

# Zahlenbasen Umwandlung – in Theorie und Praxis

Alexander Hermann

26. April 2016



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>9</b>
1.1	Zahlendarstellung . . . . .	9
1.2	Hier mögliche Zahlen . . . . .	11
<b>2</b>	<b>Umwandlungen</b>	<b>13</b>
2.1	Umwandlung von Zahlen der Basis $b$ in das Zahlensystem der Basis 10 . .	13
2.1.1	Allgemeine Formel zur Wandlung von Basis $b$ zu Basis 10 . . . . .	13
2.1.2	Berechnungsablauf . . . . .	13
2.2	Umwandlung von Zahlen der Basis 10 in das Zahlensystem der Basis $b$ . .	16
2.2.1	Allgemeine Formel zur Wandlung von Basis 10 zu Basis $b$ . . . . .	16
2.2.2	Berechnungsablauf . . . . .	16
<b>A</b>	<b>Beispiele</b>	<b>19</b>
A.1	Umwandlung ins Dezimalsystem . . . . .	19
A.1.1	Beispiel der Zahlenbasis $b_1 = 2$ . . . . .	19
A.1.2	Beispiel der Zahlenbasis $b_1 = 8$ . . . . .	20
A.1.3	Beispiel der Zahlenbasis $b_1 = 16$ . . . . .	20
A.2	Umrechnung vom Dezimalsystem in andere Zahlensysteme . . . . .	21
A.2.1	Beispiel der Zahlenbasis $b_2 = 2$ . . . . .	21
A.2.2	Beispiel der Zahlenbasis $b_2 = 3$ . . . . .	23
A.2.3	Beispiel der Zahlenbasis $b_2 = 8$ . . . . .	24
<b>B</b>	<b>Umsetzung in Programmiersprachen</b>	<b>27</b>
B.1	JAVA-Codierung . . . . .	27
B.2	PHP-Codierung . . . . .	27
B.2.1	Interface . . . . .	27
B.2.2	Implementierung des Interfaces: . . . . .	28
B.3	C#-Codierung . . . . .	32
B.3.1	Interface . . . . .	32
B.3.2	Exception . . . . .	32
B.3.3	Implementierung des Interfaces . . . . .	33



# Listings

B.1	JAVA implementierung der Zahlenbasis . . . . .	27
B.2	PHP Interface der Zahlenbasis . . . . .	27
B.3	PHP Implementierung der Zahlenbasis . . . . .	28
B.4	C-Sharp Interface der Zahlenbasis . . . . .	32
B.5	C-Sharp ZahlensystemException . . . . .	32
B.6	C-Sharp Implementierung des Interfaces . . . . .	33



# Abbildungsverzeichnis

2.1	Struktogramm Umwandlung in das Dezimalsystem . . . . .	14
2.2	Struktogramm Berechnung des Character-Werts . . . . .	15
2.3	Struktogramm Berechnung des Zwischenergebnisses . . . . .	15
2.4	Struktogramm Umwandlung vom Dezimalsystem . . . . .	17
2.5	Struktogramm Umwandlung einer Zahl in einen Character-Wert . . . . .	18





# Kapitel 1

## Einleitung

Zahlen in verschiedenen Zahlenbasen werden im Wesentlichen für eine Vorvereinfachung zur menschlichen Kommunikation bzw. zur Umschreibung mit maschinellen Automatisierungen verwendet. Bei der Ausführung von Software auf reiner Hardware-Ebene läuft alles letztendlich rein binär<sup>1</sup> ab.

Da wir Menschen es gewohnt sind im Dezimalsystem<sup>2</sup> zu rechnen – was möglicherweise daran liegt, dass der Mensch zehn Finger hat – und auch dafür ausgebildet wurden, ist es im Allgemeinen einfacher, auf dieser Basis zu rechnen.

### 1.1 Zahlendarstellung

In vielen Programmiersprachen werden die Zahlensysteme **binär**, **octal**, **dezimal** und **hexadezimal** im Programmiercode zur Vereinfachung bzw. zur korrekten Interpretation durch den Compiler unterschiedlich eingegeben.

- **binär:** 0b101110
- **octal:** 0c576302
- **dezimal:** 964
- **hexadezimal:** 0xAFFE09

Da hier aber generell alle möglichen Zahlensysteme verwendet werden, bzw. die verallgemeinerte Form der Umrechnung erklärt werden soll, werden im Folgenden Zahlen eines bestimmten Zahlensystems der Basis  $b$  wie folgt dargestellt:

$$bxv$$

Wobei  $b$  für die entsprechende Basis steht,  $x$  zur Markierung immer als **x** verwendet wird und  $v$  der Wert im entsprechenden Zahlensystem ist. Ähnlich wie oben:

---

<sup>1</sup>auf der Basis  $b = 2$

<sup>2</sup>Basis  $b = 10$

- **binär:** 2x101110
- **octal:** 8x576302
- **dezimal:** 10x964
- **hexadezimal:** 16xAFFE09

Wäre das alles, wäre es wohl kaum nötig eine zusätzliche Darstellung zu verwenden. Aber an eher seltenen Zahlensystemen, ist eine generalisierte Darstellung dann doch vorteilhaft:

- **ternär:** 3x211201
- **quinär:** 5x402314
- **tridezimal** 13x5A9C0B3
- **oktovigesimal** 28xNOR7OKRANK
- **hexatridezimal:** 36xGIRAFFE0Z6A

Diese Methode der Darstellung wählte ich, da es eine Ähnlichkeit zur oben gezeigten Softwareversion hat.

