Zusammenfassung: Jahr 1

Inhaltsverzeichnis

1 Lernfeld 6 - Programmieren

1.1 tl;dr - Zusammenfassung der Zusammenfassung

1.2 Einführung in HTML und PHP

HTML ist, wie der Name – HyperText Markup Language – schon sagt, eine Beschreibungsssprache. Der aktuelle Standard lautet HTML 5 und basiert auf XHTML. Im Gegensatz zu Word sieht das interpretierte Ergebnis eines HTML-Dokuments nicht so aus, wie sie geschrieben wurde. HTML ist also eher mit LaTeXzu vergleich. Das heißt, HTML ist keine Programmiersprache.

In den folgenden Abschnitten werden zunächst die wichtigsten Tags beschrieben, um HTML-Dokumente zu strukturieren. Anschließend werden die Möglichkeiten von HTML anhand von Aufgaben und deren Lösungen dargestellt.

1.2.1 HTML5: Kurzreferenz

In diesem Abschnitt werden die wichtigsten Tags beschrieben, um ein HTML-Dokument zu strukturieren. Alle erwähnten Dateien befinden sich unter `code/lf06prog-code`.

Überschriften. Ähnlich wie in TeX können Überschriften in mehreren Ebenen beschrieben werden. Wo TeX bloß drei Ebenen vorsieht (`\section, \subsection} und \subsubsection`), sind durch HTML prinzipiell keine Grenzen gesetzt. Überschriften werden in HTML durch die Tags `<hX>` und `</hX>` beschrieben. Dabei steht X für die jeweilige Hierarchie. Abhängig von X wird die Größe der Überschrift gesetzt. Die Datei `lf06prog-headlines.html` macht das Beschriebene anschaulich.

Absätze. Möchte man einen Textblock als zusammengehörigen Absatz definieren, setzt man dafür die Tags `<p>` und `</p>`. Harte Zeilenumbrüche werden durch das einzelne Tag `<br />` beschrieben. Weil zwischen den Tags `<br> <br/>` eh nichts steht, wurde dieses vereinfacht. Daher ist auf das Leerzeichen in `<br />` zu achten.

Hervorhebungen.

`<b> </b>`	**fett** (physikalisch ≙ Stilelement)
`<strong> </strong>`	**fett** (logisch ≙ wichtig)
`<i> </i>`	*kursiv*
`<em> </em>`	betont; meist *kursiv*
`<sup> </sup>`	^{hochgestellt}
`<sub> </sub>`	$_{tiefgestellt}$
`<u> </u>`	~~durchgestrichen~~

Listen. Mit den Tags `<ul> </ul>` und `<ol> </ol>` lassen sich Listen definieren. `<li> </li>` definieren jeweils die Listenelemente, wobei den Listenelementen bei `<ul>` ein Punkt vorangestellt wird und bei `<ol>` die Listenelemente durchnummeriert werden. Siehe dazu auch `lf06prog-listen.html` und http://wiki.selfhtml.org/wiki/HTML/Textstrukturierung/Listen.

Umlaute. HTML kann Umlaute nicht ohne weiteres darstellen. Daher müssen Umlaute durch die folgenden Befehle definiert werden:

ä	&auml;
Ä	&Auml;
ö	&ouml;
Ö	&Ouml;
ü	&uuml;
Ü	&Uuml;
ß	&szlig;
€	&euro;

Tabellen. Besondere Bedeutung in HTML haben Tabellen. Mit diesen lässt sich der Aufbau einer Seite gestalten. Statt den Aufbau von Tabellen umständlich zu beschreiben, verweise ich auf die Datei `lf06prog-listen.html`. Diese sollte sich sowohl als Quellcode als auch im Browser angesehen werden.

Links. . . sind einer der großen Fortschritte, die das Internet erst zu dem machen, was es ist.

