de functie `faculteit` een recursieve functie: de faculteit van een getal groter dan 1 kan berekend worden als dat getal maal de faculteit van het getal min 1:

```

faculteit(5) = 5 * 4 * 3 * 2 * 1
             = 5 * faculteit(4)

faculteit(4) = 4 * 3 * 2 * 1
             = 4 * faculteit(3)
...

faculteit(0) = 1.

```

vb10.php

```

1 <?php
2 function faculteit($a){
3     if (is_numeric($a) && $a >=0){
4         if ($a >1){
5             return $a*faculteit($a-1);
6         }
7         else{
8             return 1;
9         }
10    }
11    else{
12        return "fout";
13    }
14 }
15
16 echo faculteit(5);
17 ?>

```

Ook de functie `printDirectory` in `vb11.php` is een recursieve functie. Via `printDirectory` worden alle bestanden en subdirectories van een directory afgedrukt. Ook wordt voor elke subdirectory opnieuw de functie `printDirectory` aangeroepen. Op deze manier wordt een overzicht van alle bestanden en directories onder een directory gegeven.

vb11.php

```

1 <?php
2 function printDirectory($dir){
3     $dir = realpath($dir);
4     $handle = opendir($dir);
5     $f = readdir($handle);
6     while ($f !== false) {
7         if ($f !== "." && $f !== ".."){
8             $f = $dir . "/" . $f;
9             echo htmlentities($f, ENT_QUOTES, 'ISO-8859-1') . "<br/>\n";
10            if(is_dir($f)){
11                printDirectory($f);
12            }
13        }
14        $f = readdir($handle);
15    }
16 }
17
18 printDirectory('..');
19 ?>

```

7.7 Voorgedefinieerde functies

In deze sectie worden enkele categorieën van voorgedefinieerde functies besproken. Voor meer informatie wordt de lezer verwezen naar de documentatie ([funcref.html](#)).

7.7.1 Wiskundige functies

Zie ook `php_doc/html/book.math.html`.

<code>pow(\$g, \$m)</code>	machtsverheffing $\$g$ tot de macht $\$m$
<code>sqrt(\$g)</code>	vierkantswortel van $\$g$
<code>sin(\$g)</code>	sinus van $\$g$
<code>cos(\$g)</code>	cosinus van $\$g$
<code>tan(\$g)</code>	tangens van $\$g$
<code>exp(\$g)</code>	exponent van $\$g$
<code>log(\$g)</code>	(natuurlijk) logaritme van $\$g$
<code>log10(\$g)</code>	tientallig logaritme van $\$g$
<code>round(\$g)</code>	$\$g$ afgerond $12.6 \rightarrow 13$
<code>round(\$g, \$i)</code>	$\$g$ afgerond tot op de $\$i^e$ decimaal
<code>floor(\$g)</code>	$\$g$ afgekapt $12.6 \rightarrow 12$
<code>mt_rand(\$min, \$max)</code>	willekeurig geheel getal gelegen tussen $\$min$ en $\$max$

Opdracht 6 $\$a = 12.32$, bereken $\sqrt{4\$a^3}$ en rond af op de 2e decimaal.

Opdracht 7 Bekijk `mt_rand` in de documentatie. Gebruik deze functie om 50 willekeurige kommagetallen tussen 0 en 5 te generen en af te drukken. De kommagetallen hebben 1 cijfer na de komma. De uitvoer ziet er uit als volgt:

```
3.7
4.3
0.6
...
```

7.7.2 String functies

Zie ook `ref.strings.html`.

<code>strlen(\$s)</code>	lengte van de string <code>\$s</code>
<code>strpos(\$s, \$z)</code>	- eerste positie van een zoekstring <code>\$z</code> in de string <code>\$s</code> - false als <code>\$s</code> niet gevonden wordt
<code>strpos(\$s, \$z,\$i)</code>	- positie van een zoekstring <code>\$z</code> in de string <code>\$s</code> , er wordt gezocht vanaf positie <code>\$i</code> - false als <code>\$s</code> niet gevonden wordt
<code>substr(\$s, \$i)</code>	substring van de string <code>\$s</code> , beginnende vanaf positie <code>\$i</code>
<code>substr(\$s, \$i, \$l)</code>	substring met lengte <code>\$l</code> van de string <code>\$s</code> , beginnende vanaf positie <code>\$i</code>
<code>strtolower(\$s)</code>	geef de string <code>\$s</code> omgezet naar kleine letters
<code>strtoupper(\$s)</code>	geef de string <code>\$s</code> omgezet naar hoofdletters
<code>trim(\$s)</code>	geef de string <code>\$s</code> zonder spaties voor en achteraan
<code>print(\$s)</code>	druk de string <code>\$s</code> af
<code>printf(\$s)</code>	druk <code>\$s</code> af in een aangepast formaat zie <code>php_doc/htmlfunction.printf.html</code>
<code>sprintf (\$s)</code>	geef de string <code>\$s</code> in een aangepast formaat zie <code>php_doc/html/function.sprintf.html</code>

Opdracht 8 *Bekijk de voorbeelden bij de functie `sprintf` en `printf`. Druk het getal 457423 af in binaire, hexadecimale en octale vorm.*

Opdracht 9 *Druk alle posities van het plus-teken in `$a` af.*

```
$a = "+5*3+4=19=1+18";
```

Maak hierbij gebruik van `strpos`.