Eine Andere, erwähnenswerte Alternative, wäre eine Darstellung, wie sie z.B. im „*Taschenbuch mathematischer Formeln und moderner Verfahren*“ [Stö99] verwendet wird<sup>3</sup>:

- **binär:** 101110<sub>(2)</sub>
- **ternär:** 211201<sub>(3)</sub>
- **quinär:** 402314<sub>(5)</sub>
- **octal:** 576302<sub>(8)</sub>
- **dezimal:** 964<sub>(10)</sub>
- **tridezimal** 5A9C0B3<sub>(13)</sub>
- **hexadezimal:** AFFE09<sub>(16)</sub>
- **oktovigesimal** NOR7OKRANK<sub>(28)</sub>
- **hexatridezimal:** GIRAFFE0Z6A<sub>(36)</sub>

---

<sup>3</sup>gleiche Zahlen wie oben

## 1.2 Hier mögliche Zahlen

Es gibt auch durchaus Umwandlungsmethoden, um Reale Zahlen umzurechnen. Hier wird aber nur mit Natürlichen Zahlen gearbeitet. Bei Stöcker [Stö99, Seite 4] wird folgendes für die „Darstellung einer Zahl  $z$  im Zahlensystem zur Basis  $B$ “ dargestellt:

$$Z_{(B)} = \sum_{i=-m}^n z_i B^i, \quad B \in \mathbb{N}, \quad B \geq 2$$



# Kapitel 2

## Umwandlungen

### 2.1 Umwandlung von Zahlen der Basis $b$ in das Zahlensystem der Basis 10

#### 2.1.1 Allgemeine Formel zur Wandlung von Basis $b$ zu Basis 10

Diese Formel ist für Zahlenbasen der Basis  $b = 2$  bis Basis  $b = 36^1$  mit den Ziffern 0 bis 9 und den Buchstaben  $A$  bis  $Z$  möglich. Die Formel setzt sich zusammen aus der Basis  $b$ , der Stellenposition<sup>2</sup>  $s$  und dem angezeigten Wert  $w$ . Wenn der Wert ein Buchstabe ist, ist der Wert gleich Buchstabenstelle  $bu_s$  im Alphabet  $+9$   $w = bu_s + 9$  ansonsten der Zahlenwert  $w = w$ . Die Anzahl der maximalen Zeichen ist der Basiswert.<sup>3</sup> Im „normalen“, dezimalen Zahlensystem von 0 bis 9 ist die 10 bereits *zweistellig*.

$$x_s = b^{s-1} * w$$

**Merke:**  
das erste  
Zeichen  
(2.1) ist immer  
0!

Die Ergebnisse der einzelnen Stellen werden summiert.

#### 2.1.2 Berechnungsablauf

Der Berechnungsablauf kann wie in dem, in Abbildung 2.1 dargestellten Struktogramm dargestellt werden. Die meisten Programmiersprachen haben vordefinierte Funktionen zur Längenberechnung von `string`-Variablen; ebenso gibt es Funktionen um an bestimmten Stellen eines Strings einzelne Zeichen abzurufen. Damit entfällt die genauere Beschreibung der Längenabfrage und der Stellenabfrage. Was hier noch fehlt, ist die Berechnung des Character-Werts, dies wird im Struktogramm in Abbildung 2.2 dargestellt, falls es sich **nicht** um eine Zahl handelt, so wie die Berechnung des Zwischenergebnisses, welches im Struktogramm in Abbildung 2.3 dargestellt wird.

---

<sup>1</sup>Basis 36 bei ASCII; bei UTF-8 auch größer

<sup>2</sup>von rechts nach links

<sup>3</sup>Deswegen ist 2 auch die kleinstmögliche Zahlenbasis, weil bei nur einem Zeichen kein Unterschied mehr möglich ist.