Interner Link	`<a href="interner-link.html">Beschreibung</a>`
Externer Link	`<a href=\"https://externer-link.html\">Beschreibung</a>`
Link auf Bild	`<a href=\"lokales-bild.jpg\">Bild</a>`
Link auf PDF	`<a href=\"lokales-pdf.pdf\">Beschreibung</a>`
Mail als Link	`<a href=\"mailto:email@email.email\">Schreib mich an!</a>`
Sprungmarke	`<a href="\#anker">Spring zu Anker</a>`
	benötigt: `<a name=\"anker\">Anker</a>`

Grafiken.

```
<img src="bild.jpg" width="" height="" border="0" alt="" title="" />
```

src	Wo die Datei liegt
width	Wie breit das Bild dargestellt werden soll
height	Wie hoch das Bild dargestellt werden soll
border	Rahmen?
alt	Beschreibung des Bildes
title	Titel des Bildes

Kommentare. . . lassen sich in HTML-Dokumenten einfügen, indem sie zwischen `<!-- -->` geschrieben werden.

1.3 PHP

PHP ist eine Skriptsprache und basiert auf der C-Syntax.

1.3.1 Quotation unter PHP

PHP unterscheidet wie die Bash zwischen schwacher und starker Quotation. Schwache Quotation bedeutet, dass Funktionen innerhalb der Anführungszeichen ïnterpretiert werden. Das bedeutet, dass Variablennamen aufgelöst werden und statt ihrer der entsprechende Werte ausgegeben wird. Starke Quotation lässt sich mit einfachen Anführungszeichen úmsetzen. Innerhalb der einfachen Anführungszeichen werden die Funktionen nicht interepretiert, sodass statt des Wertes der Variablenname ausgegeben wird. Im Listing [Nr] befindet sich ein Beispiel für schwache und starke Quotation in PHP.

```
<?php
 $ein_array = array();
 $ein_array[0] = $_POST["txt_eingabe_1"];
 $ein_array[1] = $_POST["txt_eingabe_2"];

 $noch_ein_array = array();
 $noch_ein_array[] = $_POST["txt_eingabe_1"];
 $noch_ein_array[] = $_POST["txt_eingabe_2"];

 $alternatives_array = array("$_POST["txt_eingabe_1"]", \
                             "$_POST["txt_eingabe_2"]");

 echo '$ein_array[0] gibt den Wert '.$ein_array[0].' aus.<br />';
 echo '$noch_ein_array[0] gibt ebenfalls den Wert '.$noch_ein_array[0]. \
      ' aus.<br />';
```

```
?>
```

Listing 1: Indexadressierte Arrays in PHP

Möchte man sowohl den Variablennamen als auch den Wert ausgeben, lässt sich die Quotation auch unterbrechen. Dies ist im dritten `echo`-Befehl des Listings zu sehen. Die Punkte vor und nach dem Variablennamen bedeuten, dass der Inhalt der Variable der Ausgabe angehängt wird.

1.4 Formulardaten

1.4.1 Formulardaten: HTML

Im Listing [Nr] steht `action=" script.php"`. Dies bezieht sich auf den Namen des Skriptes, an welches die Formulardaten beim Drücken auf den Submit-Button weitergereicht werden sollen.

```
<html>
 <body>
  <form action = "script.php" method = "POST">
   <input type = "text" name = "txt_variablenname"/><br />
   <input type = "radio" name = "rb_variablenname" value = "value"/><br />
   <input type = "checkbox" name = "cb_variablenname"/> <br />
   <input type = "submit" name = "btn_ok" value = "beschriftung"/>
  </form>
 </body>
</html>
```

Listing 2: Ein Beispiel für den HTML-Teil von Formulardaten

Als `method` lässt sich `POST` oder `GET` wählen. Der Unterschied zwischen `POST` und `GET` besteht darin, dass die Formulardaten von `GET` in der URL auftauchen. Dadurch werden die geforderten Daten sichtbar und durch brute force angreifbar. Ein weiterer Nachteil von `GET` besteht darin, dass die übergebenen Daten nicht größer als 2KB sein dürfen. Diese Nachteile und Beschränkungen gelten nicht für `POST`. Daher ist `POST` zu bevorzugen.

Der wichtigste Teil, um Formulardaten abzufragen, sind die Felder, in denen sie eingetragen werden können. Die Felder lassen sich wie im Listing gezeigt mit `<input .../>` einfügen. Beispielsweise kann mit dem type `text` ein String in der Variable `txt_variablenname` an das oben genannte Skript übergeben werden.