7.7.3 Array functies

Zie sectie 6.3.

<code>count(\$a)</code>	geef het aantal waarden in de rij <code>\$a</code>
<code>array_keys(\$a)</code>	geef een rij met alle keys van <code>\$a</code>
<code>array_keys(\$a, val)</code>	geef een rij met de keys die overeenkomen met de waarde <code>val</code> (zoekfunctie)
<code>array_val(\$a)</code>	geef een rij met alle values van <code>\$a</code>
<code>print_r(\$a)</code>	print de rij <code>\$a</code>
<code>sort(\$a)</code>	sorteer de rij <code>\$a</code>
<code>shuffle(\$a)</code>	shud de rij <code>\$a</code>
<code>min(\$a)</code>	minimum van de rij <code>\$a</code>
<code>max(\$a)</code>	maximum van de rij <code>\$a</code>

7.7.4 Functies voor variabelen

<code>define</code>	definieer een constante
<code>unset(\$a)</code>	vernietig de variabele <code>\$a</code>
<code>isset(\$a)</code>	is de variabele <code>\$a</code> set?
<code>is_bool(\$a)</code>	is de variabele <code>\$a</code> een boolean?
<code>is_double(\$a)</code>	double?
<code>is_string(\$a)</code>	string?
...	
<code>get_type(\$a)</code>	geef het type van het variabele <code>\$a</code>
<code>var_dump(\$a)</code>	druk alle gegevens van de variabele <code>\$a</code> af
<code>get_defined_vars()</code>	geef alle gedefinieerde variabelen

Opdracht 10 *Zoek op in de documentatie hoe een constante gedefinieerd kan worden m.b.v. de functie `define`. Maak de constante `PI` (3.1415) en gebruik `PI` om de omtrek van een cirkel met straal 12 (`$straal=12;`) te berekenen ($2\pi r$).*

Opdracht 11

Bekijk

```
<?php
$a = get_defined_vars();
echo '<pre>';
var_dump($a);
echo '</pre>';
?>
```

Welke informatie kan je hiermee opzoeken?

7.7.5 Include en require

Via de functies `include` en `require` wordt een bestand geopend, de PHP-code in dit bestand wordt uitgevoerd en het resultaat wordt geplaatst op de positie waar de functie aangeroepen werd. Het verschil tussen `include` en `require` ligt in hoe fouten behandeld worden. Indien het te openen bestand niet bestaat geeft `include` een warning terwijl `require` een fatale error geeft. `include_once` en `require_once` zijn vergelijkbaar met `include` en `require`. Enkel wordt er nu voor gezorgd dat elk bestand maar één keer ingevoegd wordt.

Via deze functies kunnen programma's opgesplitst worden in meerdere bestanden. Vanuit `vb12.php` wordt het bestand `functies.php` geopend. In `functies.php` staan de definitie van de functies `omtrek` en `oppervlak`. Vanuit `functies.php` wordt `constantes.php` geopend. In dit bestand worden de constanten `PI` en `E` gedefinieerd.

vb12.php

```
1 <?php
2 include(__DIR__ . "/functies.php");
3 include (__DIR__ . "/constantes.php");
4 $o = omtrek(12);
5 echo $o;
6 ?>
```

functies.php

```
1 <?php
2 echo "functies.php<br/>\n";
3
4 function omtrek($straal){
5     return 2*PI*$straal;
6 }
7
8 function oppervlak($straal){
9     return PI*pow ($straal , 2);
10 }
11 ?>
```

constantes.php

```
1 <?php
2 echo "constantes.php<br/>\n";
3 define('PI', 3.1415);
4 define('E', 2.7282);
5 ?>
```

7.7.6 Header

Via de functie `header` kan de response header van de server gewijzigd worden. Belangrijk is dat het aanroepen van de functie gebeurt vóór output verstuurd wordt. De functie `header` wordt daarom meestal bovenaan in een PHP-bestand aangeroepen.