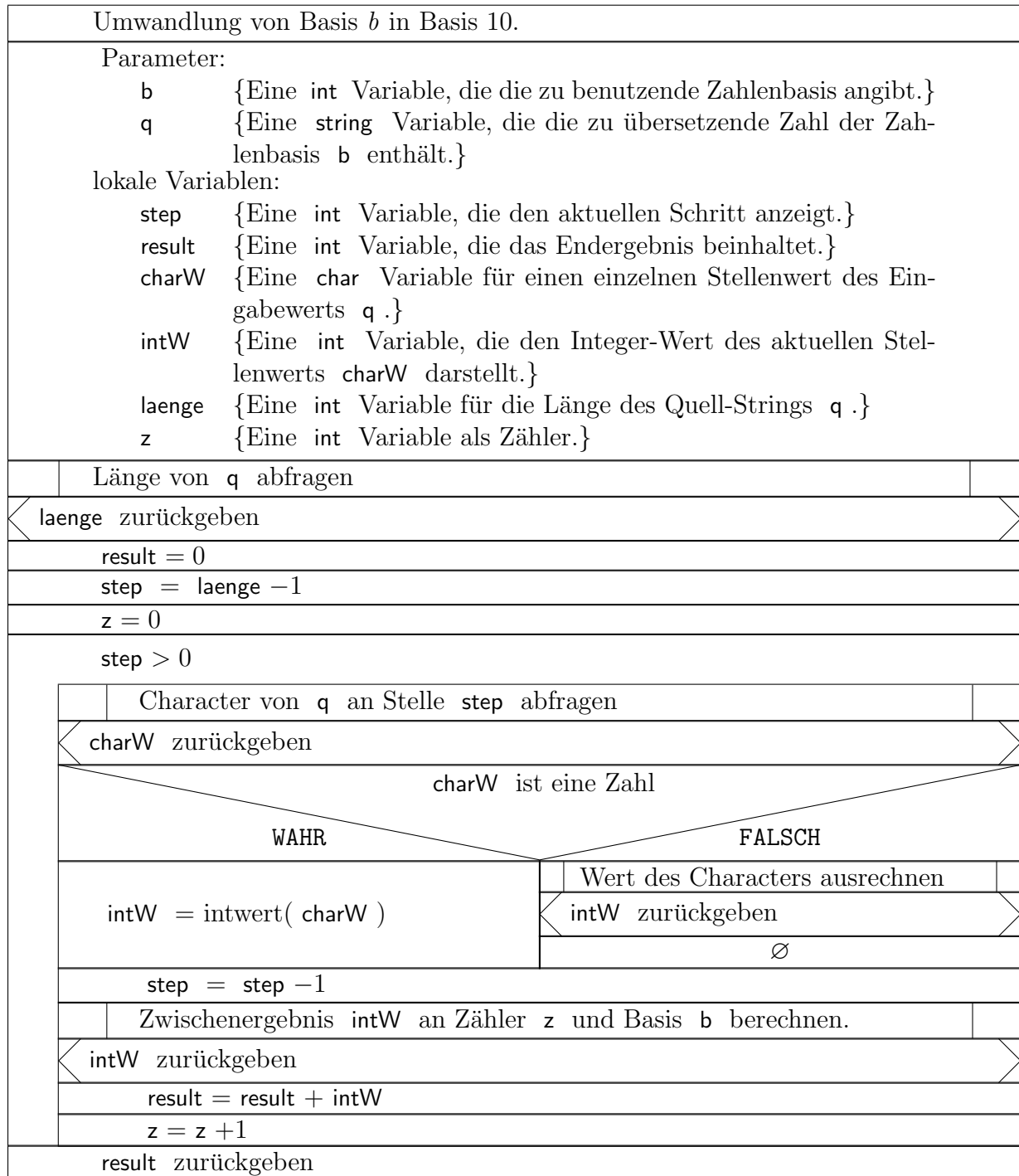


Abbildung 2.1: Struktogramm Umwandlung in das Dezimalsystem



## 2.2 Umwandlung von Zahlen der Basis 10 in das Zahlensystem der Basis $b$

Eine der Anleitungen fand ich im Web<sup>4</sup>. Die umzurechnende Zahl  $z$  wird durch die Basis  $b$  geteilt; der Quotient  $q$  wird zur erneuten Rechnung verwendet; der jeweilige Rest  $r$  wird mit 10 hoch dem Rechenschritt  $s$  multipliziert; der erste Rechenschritt ist  $s = 0$ . Es wird so häufig gerechnet, bis der Quotient 0 ist.

### 2.2.1 Allgemeine Formel zur Wandlung von Basis 10 zu Basis $b$

In dieser Formel wird der Quotient des vorherigen Rechenschritts als das Zwischenergebnis  $s_n$  bezeichnet, wobei  $n$  die Nummer des Rechenschritts ist. Die Zählung der Rechenschritte fängt mit 0 an. Also ist für die erste Stelle  $s_0$  die umzuwandelnde Zahl  $z$  der Basis  $b$  zu verwenden.

$$\frac{s_n}{b} = q_n; r_n \quad (2.2)$$

### 2.2.2 Berechnungsablauf

Nach dem ersten Rechenschritt (wenn der Quotient  $q \neq 0$  ist),  $n \geq 1$  gilt:

$$s_n = q_{n-1} \quad (2.3)$$

Das Gesamtergebnis  $g$  ergibt sich wie folgt, wenn die Zielbasis  $b < 10$  ist:

$$g = r_0 * 10^0 + r_1 * 10^1 \cdots + r_n * 10^n \quad (2.4)$$

Wenn die Zielbasis  $b > 10$  ist, müssen die einzelnen Zeichendarstellungen der Reste  $r_n$  rückwärts in eine Zeichenfolge zusammengesetzt werden. Das Ganze ist auch im Struktogramm in Abbildung 2.4 dargestellt. Die Umwandlung einer Zahl  $> 9$  läuft ähnlich wie im Struktogramm in Abbildung 2.2; nur umgekehrt. Siehe dazu das Struktogramm in Abbildung 2.5

---

<sup>4</sup>[www.arndt-bruenner.de](http://www.arndt-bruenner.de) [Brü15]