- `text`: der type `text` nimmt noch weitere Attribute, wie bspw. `size` und `maxlength`. Ersteres gibt die sichtbare Länge des Feldes an und letzteres die maximale Länge des Inputs.
- `radio`: auch hier gilt, dass mit dem `name` der Variablenname definiert wird. Darüber hinaus beschreibt `value` den Wert, den die Variable annimmt, wenn der Radiobutton angeklickt wird.
- `submit`: im Gegensatz zum type `radio` beschreibt `value` hier die Beschriftung des Submit-Buttons.

1.4.2 Formulardaten: PHP

Das nächste Listing zeigt, wie die Formulardaten in PHP verarbeitet werden können. Zu Anfang des Listings findet sich der Bereich „Variablen setzen“. Diese Aufteilung ist nicht notwendig, sondern vereinfacht nur die Les- und Wartbarkeit des Codes. Sollten sich die Variablennamen später einmal ändern, muss nur die eine Zuweisung angepasst werden und nicht jedes Vorkommen der Variable.

```
<?php
 #Variablen setzen
 $textvar = $_POST["txt_variablenname"];
 $rbvar = $_POST["rb_variablenname"];
 $cbvar = $_POST["cb_variablenname"];
```

```
#Zuweisung einer GET-Variablen:
#$get_var = $_GET["variablenname"];

echo 'Inhalt der Variable $textvar: '.$textvar.'.';
?>
```

Listing 3: Ein Beispiel für den PHP-Teil von Formulardaten

1.5 Strukturierte Programmierung

Strukturierte ist ein programmiersprachenübergreifendes Programmierparadigma. Es beinhaltet zum einen die baumartige Zerlegung eines Programms in Teilprogramme und enthält somit das Paradigma der prozeduralen Programmierung. Zum anderen verlangt die strukturierte Programmierung auf der untersten Ebene die Beschränkung auf drei Kontrollstrukturen: (1) Sequenzen, (2) Verzweigung und (3) Schleifen.

1.6 Programmablaufplan und Struktogramm

Programmablaufplan (PAP)

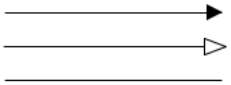
Ein Programmablaufplan (PAP) (auch Flussdiagramm, engl. *flowchart*) ist eine graphische Darstellung von Algorithmen in einem Programmen und beschreibt die Operationen zur Lösung einer Aufgabe. Die Symbole für Programmablaufpläne sind in der DIN 66001 genormt. Dort werden auch Symbole für Datenflusspläne definiert. Im Bereich der Softwareerstellung werden sie nur noch selten verwendet: Programmcode moderner Programmiersprachen bietet ähnlichen Abstraktionsgrad, ist jedoch einfacher zu erstellen und in der Regel sehr viel einfacher zu verändern als ein Ablaufdiagramm.

Das Konzept der Programmablaufpläne stammt, ebenso wie das etwas jüngere Nassi-Shneiderman-Diagramm (Struktogramm), aus der Zeit des imperativen Programmierparadigmas.

Elemente:

- Oval: Start/Stopp

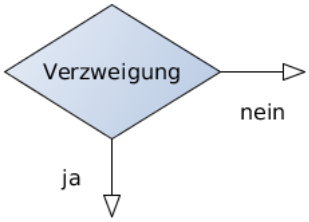
- Pfeil/Linie: Verbindung zum nächstfolgenden Element



- Rechteckt: Operationen

- Rechteck mit doppelten, vertikalen Linien: Unterprogrammaufruf

- Raute: Verzweigung



- Parallelogramm: Ein- und Ausgabe (nicht nach DIN 66001 1983!)

- Ergänzung: Schleife

Struktogramm

Nassi-Shneiderman-Diagramme, auch Strukogramme genannt,

1.7 Verzweigungen und Schleifen

In den folgenden Abschnitten werden wir uns mit IF-Verzweigungen sowie FOR- und WHILE-Schleifen in PHP beschäftigen.

1.7.1 Vergleichsoperatoren, Verknüpfungen und Überprüfung von Eingabewerten

PHP unterstützt ein Reihe von Vergleichsoperatoren. = ist kein Vergleichsoperator, sondern eine Zuweisung.

