

Junioraufgabe - Maya-Zahlen

Lösungsidee

Die Lösung der Idee liegt schon auf der Hand, da in der Aufgabe beschrieben ist, was man zu tun hat: Dezimalzahlen müssen in Zahlen der Basis 20 umgerechnet werden.

Man nehme die eingegebene Zahl n .

Wir prüfen jetzt, wie oft die Zahl durch 20 teilbar ist.

$$n = q_1 \cdot 20 + r_1$$

Diese Überprüfung geschieht mit der Modulo-Funktion, die uns den Rest aus einer Ganzzahldivision ausspuckt. Der eventuell bleibende Rest r_1 bildet unsere letzte Ziffer der Zahl im System zur Basis 20, sonst ist das die Null.

Mit dem Ganzzahlquotienten q_1 machen wir das Gleiche wie mit n : die Teilbarkeit durch 20 prüfen:

$$q_1 = q_2 \cdot 20 + r_2$$

Bleibt auch hier ein Rest r_2 , so wird damit unsere vorletzte Ziffer der Maya-Zahl gebildet, ansonsten ist das auch hier die Null.

Allgemein können wir sagen:

$$q_n = q_{n+1} \cdot 20 + r_{n+1}$$

Wir rechnen so lange, bis $q_{n+1} = 0$, wir also nur noch einen Rest r_{n+1} bekommen und keinen Ganzzahlquotienten mehr. Wobei r_{n+1} dann die erste Ziffer unserer Zahl bildet. Da wir jetzt alle Ziffern der Maya-Zahl haben sind wir fertig. Für die Ausgabe der Zahl geben wir die Ziffern in ihrer korrekten Reihenfolge (von r_{n+1} bis r_1) einfach hintereinander aus.

Programm-Dokumentation

Für die Umsetzung haben wir uns ein PHP-Skript gebastelt – quick & dirty!
Die Eingabe erfolgt über die Adresszeile – wir übergeben unsere Zahl n also per GET-Methode an den Server.

Wir rufen also beispielsweise auf dem lokalen Webserver in unserem Lieblingsbrowser die URL <http://127.0.0.1/maya/?n=42> auf.

Das Skript könnte jetzt noch die Gültigkeit der Eingabe überprüfen. Dazu könnte uns der reguläre Ausdruck † dienen – jedes Zeichen der Eingabe müsste also eine Ziffer [0-9] sein und die Eingabe mindestens ein Zeichen lang. PHP stellt aber zum Beispiel auch die Funktion `is_int()` zur Verfügung und wir wollen im Prinzip ja nur natürliche Zahlen in der Eingabe (und genau das ist der Datentyp integer ja). Allerdings können wir diese nicht verwenden, da die GET-Variablen automatisch als Zeichenketten (strings) angesehen werden. Also hätten wir da noch die Funktion `is_numeric()`, die prüft, ob eine Variable eine Zahl oder ein numerischer string ist. Allerdings hilft uns das auch nicht wirklich weiter, denn auch andere Zahlen als Ganzzahlen werden hier durchgelassen. Bei beiden Funktionen müssen wir des Weiteren noch prüfen, ob diese größer Null sind – eine negative Zahl lässt unseren Webserver in einer Endlosschleife rechnen und hängt sich dann irgendwann ausgelastet aus. Ausweg: man nehme den Betrag der Eingabe – oder wir greifen einfach wieder auf unseren regulären Ausdruck zurück.

Aber zurück zum Skript, wie es ist:

In einer Schleife errechnen wir mittels Modulfunktion den Teilerrest von sq mit 20 und speichern ihn in der Variablen sr . sq ist dabei der Ganzzahlquotient des letzten Durchlaufs. Der Rest sr wird im Array $\$stellen$ abgespeichert. Jetzt berechnen wir den Ganzzahlquotient – dieser ergibt sich durch Abrunden des Ergebnisses das bei der Division durch 20 entsteht. Mit diesem sq können wir also im nächsten Durchlauf den neuen Rest sr berechnen. Die Schleife läuft so lange sq nicht Null wird.

Anschließend müssen wir das Array $\$stellen$ in der umgekehrten Reihenfolge sortieren, denn bisher war das erste Element die letzte Ziffer, das zweite Element die vorletzte Ziffer usw.; deswegen drehen wir die Reihenfolge der Elemente mit der Funktion `krsort()` um!