`Header` kan gebruikt worden om een **header redirect** te maken. Onderstaande regel zorgt ervoor dat de browser doorverwezen wordt naar een andere pagina. De opdracht `exit()` zorgt ervoor dat de rest van het huidige script niet uitgevoerd wordt.

```
1 <?php
2 header("Location: http://www.demorgen.be/");
3 exit();
4 ?>
```

Dit kan ook gebeuren via een **header refresh**. Er kan nu ook ingesteld worden na hoeveel seconden de redirectie moet gebeuren.

```
1 <?php
2 $sec = 5;
3 header("Refresh: $sec; url=http://www.demorgen.be");
4 ?>
```

Via de functie `header` kan ook het MIME-type van de verstuurd inhoud bepaald worden. In onderstaand voorbeeld wordt een bestand als PDF verstuurd.

```
1 <?php
2 $file = 'a.pdf';
3 header('Content-type: application/pdf');
4 header('Content-Disposition: attachment; filename="'. $file . '");
5 readfile($file);
6 ?>
```

Zie ook `vb4.php` uit sectie ??.

Opdracht 12 *Bekijk de informatie en de extra voorbeelden op <http://php.net/manual/en/function.header.php>*

7.7.7 Output buffers

Opdracht 13 *Zoek op hoe je een pagina kan comprimeren en kan doorsturen als gzip.*

- <http://www.php.net/manual/en/function.ob-start.php>

- *<http://www.php.net/manual/en/function.ob-gzhandler.php>*
- *<http://www.php.net/manual/en/function.ob-end-flush.php>*

Hoofdstuk 8

Formulieren, cookies en sessions

8.1 Formulieren

8.1.1 Voorbeeld 1

Een formulier (Eng. form) is een stuk HTML-code dat gebruikt wordt om invoer van een gebruiker te verwerken. In `vb1_get.html` wordt een formulier met enkele labels, tekstvelden, radiobuttons en een selectiebox aangemaakt. In figuur 8.1 wordt dit voorbeeld ook getoond zoals het geopend wordt in een browser. De HTML-code van het formulier kan opgesteld worden via het palette in netbeans, zie ook figuur ??.



Figuur 8.1: *vb1_get.html*

Wanneer in het formulier op de verzendknop gedruwd wordt dan wordt als actie het bestand `vb1verwerking.php` geopend:

```
<form action="vb1verwerking.php" method="get">
```

Het attribuut `method` met waarde `get` in het `form`-element zorgt er voor dat alle ingevoerde waarden geplaatst worden in de URL. Een mogelijke URL ziet er als volgt uit: `http://localhost/cursus/hfst12/vb1/vb1verwerking.php?vnaam=j&anaam=w&email=jw%40phl.be&gesl=m&lft=lftTss21en40`

Bij elke `input`-element in een formulier hoort normaal een naam (`name`) en een waarde (`value`).

Bij een tekstveld wordt in het formulier de naam van het tekstveld gespecificeerd:

```
<input type="text" name = "vnaam"/>
```

Er wordt in dit voorbeeld geen initiële waarde in het tekstveld ingegeven, indien dit wel nodig is dan kan dit via het attribuut value:

```
<input type="text" name = "vnaam" value = "init. waarde"/>
```

Bij een radiobutton wordt naam en waarde gespecificeerd:

```
<input type="radio" name="gesl" value="m"/> m
```

Een selectiebox wordt aangemaakt via de code

```
<select id="lft" name = "lft" >
  <option value = "lftTot20">jonger dan 20</option>
  <option value = "lftTss21en40">tussen 21 en 40</option>
  <option value = "lftTss41en60">tussen 41 en 60</option>
  <option value = "lftGroterDan60">ouder dan 60</option>
</select>
```

De selectiebox heeft een waarde voor het attribuut name. Binnen de selectiebox worden vier opties aangemaakt. Elke optie heeft een value.

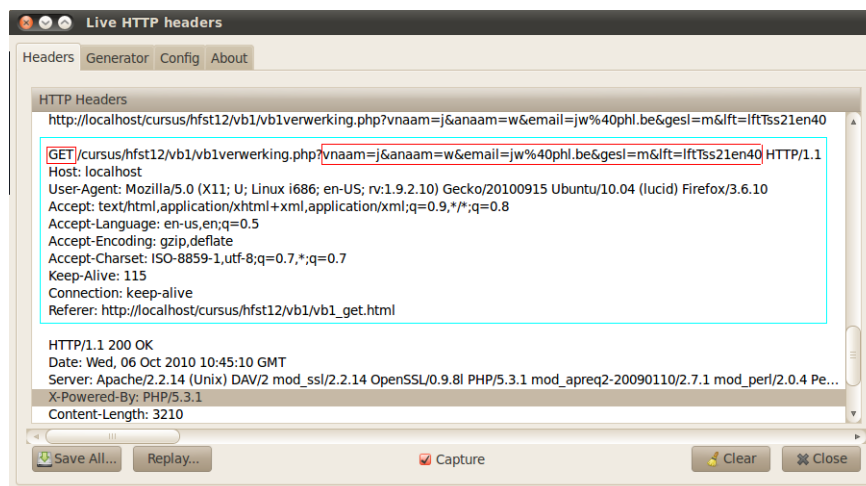
Voor een meer uitgebreid overzicht van formulieren wordt de lezer verwezen naar de link <http://www.w3.org/TR/REC-html40/interact/forms.html>.