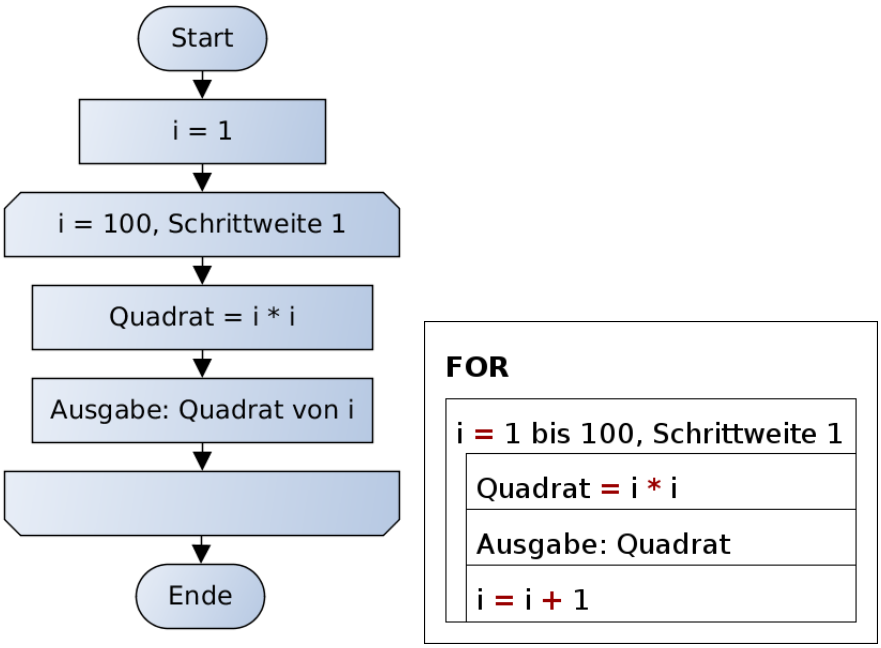
Operator	**Bedeutung**	**Beispiel**
==	Vergleich auf Gleichheit	$zahl == 5
===	Vergleich auf Typengleichheit	$zahl === 5
!=	ungleich	$zahl != 5
<	kleiner	$zahl < 5
<=	kleiner oder gleich	$zahl <= 5
>	größer	$zahl > 5
>=	größer oder gleich	$zahl >= 5
&&	logisches Und	$zahl <= 1 && $zahl > 4
||	logisches Oder	$zahl == 0 || $zahl < 0
!	logische Negation	!($zahl <= 1 && $zahl > 4)
`is_numeric( $var )`	Variable Zahl?	
`isset( $var )`	Existiert Variable?	
`empty( $var )`	Variable leer?	

1.7.2 IF

1.7.3 Schleifen

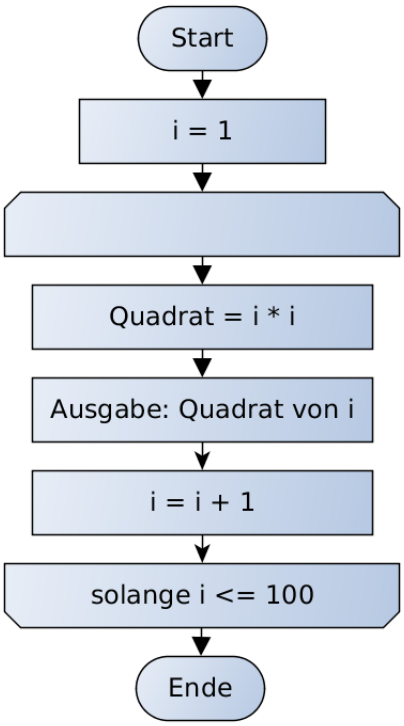
In Programmiersprachen stellt eine Schleife ein Kontrollstruktur dar. Sie wiederholt einen Anweisungsblock, den sogenannten Schleifenkörper, solange, wie eine Bedingung gültig ist oder bis eine Abbruchbedingung eintritt. Es werden vorprüfende und nachprüfende Schleifen unterschieden. Zu den vorprüfenden Schleifen gehören die kopfgesteuerte WHILE-Schleife, die FOR-Schleife und die Mengenschleife (FOREACH). Die fußgesteuerte WHILE-Schleife fällt zusammen mit der REPEAT-Schleife unter die nachprüfenden Schleifen.