## 2.2. UMWANDLUNG VON ZAHLEN DER BASIS 10 IN DAS ZAHLENSYSTEM DER BASIS B17

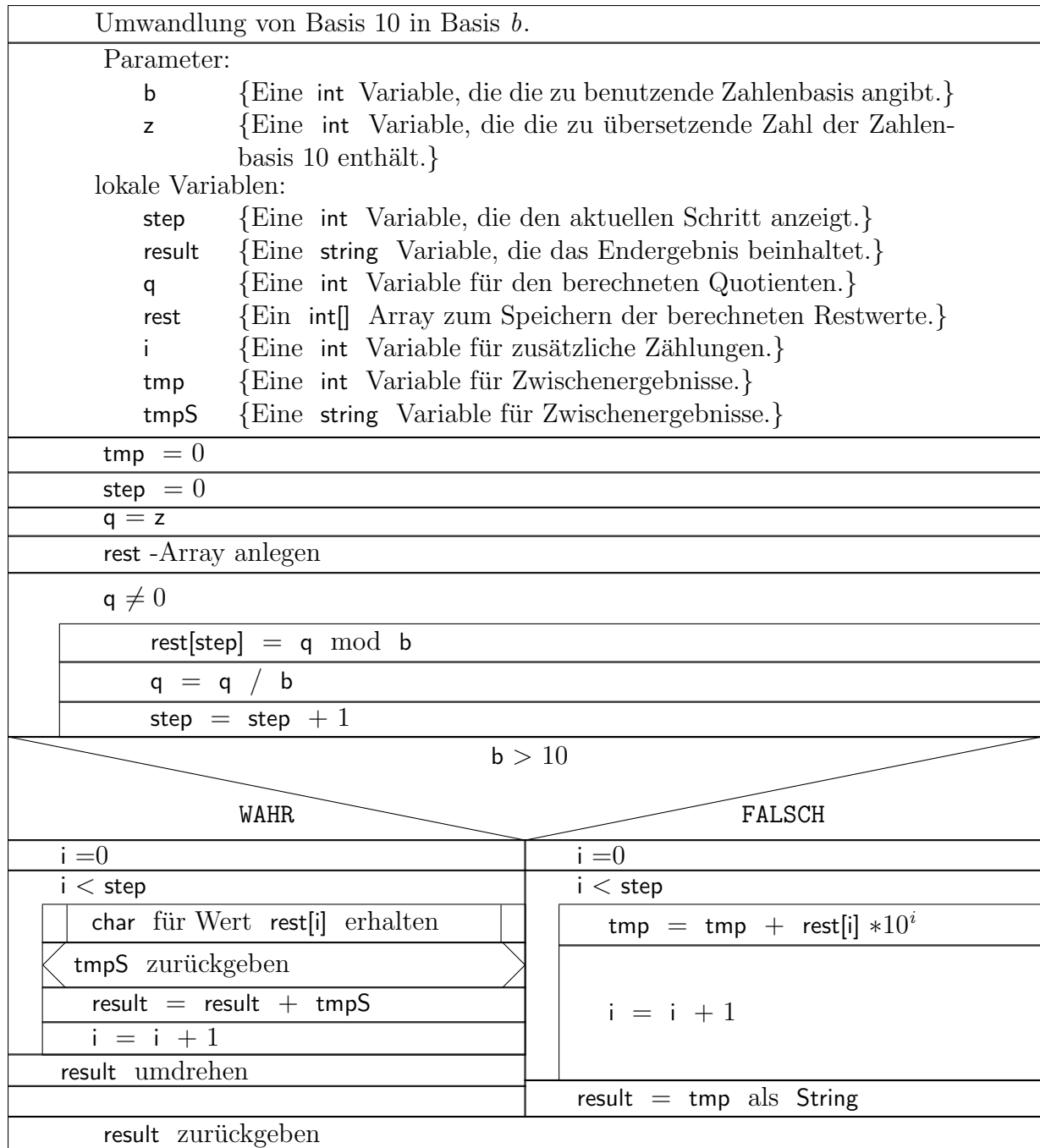


Abbildung 2.4: Struktogramm Umwandlung vom Dezimalsystem

Berechnung eines Character-Werts
Parameter: $x$ {Eine <code>int</code> Variable, die den auszuwertenden wert angibt.} lokale Variablen: <code>result</code> {Eine <code>char</code> Variable, die das Endergebnis beinhaltet.} <code>z</code> {Eine <code>int</code> Variable als Zwischenwert.}
Lese den <b>ASCII</b> oder <b>UTF-8</b> Wert des Buchstabens <b>A</b> aus und schreibe es in das Zwischenergebnis <code>z</code> .
<code>result = x - 9 + z</code>
<code>result</code> zurückgeben

Abbildung 2.5: Struktogramm Umwandlung einer Zahl in einen Character-Wert

# Anhang A

## Beispiele

### A.1 Umwandlung ins Dezimalsystem

Beispiele der Umrechnung von der Zahlenbasis  $b_1 = x$  in die Zahlenbasis  $b_2 = 10$ .

#### A.1.1 Beispiel der Zahlenbasis $b_1 = 2$

Im Binärsystem gibt es die zwei Zeichen 0 und 1.

##### Beispiel 1

Die Binärzahl  $2x100$  wird wie folgt nach Formel 2.1 umgerechnet: von rechts nach links:

1. An Stelle  $s = 1$ :

$$x_1 = 2^0 * 0 = 1 * 0 = 0$$

2. An Stelle  $s = 2$ :

$$x_2 = 2^1 * 0 = 2 * 0 = 0$$

3. An Stelle  $s = 3$ :

$$x_3 = 2^2 * 1 = 4 * 1 = 4$$

Die Summierung von  $x_1$  bis  $x_3$  ist:

$$0 + 0 + 4 = 4$$

##### Beispiel 2

Die Binärzahl  $2x110101$  wird wie folgt nach Formel 2.1 umgerechnet: von rechts nach links:

1. An Stelle  $s = 1$ :

$$x_1 = 2^0 * 1 = 1 * 1 = 1$$

2. An Stelle  $s = 2$ :

$$x_2 = 2^1 * 0 = 2 * 0 = 0$$

3. An Stelle  $s = 3$ :

$$x_3 = 2^2 * 1 = 4 * 1 = 4$$

4. An Stelle  $s = 4$ :

$$x_4 = 2^3 * 0 = 8 * 0 = 0$$

5. An Stelle  $s = 5$ :

$$x_5 = 2^4 * 1 = 16 * 1 = 16$$

6. An Stelle  $s = 6$ :

$$x_6 = 2^5 * 1 = 32 * 1 = 32$$

.

Die Summierung von  $x_1$  bis  $x_6$  ist:

$$1 + 0 + 4 + 0 + 16 + 32 = 53$$

### A.1.2 Beispiel der Zahlenbasis $b_1 = 8$

Im Oktalsystem gibt es acht Zeichen von 0 bis 7.

#### Beispiel 1

Die Oktalzahl  $8x70$  wird wie folgt nach Formel 2.1 umgerechnet: von rechts nach links:

1. An Stelle  $s = 1$ :

$$x_1 = 8^0 * 0 = 1 * 0 = 0$$

2. An Stelle  $s = 2$ :

$$x_2 = 8^1 * 7 = 8 * 7 = 56$$

Die Summierung von  $x_1$  bis  $x_2$  ist:

$$0 + 56 = 56$$

### A.1.3 Beispiel der Zahlenbasis $b_1 = 16$

Im Hexadezimalsystem gibt es sechzehn Zeichen von 0 bis  $F$ .