`vb1_post.html` bevat grotendeels dezelfde code als `vb1_get.html`. Enkel heeft het element `form` een andere waarde voor het attribuut `method` gekregen.

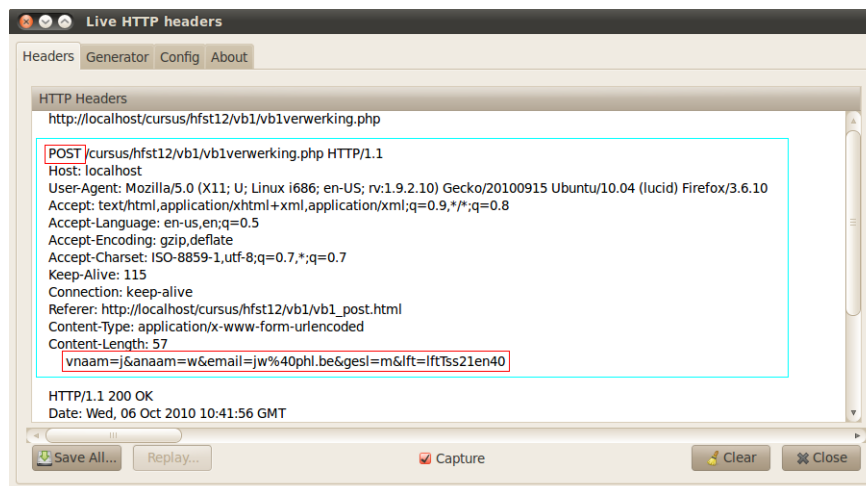
```
<form action="vb1verwerking.php" method="post">
```

Via `method="post"` wordt gezorgd dat de ingevoerde waarden niet zichtbaar zijn in de URL (maar wel meegegeven worden in de body van het http-request).

Het verschil tussen `get` en `post` wordt geïllustreerd in figuren 8.2 en 8.3. Bij methode `get` worden alle ingevoerde waarden in de URL geplaatst, bij `post` staat deze informatie in de request-header.



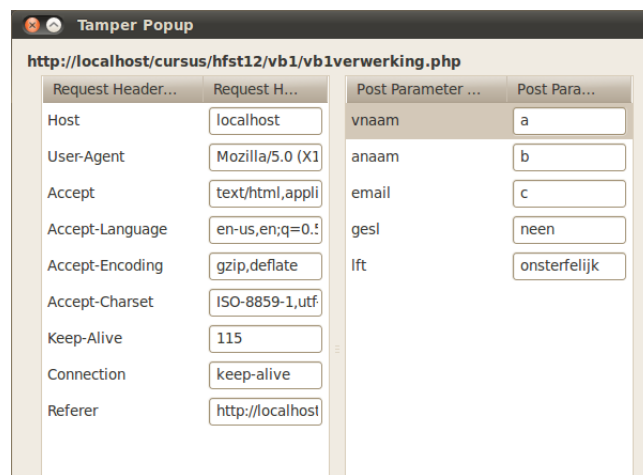
Figuur 8.2: Request-header bij het openen van `vb1verwerking.php` vanuit `vb1_get.html`.



Figuur 8.3: Request-header bij het openen van `vb1verwerking.php` vanuit `vb1_post.html`.

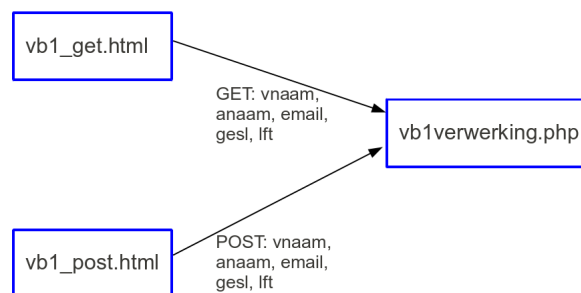
Via het attribuut `method="get"` in het form-element worden alle ingevoerde gegevens in de URL geplaatst. Deze gegevens kunnen dus eenvoudig aangepast worden in de browser. Via het attribuut `method="post"` worden alle ingevoerde gegevens in de request-header geplaatst. Dit biedt echter geen extra bescherming. Bijvoorbeeld via de Tamper Data-plugin¹ kunnen de waarden in de header gewijzigd worden. Zie ook figuur 8.4.

¹<https://addons.mozilla.org/en-US/firefox/addon/966/>



Figuur 8.4: Het gebruik van Tamper Data om de waarden gepost in `vb1_post.html` te wijzigen.