FOR - Zählschleife



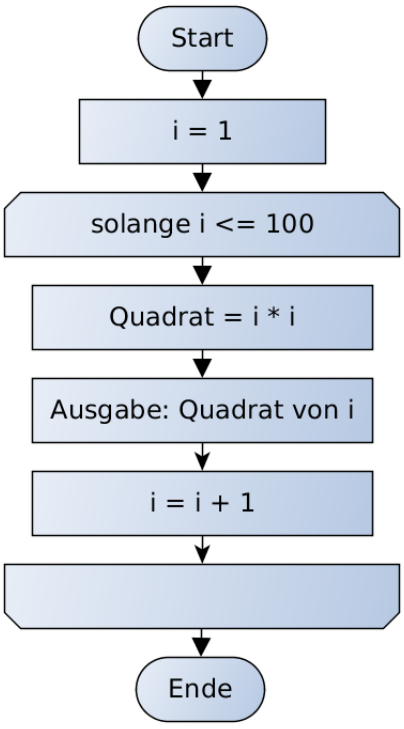
WHILE

Die kopfgesteuerten WHILE-Schleifen unterscheiden sich von den fußgesteuerten dadruch, dass die Bedingung der Schleife bei letzterer erst nach dem einmaligen Durchlaufen der Schleife geprüft wird. Das bedeutet, dass fußgesteuerte Schleifen wie REPEAT immer mindestens einmal ausgeführt werden. Überspitzt lässt sich sagen, dass REPEAT erst schießt und dann fragt.



WHILE (Fuß)

```
i = 1
    Quadrat = i * i
    Ausgabe: Quadrat
    i = i + 1
solange i <= 100
```



WHILE (Kopf)

```
i = 1
solange i <= 100
    Quadrat = i * i
    Ausgabe: Quadrat
    i = i + 1
```

1.7.4 Switch-Case

```
<?php
 switch ($i)
 {
  case 1:
   echo "1";
   break;
  case 2:
```

```
    echo "2";
    break;
  }
?>
```

Listing 4: Ein Beispiel für Switch-Case-Anweisungen

1.7.5 Arrays

Der Datentyp Array kann beliebig viele, gleichartige Werte speichern. Im Gegensatz dazu können Variablen nur je einen Wert speichern.

Indexadressierte Arrays

Indexadressierte Arrays arbeiten mit Indizes, um auf die gespeicherten Werte zu verweisen. Im Listing [Nr] ist die Zuordnung und Ausgabe von indexadressierten Arrays am Beispiel von PHP gezeigt. Darin zeigt sich der Charakter eines Indizes: um einen Wert auszugeben, muss der entsprechende Index referenziert werden. Bevor eine Variable als Array verwendet werden kann, muss sie als solches definiert werden. Dies geschieht mit der Funktion `array()`. Eine Eigenheit von PHP besteht darin, dass es automatisch erkennt, dass eine Variable als Array benutzt werden soll. D.h., der Code aus dem Listing wäre auch ohne die Definition der Variablen als Arrays fehlerfrei. Im Listing zeigt sich auch die Eigenschaft von PHP, Arrays ohne explizite Indizes einen Index beginnend bei 0 zu zuordnen. In anderen Programmiersprachen muss der Index immer explizit angegeben werden.

```
<?php
  $ein_array = array();
  $ein_array[0] = $_POST["txt_eingabe_1"];
  $ein_array[1] = $_POST["txt_eingabe_2"];

  $noch_ein_array = array();
  $noch_ein_array[] = $_POST["txt_eingabe_1"];
  $noch_ein_array[] = $_POST["txt_eingabe_2"];

  $alternatives_array = array("$_POST["txt_eingabe_1"]", \
                              "$_POST["txt_eingabe_2"]");

  echo '$ein_array[0] gibt den Wert '.$ein_array[0].' aus.<br />';
  echo '$noch_ein_array[0] gibt ebenfalls den Wert '.$noch_ein_array[0]. \
       ' aus.<br />';
?>
```

Listing 5: Indexadressierte Arrays in PHP

Assoziative Arrays

1.7.6 Aufgaben und Beispiele