**Beispiel 1**

Die Hexadezimalzahl  $16xD4$  wird wie folgt nach Formel 2.1 von rechts nach links umgerechnet:

1. An Stelle  $s = 1$ :

$$x_1 = 16^0 * 4 = 1 * 4 = 4$$

2. An Stelle  $s = 2$ :

$$x_2 = 16^1 * (4 + 9) = 16 * 13 = 208$$

Die Summierung von  $x_1$  bis  $x_2$  ist:

$$4 + 208 = 212$$

**Beispiel 2**

Die Hexadezimalzahl  $16xAFFE$  wird wie folgt nach Formel 2.1 von rechts nach links umgerechnet:

1. An Stelle  $s = 1$ :

$$x_1 = 16^0 * (5 + 9) = 1 * 14 = 14$$

2. An Stelle  $s = 2$ :

$$x_2 = 16^1 * (6 + 9) = 16 * 15 = 240$$

3. An Stelle  $s = 3$ :

$$x_3 = 16^2 * (6 + 9) = 256 * 15 = 3840$$

4. An Stelle  $s = 4$ :

$$x_4 = 16^3 * (1 + 9) = 4096 * 10 = 40960$$

Die Summierung von  $x_1$  bis  $x_4$  ist:

$$14 + 240 + 3840 + 40960 = 45054$$

## A.2 Umrechnung vom Dezimalsystem in andere Zahlensysteme

Beispiele der Umrechnung von der Zahlenbasis  $b_1 = 10$  in die Zahlenbasis  $b_2 = x$ .

### A.2.1 Beispiel der Zahlenbasis $b_2 = 2$

Die Zahlenbasis nennt sich **Binär**.

**Beispiel 1**

Die Dezimalzahl  $z = 13$  wird wie folgt nach Formel 2.2 umgerechnet:

1. An Stelle 1:  $n = 0$ :

$$\frac{s_0 = z}{2} = \frac{13}{2} = q_0 = 6; r_0 = 1$$

2. An Stelle 2:  $n = 1$ :

$$\frac{s_1 = q_0}{2} = \frac{6}{2} = q_1 = 3; r_1 = 0$$

3. An Stelle 3:  $n = 2$ :

$$\frac{s_2 = q_1}{2} = \frac{3}{2} = q_2 = 1; r_2 = 1$$

4. An Stelle 4:  $n = 3$ :

$$\frac{s_3 = q_2}{2} = \frac{1}{2} = q_3 = 0; r_3 = 1$$

5. Gesamtergebnis  $g$  der Basis  $b = 2$ :

$$g = r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2 + r_3 * 10^3$$

$$g = 1 * 1 + 0 * 10 + 1 * 100 + 1 * 1000 = 1101$$

$$10x13 = 2x1101$$

**Beispiel 2**

Die Dezimalzahl  $z = 141$  wird wie folgt nach Formel 2.2 umgerechnet:

1. An Stelle 1:  $n = 0$ :

$$\frac{s_0 = z}{2} = \frac{141}{2} = q_0 = 70; r_0 = 1$$

2. An Stelle 2:  $n = 1$ :

$$\frac{s_1 = q_0}{2} = \frac{70}{2} = q_1 = 35; r_1 = 0$$

3. An Stelle 3:  $n = 2$ :

$$\frac{s_2 = q_1}{2} = \frac{35}{2} = q_2 = 17; r_2 = 1$$

4. An Stelle 4:  $n = 3$ :

$$\frac{s_3 = q_2}{2} = \frac{17}{2} = q_3 = 8; r_3 = 1$$

5. An Stelle 5:  $n = 4$ :

$$\frac{s_4 = q_3}{2} = \frac{8}{2} = q_4 = 4; r_4 = 0$$

6. An Stelle 6:  $n = 5$ :

$$\frac{s_5 = q_4}{2} = \frac{4}{2} = q_5 = 2; r_5 = 0$$

7. An Stelle 7:  $n = 6$ :

$$\frac{s_6 = q_5}{2} = \frac{2}{2} = q_6 = 1; r_5 = 0$$

8. An Stelle 8:  $n = 7$ :

$$\frac{s_6 = q_6}{2} = \frac{1}{2} = q_6 = 0; r_6 = 1$$

9. Gesamtergebnis  $g$  der Basis  $b = 2$ :

$$\begin{aligned} g &= r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2 + r_3 * 10^3 + r_4 * 10^4 \\ &\quad + r_5 * 10^5 + r_6 * 10^6 \\ g &= 1 * 1 + 0 * 10 + 1 * 100 + 1 * 1000 + 0 * 10000 \\ &\quad + 0 * 100000 + 0 * 1000000 + 1 * 10000000 \\ &= 10001101 \end{aligned}$$

$$10x141 = 2x10001101$$

### A.2.2 Beispiel der Zahlenbasis $b_2 = 3$

Die Zahlenbasis nennt sich **Ternär**.

#### Beispiel 1

Die Dezimalzahl  $z = 13$  wird wie folgt nach Formel 2.2 umgerechnet:

1. An Stelle 1:  $n = 0$ :

$$\frac{s_0 = z}{3} = \frac{13}{3} = q_0 = 4; r_0 = 1$$

2. An Stelle 2:  $n = 1$ :

$$\frac{s_1 = q_0}{3} = \frac{4}{3} = q_1 = 1; r_1 = 1$$

3. An Stelle 3:  $n = 2$ :

$$\frac{s_2 = q_1}{3} = \frac{1}{3} = q_2 = 0; r_2 = 1$$

4. Gesamtergebnis  $g$  der Basis  $b = 3$ :

$$\begin{aligned} g &= r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2 \\ g &= 1 * 1 + 1 * 10 + 1 * 100 = 111 \end{aligned}$$

$$10x13 = 3x111$$

**Beispiel 2**

Die Dezimalzahl  $z = 141$  wird wie folgt nach Formel 2.2 umgerechnet:

1. An Stelle 1:  $n = 0$ :

$$\frac{s_0 = z}{3} = \frac{141}{3} = q_0 = 47; r_0 = 0$$

2. An Stelle 2:  $n = 1$ :

$$\frac{s_1 = q_0}{3} = \frac{47}{3} = q_1 = 15; r_1 = 2$$

3. An Stelle 3:  $n = 2$ :

$$\frac{s_2 = q_1}{3} = \frac{15}{3} = q_2 = 5; r_2 = 0$$

4. An Stelle 4:  $n = 3$ :

$$\frac{s_3 = q_2}{3} = \frac{5}{3} = q_3 = 1; r_3 = 2$$

5. An Stelle 5:  $n = 4$ :

$$\frac{s_4 = q_3}{3} = \frac{1}{3} = q_4 = 0; r_4 = 1$$

6. Gesamtergebnis  $g$  der Basis  $b = 3$ :

$$\begin{aligned} g &= r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2 + r_3 * 10^3 + r_4 * 10^4 \\ g &= 0 * 1 + 2 * 10 + 0 * 100 + 2 * 1000 + 1 * 10000 \\ &= 12020 \end{aligned}$$

$$10x141 = 3x12020$$

**A.2.3 Beispiel der Zahlenbasis  $b_2 = 8$** 

Die Zahlenbasis nennt sich **Oktal**.

**Beispiel 1**

Die Dezimalzahl  $z = 13$  wird wie folgt nach Formel 2.2 umgerechnet:

1. An Stelle 1:  $n = 0$ :

$$\frac{s_0 = z}{8} = \frac{13}{8} = q_0 = 1; r_0 = 5$$

2. An Stelle 2:  $n = 1$ :

$$\frac{s_1 = q_0}{8} = \frac{1}{8} = q_1 = 0; r_1 = 1$$

3. Gesamtergebnis  $g$  der Basis  $b = 8$ :

$$\begin{aligned} g &= r_0 * 10^0 + r_1 * 10^1 \\ g &= 5 * 1 + 1 * 10 = 15 \end{aligned}$$

$$10x13 = 8x15$$



**Beispiel 2**

Die Dezimalzahl  $z = 141$  wird wie folgt nach Formel 2.2 umgerechnet:

1. An Stelle 1:  $n = 0$ :

$$\frac{s_0 = z}{8} = \frac{141}{8} = q_0 = 17; r_0 = 5$$

2. An Stelle 2:  $n = 1$ :

$$\frac{s_1 = q_0}{8} = \frac{17}{8} = q_1 = 2; r_1 = 1$$

3. An Stelle 3:  $n = 2$ :

$$\frac{s_2 = q_1}{8} = \frac{2}{8} = q_2 = 0; r_2 = 2$$

4. Gesamtergebnis  $g$  der Basis  $b = 8$ :

$$g = r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2$$

$$\begin{aligned} g &= 5 * 1 + 1 * 10 + 2 * 100 \\ &= 215 \end{aligned}$$

$$10x141 = 8x215$$



# Anhang B

## Umsetzung in Programmiersprachen

### B.1 JAVA-Codierung

Listing B.1: JAVA implementierung der Zahlenbasis

```
1 public class Zahlenbase
2 {
3 }
```

### B.2 PHP-Codierung

Angesehen werden kann die Umsetzung in PHP 5.x unter „<http://demo.hermann-bsd.de/zahlensysteme>“

#### B.2.1 Interface

Zuerst als abstraktes Interface fuer die Definition von Zahlenbasen

Listing B.2: PHP Interface der Zahlenbasis

```
1 namespace ahbsd\Zahlensysteme
2 {
3     /**
4      * Interface fuer grundlegende Funktionen, der Basis eines
5      * Zahlensystems.
6      *
7      * @author A. Hermann
8      * @copy Copyright &copy; 2016
9      * Alexander Hermann - Beratung, Software, Design
10     * Zahlensysteme
11     *
12     * @version 1.0
13     */
14     interface IBase
15     {
16         /**
17          * Gibt die Bezeichnung zurueck.
18          * @return string
19          */
20         function GetName();
```

---

<sup>1</sup>Einiges funktioniert da noch nicht...

```

22     /**
23      * Gibt das Zahlensystem als Integer zurueck.
24      * @return int Zahlensystem-Basis
25      */
26     function GetSystem();

28     /**
29      * Gibt das hoechstmoegliche Zeichen zurueck.
30      * @return char hoechstmoegliches Zeichen
31      */
32     function GetMaxSign();

34     /**
35      * Gibt das Zeichen der Basis fuer den Wert $x zurueck.
36      * @param int $x Wert x
37      * @return char Zeichen der Basis fuer den Wert x
38      */
39     function GetSign($x);
40 }
41 }

```

## B.2.2 Implementierung des Interfaces:

Listing B.3: PHP Implementierung der Zahlenbasis

```

1 namespace ahbsd\Zahlensysteme
2 {
3     /**
4      * Basis eines Zahlensystems.
5      *
6      * @author A. Hermann
7      * @copy Copyright &copy; 2016 Alexander Hermann - Beratung, Software, Design
8      * Zahlensysteme
9      *
10     * @version 1.0
11     */
12
13     class Base implements IBase
14     {
15         /**
16          * Konstante, die die ASCII (und UTF-8) Position von 'A' speichert.
17          * @var int
18          */
19         const A_POS_UTF8 = 65;

21         /**
22          * System-Name
23          * @var string
24          */
25         private $systemName;

27         /**
28          * System als Integer-Zahl. Maximale Anzahl an Zeichen.
29          *
30          * @var int
31          */
32         private $systemInt;

34         /**
35          * Hoechstmoegliches Zeichen.
36          *