In figuur 8.5 wordt de grafische voorstelling van voorbeeld 1 gegeven.



Figuur 8.5: Grafische voorstelling van `vb1_get.html`, `vb1_post.html` en `vb1verwerking.php`.

8.1.2 De verwerking van voorbeeld 1

Via `vb1verwerking.php` wordt de formulieren in `vb1_get.html` en `vb1_post.html` verwerkt. Dit gebeurt dit aan de hand van de arrays `$_GET`, `$_POST`, `$_REQUEST` en `$_SERVER`.

```

                                vb1verwerking.php
1  <?php
2  echo '<h3>$_GET</h3>';
3  echo "&<table>\n";
4  foreach ($_GET as $k => $v){
5      echo "<tr><td>$k:</td><td>$v</td></tr>\n";
6      }
7
8  echo "</table>\n";
9  echo '<h3>$_POST</h3>';
10 echo "<table>\n";
11 foreach ($_POST as $k => $v){
12     echo "<tr><td>$k:</td><td>$v</td></tr>\n";
13     }
14
15 echo "</table>\n";
16 echo '<h3>$_REQUEST</h3>';
17 echo "<table>\n";
18 foreach ($_REQUEST as $k => $v){
19     echo "<tr><td>$k:</td><td>$v</td></tr>\n";
20     }
21
22 echo "</table>\n";
23 echo '<h3>$_SERVER</h3>';
24 echo "<table>\n";
25 foreach ($_SERVER as $k => $v){
26     echo "<tr><td>$k:</td><td>".
27         htmlentities($v, ENT_QUOTES, 'ISO-8859-1') .
28         "</td></tr>\n";
29     }
30 echo "</table>\n";
31 ?>

```

Opdracht 1 Voer `vb1_get.html` en `vb1_post.html` uit. Welke informatie staat in de rijen `$_GET`, `$_POST`, `$_REQUEST` en `$_SERVER` ?

Opdracht 2 Kopieer `vb1_post.html` en `vb1verwerking.php` naar de map `opdracht2`. Pas `vb1verwerking.php` aan zodanig dat de uitvoer eruit ziet zoals getoond in figuur 8.6. Vertrek van onderstaande code.

```

<?php
$vnaam = $_POST["vnaam"];
...
?>

```

```

<table>
  <tr>
    <td> Voornaam: </td>
    <td> <?php echo $vnaam;?></td>
  </tr>
  ...

```

```

Voornaam: Richard
Achternaam: Dawkins
Email:      rd@a.uk
Geslacht:   mannelijk
Leeftijd:   ouder dan 60

```

Figuur 8.6: *vb1*verwerking zoals beschreven in opdracht 2..

Opdracht 3 *Bekijk `vb1/java/Header_get.java` en `vb1/java/Header_post.java`.*

8.1.3 Voorbeeld 2

In figuur 8.7 wordt voorbeeld 2 grafisch weergegeven.



Figuur 8.7: *Grafische voorstelling van `vb2.html`, `vb2bevestig.php` en `vb2verwerking.php`*

In `vb2.html` kunnen de waarden voor voornaam (`vnaam`) en achternaam (`anaam`) ingegeven worden. De inhoud van `vnaam` en `anaam` worden geplaatst in hidden inputs in het eerste formulier van `vb2bevestig.html`. Vanuit `vb2.html` wordt `vb2bevestig.php` geopend. Via het eerste formulier wordt `vb2verwerking.php` geopend. De waarden in de hidden inputs zijn ook in `vb2verwerking.php` beschikbaar en worden hier afgedrukt. Via het tweede formulier in `vb2bevestig.php` wordt `vb2.html` geopend.

vb2.html

```

1 <form action="vb2bevestig.php" method="post">
2 <div class="tabel">
3     <div class="rij">
4         <div class="titel">Voornaam:</div>
5         <div class="invoer"><input type="text" name="vnaam" />
6     </div>
7     <div class="rij">
8         <div class="titel">Achternaam:</div>
9         <div class="invoer"><input type="text" name="anaam" />
10    </div>
11    </div>
12    <input type="submit" value="Verzend" />
13 </form>

```

vb2bevestig.php

```

1 <?php
2 $anaam = $_POST[ 'anaam' ];
3 $vnaam = $_POST[ 'vnaam' ];
4 echo "<br/>voornaam = '$vnaam' en achternaam = '$anaam'.";
5 ?>
6
7 <form action="vb2verwerking.php" method = "post">
8     <input type="submit" value="Bevestig"/>
9     <input type = "hidden" name = "anaam"
10         value = "<?php echo $anaam; ?>" />
11     <input type = "hidden" name = "vnaam"
12         value = "<?php echo $vnaam; ?>" />
13 </form>
14 <br />
15 <form action="vb2.html" method = "post">
16     <input type="submit" value="Terug" />
17 </form>