Anschließend gehen wir in einer foreach-Schleife alle Elemente des Arrays durch und können so die einzelnen Ziffern ausgeben. Wir realisieren das indem wir HTML-Code erzeugen und jede Ziffer als kleines Bild aus (geliehen von <http://www.mathezentrale.de/maya/maya1.htm>; gespeichert als gif) mit einem Zeilenumbruch nach jeder Ziffer/ jedem Bild.

Auf eventuell nötige HTML-Tags wie `<html>`, `<body>`,... wurde verzichtet. Die Methode ist wie gesagt quick & dirty. Es wäre natürlich kein Problem den Code zu konformem HTML zu erweitern.

Alternativ kam uns auch in den Sinn die Zahl dann als ganzes Bild mit Hilfe der GDLib zu erstellen. In unseren Augen reicht die Darstellung auf diese Weise allerdings, da ja nur eine „grafische Darstellung“ gewünscht war.

Programm-Ablaufprotokoll

Abgesehen vom „normalen Skript“ (`index.php`) haben wir noch ein weiteres mitgeliefert (`dokumentierte_ausgabe.php`). Dieses ist allerdings nur eine Erweiterung des ersten: die Ausgabe beinhaltet nicht nur die Bilder der Ziffern der Maya-Zahl, sondern auch noch, wie diese in den einzelnen Schritten entstanden.

3 Beispiele für die dokumentierte Ausgabe:

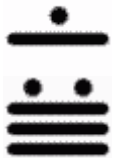
(1) $n = 1337$

Aufruf von http://127.0.0.1/maya/dokumentierte_ausgabe.php?n=1337:

```
Eingegeben: 1337
1337 = q*20 + r
  Rest r: 17
  Ganzzahlquotient q: 66
66 = q*20 + r
  Rest r: 6
  Ganzzahlquotient q: 3
3 = q*20 + r
  Rest r: 3
  Ganzzahlquotient q: 0
Ziffern in umgekehrter Reihenfolge:
Array
(
    [0] => 17
    [1] => 6
    [2] => 3
)
Ziffern in richtiger Reihenfolge:
Array
(
    [2] => 3
    [1] => 6
    [0] => 17
)

```

• • •

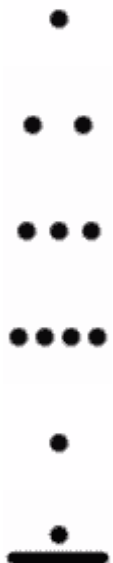


(2) $n = 3545626$

Aufruf von http://127.0.0.1/maya/dokumentierte_ausgabe.php?n=3545626:

```
Eingegeben: 3545626
3545626 = q*20 + r
  Rest r: 6
  Ganzzahlquotient q: 177281
177281 = q*20 + r
  Rest r: 1
  Ganzzahlquotient q: 8864
8864 = q*20 + r
  Rest r: 4
  Ganzzahlquotient q: 443
443 = q*20 + r
  Rest r: 3
  Ganzzahlquotient q: 22
22 = q*20 + r
  Rest r: 2
  Ganzzahlquotient q: 1
1 = q*20 + r
  Rest r: 1
  Ganzzahlquotient q: 0
```

```
Ziffern in umgekehrter Reihenfolge:
Array
(
    [0] => 6
    [1] => 1
    [2] => 4
    [3] => 3
    [4] => 2
    [5] => 1
)
Ziffern in richtiger Reihenfolge:
Array
(
    [5] => 1
    [4] => 2
    [3] => 3
    [2] => 4
    [1] => 1
    [0] => 6
)
```



(3) $n = 524287 (= \text{Primzahl } 2^{19}-1)$

Aufruf von [http://127.0.0.1/maya/dokumentierte_ausgabe.php?n=524287:](http://127.0.0.1/maya/dokumentierte_ausgabe.php?n=524287)

```
Eingegeben: 524287
524287 = q*20 + r
  Rest r: 7
  Ganzzahlquotient q: 26214
26214 = q*20 + r
  Rest r: 14
  Ganzzahlquotient q: 1310
1310 = q*20 + r
  Rest r: 10
  Ganzzahlquotient q: 65
65 = q*20 + r
  Rest r: 5
  Ganzzahlquotient q: 3
3 = q*20 + r
  Rest r: 3
  Ganzzahlquotient q: 0
```

```
Ziffern in umgekehrter Reihenfolge:
Array
(
    [0] => 7
    [1] => 14
    [2] => 10
    [3] => 5
    [4] => 3
)
Ziffern in richtiger Reihenfolge:
Array
(
    [4] => 3
    [3] => 5
    [2] => 10
    [1] => 14
    [0] => 7
)
```



