Einstieg: Wiederholung des Variablenbegriffes

Wir haben in PHP Variablen kennen gelernt.

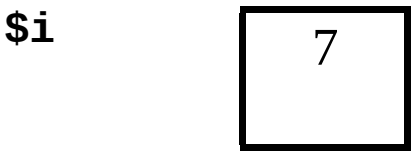
```
$i=5;
$i=7;
echo $i; // ergebnis: 7
```

Dabei haben wir uns eine Variable vorgestellt als Schachtel (Kiste), die einen Namen hat, hier $i.

In diesr Schachtel befindet sich ein Zettel, auf dem ein Wert oder eine Zeichenfolge stehen kann, z.B. 5.

Dann kann die Variable als Platzhalter verwendet werden.

Wird ein neuer Wert der Variable zugewiesen, so ist dies gleichsam einem Herausnehmen des alten Zettels und Hineinlegen eines neuen Zettels mit dem Wert 7.



Datenfeld (Array)

Ein Datenfeld ist eine Erweiterung des Variablenbegriffes. Es lassen sich nun viele Zettel in eine Schachtel legen.

$j

7	5	3	17

Um jeden Inhalt eindeutig ansprechen zu können, benötigt man bei einem Datenfeld stets auch einen Index, der in eckigen Klammen angegeben wird. Der Index beginnt (das ist in der Informatik durchaus üblich bei 0).

```
echo $j[0];         // Ergebnis: 7
echo $j[2];         // Ergebnis: 3
```

Das Erzeugen eines Arrays ist genauso einfach:

```
$zahl[0]=7;
$zahl[1]=5;
$zahl[2]=3;
```

Um an das Ende eines Datenfeldes eine neue Zahl anzufügen, muss man den Index nicht dazuschreiben:

```
$zahl[]=17;

echo $zahl[3];      // Ergebnis: 17;
```

Hinweis: Ein Array kann nicht nur Zahlen beinhalten! Die Länge eines Arrays muss in PHP nicht beim Erstellen angegeben werden. Er kann im Laufe des Programms beliebig erweitert werden.

Arrays innerhalb von Schleifen

Der Vorteil von Arrays liegt darin, dass er in Schleifen eingesetzt werden kann, um mit einem Befehl alle Variablen des Arrays zu bearbeiten.

Beispiel:

```
<?php
$zahl[0]=7;
$zahl[1]=5;
$zahl[2]=3;
$zahl[3]=17;

for ($i=0;$i<4;$i++) {

echo $zahl[$i],", ";

}

// Ergebnis: 7, 5, 3, 17
?>
```

Es werden also mit einem Befehlsaufruf alle Variablen ausgegeben. Dies ist für uns im Moment der wichtigste Vorteil dieses neuen Konstruktes!

Die Count-Funktion

Programmiersprachen zeichnen sich dadurach aus, dass mehr oder weniger viele Funktionen existieren, die der Programmierer verwenden kann. In PHP existiert eine Funktion mit der man die Länge eines Arrays ermitteln kann, die Funktion **count()**.

```
echo count($zahl);  //Ergebnis: 4
```

Damit lässt sich das Beispiel von oben so umgestalten, dass immer das gesamte Feld ausgegeben wird!

```
<?php
$zahl[0]=7;
$zahl[1]=5;
$zahl[2]=3;
$zahl[3]=17;

for ($i=0;$i<count($zahl);$i++)
{
  echo $zahl[$i],", ";
}

// Ergebnis: 7, 5, 3, 17
?>
```

Das Sortieren von Datenfeldern

Das Sortieren, z.B. von Datenfeldern, ist ein wichtiges Problem der Informatik. Es wurden im Laufe der Zeit viele verschiedene Methoden entwickeln. Auch wir wollen das Problem lösen und beginnen damit zu beschreiben, wie der Mensch Kärtchen sortiert, auf denen Zahlen stehen, z.B. unser Feld:

$j

17	5	3	7

Beschreibung einer Methode:
1.Durchsuche das ganze Feld und suche die kleinste Zahl.
2.Vertausche diese Zahl mit dem ersten Eintrag.
  Damit ist die erste Stelle des Feldes richtig sortiert.
3.Wiederhole den Vorgang ab der Stelle zwei usw. bis zum
  letzten Feld!

In jedem Durchgang wird eine weitere Stelle sortiert:

Durchgang	Feld
0	17, 5, 3, 7
1	3, 5, 17, 7
2	3, 5, 17, 7
3	3, 5, 7, 17

Dieser Sortieralgorithmus soll nun in PHP implementiert werden.

Wir gehen Schritt für Schritt vor:

Zuerst benötigen wir einen Algorithmus, der uns die kleinste Zahl eines Feldes liefert.:

```
function GetMin($zahl){
// wir tun so, als ob die erste Zahl die kleinste ist
$Zwischenergebnis=$zahl[0];

// jetzt prüfen wir jede Zahl im Feld ob sie nicht kleiner ist
// wir beginnen mit $zahl[1], da $zahl[0] schon beachtet wurde
for ($i=1; $i<count($zahl); $i++)
{

// Wenn wir eine kleinere Zahl finden, dann ist das unsere
// vorläufiges Zwischenergebnis
   if ($zahl[$i]<$Zwischenergebnis)
               $Zwischenergebnis=$zahl[$i]

}
// Am Ende ist $Zwischenergebnis die kleinste Zahl!
return $Zwischenergebnis;
}

echo GetMin($zahl); // Ergebnis: 3
```

Hinweis: In der Informatik lassen sich viele Ergebnisse so erzielen, indem man davon ausgeht, dass das erste Element das richtige Ergebnis ist, dann prüft man alle anderen Werte eines Feldes und korrigiert ggf. das Ergebnis. Am Ende des Durchlaufs durch das Datenfeld muss dann zwangsläufig das richtige Ergebnis herauskommen.