```

```

37     * @var char
38     */
39     private $maxSign;

41     /**
42     * Konstruktor
43     *
44     * @param int $system Zahlensystem-Basis (Maximale Anzahl an Zeichen)
45     * @param string $name (Optional) Bezeichnung des Zahlensystems
46     */
47     public function __construct($system, $name="")
48     {
49         $this->systemInt=intval($system);
50         $this->systemName=$name;

52         if ($name=="")
53         {
54             $this->systemName = sprintf("Basis %1\$s", intval($system));
55         }
56         $tmp = A_POS_UTF8 - 11 + intval($system);

58         if ($system <= 10)
59         {
60             $this->maxSign = $system - 1;
61         }
62         else
63         {
64             $this->maxSign = mb_convert_encoding('&#' . $tmp . ';' , 'UTF-8', 'HTML-ENTITIES
65             ');
66         }

68     /**
69     * (non-PHPdoc)
70     * @see \ahbsd\Zahlensysteme\IBase::GetSign()
71     */
72     public function GetSign($x)
73     {
74         $tmp = 65-11+intval($x+1);
75         $result = $x;

77         if(intval($x) >= 10 || intval($x) < 0)
78         {
79             $result = mb_convert_encoding('&#' . $tmp . ';' , 'UTF-8', 'HTML-ENTITIES');
80         }

82         return $result;
83     }

85     /**
86     * (non-PHPdoc)
87     * @see \ahbsd\Zahlensysteme\IBase::GetName()
88     */
89     public function GetName()
90     {
91         return $this->systemName;
92     }

94     /**
95     * (non-PHPdoc)
96     * @see \ahbsd\Zahlensysteme\IBase::GetSystem()
97     */
98     public function GetSystem()
99     {
100         return $this->systemInt;

```

```

101     }

103     /**
104     * (non-PHPdoc)
105     * @see \ahbsd\Zahlensysteme\IBase::GetMaxSign()
106     */
107     public function GetMaxSign()
108     {
109         return $this->maxSign;
110     }

112     /**
113     * Statische Funktion zur Umwandlung einer Zahl aus dem Dezimalsystem in eine
114     * Zahl des Zahlensystems bX.
115     *
116     * @param int $b10 Umzuwandelnde Zahl aus dem Dezimalsystem.
117     * @param int $bX Zahlensystem in das b10 umgewandelt werden soll.
118     * @param bool $rechenweg (Optional) Gibt an, ob der Rechenweg angezeigt werden
119     * soll oder nicht; ohne Angabe standardmaessig FALSE.
120     * @return string Ergebnis in Basis bX
121     */
122     public static function Base10toBaseX($b10, $bX, $rechenweg=false)
123     {
124         $targetBase = new Base(intval($bX));
125         $result = array();
126         $restArray = array();
127         $rOut = "";

128         $quotient = intval($b10);
129         $rest = 0;
130         $cnt = 0;

132         if ($rechenweg)
133         {
134             echo "\n<!-- start Rechenweg --><pre>\n";
135             echo "Rechenweg:\n";
136         }

138         while (intval($quotient) != 0)
139         {
140             $rest = $quotient % $bX;
141             if ($rechenweg) echo "quotient : $bX = " . intval($quotient / $bX) . " Rest
142                 $rest [" . $targetBase->GetSign($rest) . "]\n";
143             $restArray[] = $rest;
144             $quotient = intval($quotient / $bX);
145         }
146         if ($rechenweg) echo "-----\n";
147         $cnt = count($restArray);

148         for($i=0;$i < $cnt; $i++)
149         {
150             $result[$cnt - ($i + 1)] = $targetBase->GetSign($restArray[$i]);
151         }

153         for($i=0; $i < $cnt; $i++)
154         {
155             $rOut .= $result[$i];
156         }

158         if ($rechenweg)
159         {
160             printf("Das Ergebnis der Umwandlung von %1\$s der Basis 10 in die %2\$s ist
161                 '%3\$s'\n", $b10, $targetBase->GetName(), $rOut);
162             echo "</pre><!-- ende Rechenweg -->\n\n";
163         }

```

```

163     return $rOut;
164 }

166 /**
167  *
168  * @param string $bxVal Der Wert der Basis $bX
169  * @param int $bX Die Quell Basis.
170  * @param bool $rechenweg Gibt an, ob der Rechenweg ausgegeben werden soll,
171  *     oder nicht.
172  * @return int Der Wert $bxVal umgerechnet in Basis 10.
173  */
174 public static function BaseXtoBase10($bxVal, $bX, $rechenweg=false)
175 {
176     $sourceBase = new Base(intval($bX));
177     $step = strlen($bxVal) - 1;
178     $result = 0;
179     $z = 0;
180     $curCarCorrect = false;
181     $intW = 0;

183     if ($rechenweg)
184     {
185         echo "\n<!-- start Rechenweg --><pre>\n";
186         echo "Rechenweg:\n";
187     }

189     while ($step >= 0) {
190         $charW = $bxVal[$step];
191         $tmpIntVal = intval($charW, 10);

193         $curCarCorrect = ('' . $tmpIntVal . ''==$charW);

195         if ($rechenweg) printf("%3\$s) Zeichen '%1\$s' an Stelle %2\$s ", $charW,
            $step, $z+1);

197         if ($curCarCorrect) {
198             $intW = $tmpIntVal;
199         }
200         else {
201             // CharWert Umrechnung
202             $tmp2 = ord($charW);

204             $intW = intval($tmp2) - 65 + 10; // A_POS_UTF8;

206             if ($rechenweg) printf("= (int) %1\$s", $intW);
207         }

209         if ($rechenweg) echo "\n";

211         $tmpR = 1;

213         for ($i = 0; $i < $z; $i++) {
214             $tmpR = $tmpR * $bX;
215         }

217         if ($rechenweg) printf("%1\$s^%2\$s=%3\$s\n%3\$s * %4\$s = ", $bX, $z, $tmpR,
            $intW);

219         $tmpR = $tmpR * $intW;

221         if ($rechenweg) echo $tmpR . "\n\n";

223         $step--;
224         $result += $tmpR;
225         $z++;

```

```

226     }

228     if ($rechenweg)
229     {
230         printf("Das Ergebnis der Umwandlung von '%1\$s' der %2\$s in die Basis 10 ist
                %3\$s\n", $bxVal, $sourceBase->GetName(), $result);
231         echo "</pre><!-- ende Rechenweg -->\n\n";
232     }

234     return $result;
235 }
236 }
237 }

```

## B.3 C#-Codierung

### B.3.1 Interface

Zuerst ein generalisiertes Interface, um die grundlegenden Eigenschaften und Methoden zu definieren.