```

vb2verwerking.php

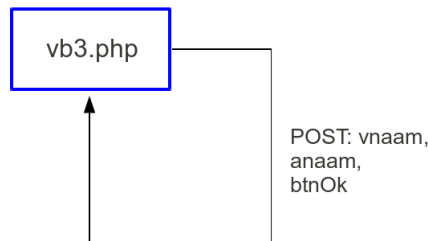
```

1 <?php
2 $anaam = $_POST[ 'anaam' ];
3 $vnaam = $_POST[ 'vnaam' ];
4 echo "Ingevoerd: voornaam $vnaam en achternaam $anaam.";
5 ?>

```

8.1.4 Voorbeeld 3

In figuur 8.8 wordt voorbeeld 3 grafisch weergegeven. In `vb3.php` wordt eerst gecontroleerd of op `btnOk` gedruwd werd en of `vnaam` en `anaam` ingevuld zijn. Indien dit zo is dan worden `vnaam` en `anaam` afgedrukt anders wordt het formulier getoond waar `vnaam` en `anaam` ingevuld kunnen worden.



Figuur 8.8: Grafische voorstelling van `vb3.php`

```

vb3.php
1 <?php
2 if (isset($_POST['btnOk'])) {
3     $anaam = $_POST['anaam'];
4     $vnaam = $_POST['vnaam'];
5     echo "Ingevoerd: voornaam '$vnaam' en achternaam '$anaam'.<br/>";
6 }
7 else {
8     ?>
9     <form action="vb3.php" method="post">
10    <div class="tabel">
11        <div class="rij">
12            <div class="titel">Voornaam:</div>
13            <div class="invoer"><input type="text" name="vnaam" />
14        </div>
15    </div>
16    <div class="rij">
17        <div class="titel">Achternaam:</div>
18        <div class="invoer"><input type="text" name="anaam" />
19    </div>
20 </div>
21 <input type="submit" name="btnOk" value="Ok" />
22 </form>
23 <?php } ?>
  
```

Opdracht 4 *Bekijk vb4.*

8.1.5 Htmlelements

Opdracht 5 Geef de tekst

```
<script>alert("hallo");</script>
```

in in het tekstveld in vb5/vb5a/vb5.php en vb5/vb5b/vb5.php.

In vb5/vb5a/vb5.php wordt het javascript letterlijk in de HTML-code geplaatst en uitgevoerd in de browser.

In vb5/vb5b/vb5.php wordt de invoer vervangen door

```
&lt;script&gt;alert(&quot;hallo&quot;);&lt;/script&gt;
```

8.2 Cookies

Een cookie is een klein bestand dat door de server op de computer van een gebruiker (client) geplaatst wordt. Deze bestanden bevatten tekstinformatie, die gebruikt kan worden door de server.

Een typisch voorbeeld van het gebruik van cookies is het bewaren van keuzes bij het bezoeken van een website. Een gebruiker kiest bijvoorbeeld de eerste keer dat hij een site bezoekt Nederlands als taal. Deze waarde wordt vervolgens bewaard in een cookie en de gebruiker wordt doorverwezen naar de Nederlandstalige pagina's. Bij een volgend bezoek wordt dit cookie gelezen en wordt de gebruiker onmiddellijk verwezen naar de Nederlandstalige pagina's.

Cookies worden aangemaakt via de functie `setCookie`. Deze functie wordt meestal met twee of drie argumenten gebruikt. Bijvoorbeeld de code

```
setcookie ('taal', 'NL');
```

maakt een cookie aan met naam `taal` en inhoud `'NL'`, het cookie blijft geldig tot het eind van deze sessie (tot de browser afgesloten wordt).

```
setcookie ('taal', 'NL',time() + 24 * 60 *60 );
```

Doet hetzelfde enkel blijft het cookie nu bestaan tot 1 dag (= 24 *60*60 seconden) na de huidige tijd.

Via de functie `setcookie` kan een bestaande cookie ook gewijzigd worden en kan een cookie verwijderd worden door het derde argument een waarde voor de tijd te geven die al in het verleden ligt:

```
setcookie ('taal', '',time() - 24 * 60 *60 );
```

De waarden van een cookie kunnen gelezen worden via de rij `$_COOKIE`. De sleutels van deze rij zijn de namen van alle beschikbare cookies. Cookies zijn ook beschikbaar via de rij `$_REQUEST`.