Mit dieser Funktion lässt sich also das kleiste Element eines Datenfeldes herausfinden. Zum Vertauschen von zwei Feldern benötigt man aber nicht den Wert der Zahl, sondern seine Position.

Aufgabe: Verändere die Funktion GetMin($zahl) in eine Funktion GetMinPos($zahl), die die Position liefert, an der die kleinste Zahl steht!

```
function GetMinPos($zahl){
// wir tun so, als ob die erste Zahl die kleinste ist
$Zwischenergebnis=$zahl[0]; $Position=0;

// jetzt prüfen wir jede Zahl im Feld ob sie nicht kleiner ist
// wir beginnen mit $zahl[1], da $zahl[0] schon beachtet wurde
for ($i=1; $i<count($zahl); $i++)
{

// Wenn wir eine kleinere Zahl finden, dann ist das unsere
// vorläufiges Zwischenergebnis
   if ($zahl[$i]<$Zwischenergebnis) {
               $Zwischenergebnis=$zahl[$i]
               $Position=$i;
   }
}
// Am Ende ist $Zwischenergebnis die kleinste Zahl!
return $Position;
}
```

Ich will ja jetzt die Position haben, ich muss neben der Zahl also nur die Position mitführen. Für die If-Anweisung benötige ich nun die geschweiften Klammern, da mehr als eine Anweisung ausgeführt wird, falls die Bedingung zutrifft.

Am Ende wird die Position mit return zurückgeliefert.

Bsp:

```
echo GetMinPos($zahl);   //Ergebnis: 2
```

Der kleinste Wert steht an der Position 2 (die Zählung beginnt bei 0).

Für das **Vertauschen** von zwei Variablen in einem Feld benötigt man eine Variable, wo ich mir den Wert merken kann, den ich zuerst überschreibe. Ich nenne diese Variable $Schublade.

Ich möchte zwei Werte vertauschen, nämlich die Postion 0 mit der Position 2. Das geht so:

```
$Schublade=$zahl[0]; // Zahl merken
$zahl[0]=$zahl[2];
$zahl[2]=$Schublade;
```

Jetzt sind die beiden Felder vertauscht.

Jetzt sind wir schon fast fertig. Beim nächsten Durchgang dürfen wir aber nicht das gesamte Feld nach dem kleinsten Wert durchsuchen, sondern wir beginnen bei der zweiten Variablen, dann bei der dritten usw.

Aufgabe: Schreibe eine Funktion GetMinPosAb($zahl,$ab), die die Position des kleinsten Wertes eines Datenfeldes liefert, ab einer bestimmten Position $ab an!

```
function GetMinPosAb($zahl, $ab){
// wir tun so, als ob die $zahl[$ab] die kleinste ist
$Zwischenergebnis=$zahl[$ab]; $Position=$ab;

// jetzt prüfen wir jede Zahl im Feld oberhalb von $ab, ob sie
// nicht kleiner ist;
// wir beginnen mit $zahl[$ab+1]
for ($i=$ab+1; $i<count($zahl); $i++)
{

// Wenn wir eine kleinere Zahl finden, dann ist das unsere
// vorläufiges Zwischenergebnis
   if ($zahl[$i]<$Zwischenergebnis) {
               $Zwischenergebnis=$zahl[$i]
               $Position=$i;
   }
}
// Am Ende ist $Zwischenergebnis die kleinste Zahl!
return $Position;
}
```

Jetzt können wir den vollständigen Sortieralgorithmus aufschreiben, auch wieder mit einer Funktion MinSort();

```
function MinSort($zahl){

// Das Sortieren bedarf der wiederholten Suche des kleinsten
// Wertes und des Austauschen dieses Wertes. Insgesamt führen
// wir diese Funktion so oft durch, wie viele Einträge das
// Datenfeld hat. Der kleinste Wert wird jeweils mit dem Wert
// an Stelle $z vertauscht. Wir fangen mit dem ganzen Feld an
// also an Stelle 0

for ($z=0; $z<count($zahl), $z++)
{
// Am Anfang die ersten, dann jeweils die nächste Position
// wird in die Schublade gepackt;
$Schublade=$zahl[$z];

// Suche der Position des kleinsten Wertes in einem Restfeld
// beim ersten mal ab Stelle 0 (alle), dann immer eines
// kleiner
$MinPos=GetMinPosAb($zahl,$z);

// Kleinster Wert wird nach vorne geschoben an den Beginn des
// Restfeldes
$zahl[$z]=$zahl[$MinPos];

// Die vordere Zahl wird nach hinten kopiert an die Stelle,
// des kleinsten Feldes
$zahl[$MinPos]=$Schublade;
}

}
```

Schaut doch gar nicht mal so schlecht aus! Diesen Algorithmus zum Sortieren eines Datenfeldes nennt man **"Sortieren durch Auswählen"**.