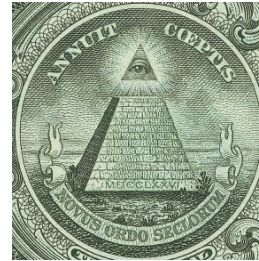
Und jetzt noch 3 Beispiele für die normale/ nicht ausführlich dokumentierte Ausgabe:

(1) $n = 23$

Aufruf von <http://127.0.0.1/maya/?n=23>:



Na wer hätte das gedacht – das erinnert uns doch
glatt daran:



(2) $n = 42$

Aufruf von <http://127.0.0.1/maya/?n=42>:



(3) $n = 638$ (unsere Teilnehmernummer)

Aufruf von <http://127.0.0.1/maya/?n=638>:



(4) $n = 25031988$

Aufruf von
<http://127.0.0.1/maya/?n=25031988>:



(5) $n = 1000$

Aufruf von <http://127.0.0.1/maya/?n=1000>:



Ein Gesicht? : || 0

Programm-Text

index.php

```
1 <?
2 # Ganzzahl auslesen und der Variablen $n zuweisen
3 $n = $_GET['n'];
4
5 # Array mit den Ziffern der Zahl erstellen
6 $stellen = array();
7
8 # für den ersten Durchlauf gilt
9 # Ganzzahlquotient = eingegebene Ganzzahl;
10 $q = $n;
11
12 # Abbruchbedingung; Ganzzahlquotient <> 0
13 while($q <> 0) {
14
15     # Rest berechnen
16     $r = $q%20;
17
18     # Stelle/Ziffer im Array speichern
19     $stellen[] = $r;
20
21     # Ganzzahlquotient berechnen
22     $q = floor($q/20);
23
24 }
25
26 # Array in umgekehrter Reihenfolge sortieren
27 # damit Ziffernreihenfolge stimmt
28 krsort($stellen);
29
30 # Ziffern nacheinander ausgeben; und zwar
31 # als Bild mit Hilfe des img-Tags in HTML
32 foreach($stellen as $stelle)
33     echo '<br>';
34
35 ?>
```

dokumentierte_ausgabe.php

```
1 <pre><?
2 # Ganzzahl auslesen und der Variablen $n zuweisen
3 $n = $_GET['n'];
4
5 # eingegebene Zahl ausgeben
6 echo "Eingegeben: $n<br>";
7
8 # Array mit den Ziffern der Zahl erstellen
9 $stellen = array();
10
11 # für den ersten Durchlauf gilt
12 # Ganzzahlquotient = eingegebene Ganzzahl;
13 $q = $n;
14
15 # Abbruchbedingung; Ganzzahlquotient <> 0
16 while($q <> 0) {
17
18     echo "$q = q*20 + r<br>";
19
20     # Rest berechnen
21     $r = $q%20;
22
23     # Stelle/Ziffer im Array speichern
24     $stellen[] = $r;
25
26     # Ganzzahlquotient berechnen
27     $q = floor($q/20);
28
29     echo "  Rest r: $r<br>",
30         "  Ganzzahlquotient q: $q<br>";
31
32 }
33
34 # Ziffern ausgeben
35 echo "<br>Ziffern in umgekehrter Reihenfolge:<br>";
36 print_r($stellen);
37
38 # Array in umgekehrter Reihenfolge sortieren
39 # damit Ziffernreihenfolge stimmt
40 krsort($stellen);
41
42 # Ziffern ausgeben
43 echo "<br>Ziffern in richtiger Reihenfolge:<br>";
44 print_r($stellen);
45
46 # Ziffern nacheinander ausgeben; und zwar
47 # als Bild mit Hilfe des img-Tags in HTML
48 foreach($stellen as $stelle)
49     echo '<img src="", $stelle, ".gif"><br>';
50
51 ?></pre>
```