Via `vb6_setcookie.php` wordt een cookie met naam `'taal'` en waarde `'NL'` geplaatst.

vb6_setcookie.php

```

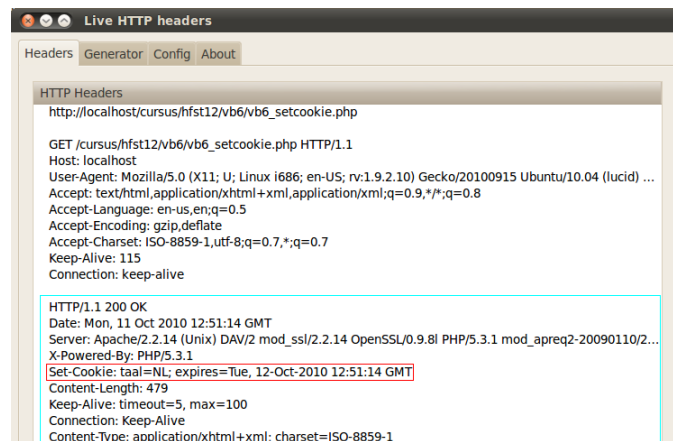
1 <?php    setcookie ( 'taal', 'NL',time() + 24 * 60 *60 );
2 ...

```

Zoals getoond in figuur 8.9 wordt aan de hand van de functie setcookie de response-header gewijzigd. De regel

Set-Cookie: taal=NL; expires=Tue, 12-Oct-2010 12:51:14 GMT

wordt in de header geplaatst. Het wijzigen van de header moet gebeuren voor er informatie naar de client gestuurd wordt. De functie setcookie moet dus steeds helemaal bovenaan in een PHP-bestand aangeroepen worden.



Figuur 8.9: Headers verstuurd bij het openen van vb6_setcookie.php.

Via vb6_tooncookie.php worden alle cookies getoond (die de server localhost bij de client geplaatst heeft).

vb6_tooncookie.php

```

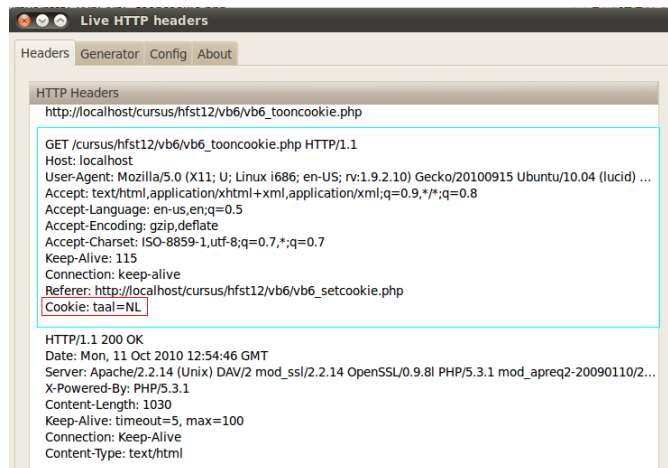
1 <p>
2 <h3>$_COOKIE</h3>
3 <pre>
4 <?php var_dump($_COOKIE); ?>
5 </pre>
6 </p>
7 <p>

```

In figuur 8.10 wordt de communicatie tussen de client en de server opnieuw getoond aan de hand van de header. In de request-header gestuurd door de client staat de informatie over alle cookies van de client.

8.3 Sessions

Ook via sessies kan informatie van de gebruiker bewaard worden. Deze informatie wordt nu niet bij de client geplaatst maar wel bewaard op de server.



Figuur 8.10: *Headers* verstuurd bij het openen van *vb6_tooncookie.php*.

Een sessie wordt gestart via het commando

```
session_start();
```

De client ontvangt vervolgens een cookie met naam PHPSESSID en als inhoud een session-identificatie. Er wordt een bestand aangemaakt met dezelfde naam als de session-identificatie. Dit bestand komt te staan in de directory `/opt/lampp/tmp` of `c:\xampp\tmp`.

Een waarde kan bewaard worden in een sessie via

```
$_SESSION["id"] = "waarde";
```

Verder kan een sessie ook vernietigd worden via

```
session_destroy()
```

Via dit commando wordt het bestand verwijderd van de server.

8.4 Oefeningen

Oefening 1 Gegeven het volgende formulier:

```

1 <form action="Oefening.php" method="get">
2   <div class="tabel">
3     <div class="rij">
4       <div class="titel">Dobbelsteen 1</div>
5       <div class="invoer">
6         <select name="dob1">
7           <option value="d2">d2</option>
8           <option value="d4">d4</option>
9           <option value="d6">d6</option>
10        </select> </div>
11      </div>
12      <div class="rij">
13        <div class="titel">Dobbelsteen 2</div>
14        <div class="invoer">
15          <select name="dob2">
16            <option value="d2">d2</option>
17            <option value="d4">d4</option>
18            <option value="d6">d6</option>
19          </select> </div>
20        </div>
21        <div class="rij">
22          <div class="titel">Dobbelsteen 3</div>
23          <div class="invoer">
24            <select name="dob3">
25              <option value="d2">d2</option>
26              <option value="d4">d4</option>
27              <option value="d6">d6</option>
28            </select> </div>
29          </div>
30          <div class="rij">
31            <div class="titel">Inzet:/td>
32            <div class="invoer">
33              <input type="text" name="inzet"/>
34            </div>
35          </div>
36        <div class="rij">
37          <div class="titel">&nbsp;</div>
38          <div class="invoer">
39            <input type="submit" value="verzend"
40              name="verzendknop"/>
41          </div>
42        </div>
43 </div>
</form>

```

Dit formulier wordt gebruikt in het bestand *oefening1.php*. Invoer en verwerking gebeuren via dit bestand.

Het formulier wordt getoond wanneer de gebruiker op de site komt (of liever als de gebruiker nog niet op de knop met opschrift verzend gedruwd heeft).

De gebruiker kan 3 dobbelstenen selecteren met de selecties `selectDob1`, `selectDob2` en `selectDob3`. Er kunnen drie soorten dobbelstenen gekozen worden: een d2 met twee zijden (en mogelijke worp 1 of 2), een d4 met vier zijden (en mogelijke worp 1, 2, 3 of 4) of een klassieke d6 met zes zijden (en mogelijke worp 1, 2, 3, 4, 5 of 6). De gebruiker kan een inzet ingeven via een text-input.

Als op de knop `Verzend` gedrukt wordt dan wordt eerst gecontroleerd of alle waarden correct ingevoerd zijn. Indien dit niet zo is dan wordt de boodschap "Verkeerde ingave" getoond. Indien de waarden wel voldoen dan worden drie willekeurige getallen gegenereerd voor de worpen van de drie dobbelstenen. Als deze drie getallen gelijk zijn, dan krijgt de gebruiker zijn inzet terug vermenigvuldigd met het totaal aantal zijden van de dobbelstenen. Bijvoorbeeld bij een inzet van 12.55 euro, gekozen dobbelstenen d2, d4 en d6 en worp 2, 2, 2 bedraagt de winst $12.55 * (2 + 4 + 6) = 150.6$ euro.

Bij een juiste ingave wordt het resultaat getoond in een tabel:

Inzet:	12.55
Gooi 1:	2
Gooi 2:	2
Gooi 3:	4
Resultaat:	0

Oefening 2 Maak de toepassing die bestaat uit `invoerOef2.html` en `verwerkingOef2.php`.

In figuur 8.11 wordt `invoerOef2.html` getoond. Via deze pagina kunnen drie komma-getallen ingevoerd worden. Als op de knop met opschrift `verzend` gedrukt wordt dan wordt een berekening uitgevoerd. Via een eerste selectiebox wordt bepaald of de eerste term in de berekening de hoogste of de laagste waarde van de drie ingevoerde getallen is. Via een tweede selectiebox wordt een operatie gekozen. De mogelijke operaties zijn '+' en '-'. Via een derde listbox tenslotte wordt bepaald of de tweede term in de berekening de hoogste of de laagste van de drie ingevoerde getallen is.

Gebruik voor de verwerking van de acties horende bij de knop het script `verwerkingOef2.php`. Er wordt eerst gekeken of alle waarden correct ingevoerd zijn. Indien dit zo is dan wordt de berekening uitgevoerd en het resultaat wordt samen met de ingevoerde waarden afgedrukt. Bijvoorbeeld:

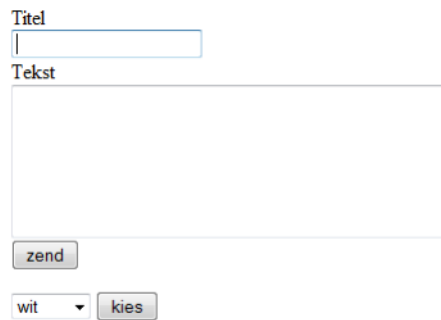
`(12,1,3,hoogste,laagste,min) = 11`

Figuur 8.11: `invoerOef2.html`

Oefening 3 Deze toepassing bestaat uit de bestanden *oefening3.php* en *oefening3Verwerking.php*.

Via *oefening3.php* kan een titel en een tekst ingegeven worden. Zie figuur 8.12. Via de knop *zend* wordt *oefening3Verwerking.php* geopend. Gebruik in deze pagina de functie *strip_tags* om de ingevoerde waarden weer te geven.

In *oefening3.php* kan ook een kleur gekozen worden uit de waarden rood, wit, groen. Indien op de knop *kies* gedrukt wordt dan wordt *oefening3.php* geopend met de gekozen kleur als achtergrond. De gekozen kleur wordt ook bewaard in een cookie.



The screenshot shows a web form with the following elements:

- A label "Titel" above a single-line text input field.
- A label "Tekst" above a multi-line text area.
- A button labeled "zend" below the text area.
- A dropdown menu with "wit" selected, followed by a button labeled "kies".

Figuur 8.12: *oefening3.php*