Listing B.4: C-Sharp Interface der Zahlenbasis

```

1  using System;

3  namespace AHBSD.Zahlensysteme
4  {
5      public interface IBase
6      {
7          string Name { get; }
8          uint System { get; }
9          Char MaxSign { get; }
10         Char GetSign(uint number);
11         uint GetNumber(char sign);
12     }
13 }

```

### B.3.2 Exception

Dann eine spezialisierte Exception, falls hier irgendetwas grandios daneben geht<sup>2</sup>

Listing B.5: C-Sharp ZahlensystemException

```

1  using System;

3  namespace AHBSD.Zahlensysteme
4  {
5      public class ZahlensystemException : Exception
6      {
7          private uint system;
8          private IBase tryBase;

10         public ZahlensystemException(IBase tb)
11         : base(String.Format("Das kleinstmoegliche Zahlensystem ist 2; {0} ist zu
                klein!!", tb.System))
12         {
13             this.tryBase = tb;

```

---

<sup>2</sup>z.B.: wenn jemand versucht eine Zahlenbasis unter 2 anzulegen...



```

14         this.system = tb.System;
15     }

17     public ZahlensystemException(uint s)
18     : base(String.Format("Das kleinstmoegliche Zahlensystem ist 2; {0} ist zu
        klein!!", s))
19     {
20         this.tryBase = null;
21         this.system = s;
22     }

24     public ZahlensystemException(IBase tb, uint s)
25     : base(String.Format("{0} {1} ist ausserhalb der Basis {2}.", tb, s, tb.System
        ))
26     {
27         this.tryBase = tb;
28         this.system = tb.System;
29     }
30 }
31 }

```

### B.3.3 Implementierung des Interfaces

Listing B.6: C-Sharp Implementierung des Interfaces

```

1  using System;
2  using System.Text;
3  using System.Collections.Generic;

5  namespace AHBSD.Zahlensysteme
6  {
7      public class Base : IBase
8      {
9          private string name;
10         private uint system;
11         private Char maxSign;
12         protected const uint A_POS=65;
13         protected const uint ZERO_POS=48;

15         public Base()
16         {
17             this.system = 10;
18             this.SetName();
19             this.maxSign = this.GetSign(9);
20         }

22         public Base(uint System)
23         {
24             this.system = System;
25             this.SetName();
26             this.maxSign = this.GetSign(System-1);

28             if (System < 2)
29             {
30                 throw new ZahlensystemException(this);
31             }
32         }

34         public Base(uint System, string Name)
35         {
36             this.system = System;
37             this.name = "Basis " + Name;

```

```

38         this.maxSign = this.GetSign(System-1);

40         if (System < 2)
41         {
42             throw new ZahlensystemException(this);
43         }
44     }

46     protected void SetName()
47     {
48         this.name = "Basis " + this.system.ToString();
49     }

51     #region IBase Members
52     public string Name { get { return this.name; } }
53     public uint System { get { return this.system; } }
54     public char MaxSign { get { return this.maxSign; } }
55     public char GetSign(uint number)
56     {
57         char result;

59         if (number < this.system)
60         {
61             result = GetSignByNumber(number);
62         }
63         else
64         {
65             throw new ZahlensystemException(this, number);
66         }

68         return result;
69     }

71     public uint GetNumber(char sign)
72     {
73         uint result = GetNumberBySign(sign);

75         if (result >= this.system)
76         {
77             throw new ZahlensystemException(this, result);
78         }

80         return result;
81     }
82     #endregion

84     public static char GetSignByNumber(uint number)
85     {
86         char result = ' ';
87         uint tmpI;

89         if (number < 10)
90         {
91             tmpI = number + ZERO_POS;
92         }
93         else
94         {
95             tmpI = number - 10 + A_POS;
96         }
97         result = (char)tmpI;

99         return result;
100     }

102     public static uint GetNumberBySign(char sign)

```

```

103     {
104         uint result = 0;
105         uint tmp;

107         if (Char.IsDigit(sign))
108         {
109             result = uint.Parse(sign.ToString());
110         }
111         else
112         {
113             tmp = (uint)sign;
114             result = tmp - A_POS + 10;
115         }

117         return result;
118     }

120     public override string ToString()
121     {
122         StringBuilder result = new StringBuilder(this.name);
123         result.Append("; MaxSign: ");
124         result.Append(this.maxSign);

126         return result.ToString();
127     }

129     public static List<string> Base10toBaseX(ulong valB10, uint X, bool Rechenweg)
130     {
131         List<string> result;
132         string resultVal = String.Empty;
133         IBase targetBase = new Base(X);
134         ulong quotient = valB10;
135         ulong tmpQuotient;
136         uint rest = 0;
137         List<uint> restList = new List<uint>();
138         string[] fmt = new string[2];
139         string tmpFmt;

141         fmt[0] = "{0} : {1} = {2} Rest {3}";
142         fmt[1] = "{0} : {1} = {2} Rest {3} [{4}]";

144         if (!Rechenweg)
145         {
146             result = new List<string>(1);
147             result.Add(String.Empty);
148         }
149         else
150         {
151             result = new List<string>();
152             result.Add(String.Empty);
153             result.Add("Rechenweg:");
154         }

156         while (quotient > 0)
157         {
158             tmpQuotient = quotient;
159             rest = (uint)(quotient % X);
160             quotient = quotient / X;
161             restList.Add(rest);
162             if (Rechenweg)
163             {
164                 if (rest > 9)
165                 {
166                     tmpFmt = fmt[1];
167                 }

```

```

168         else
169         {
170             tmpFmt = fmt[0];
171         }
172         result.Add(String.Format(tmpFmt, tmpQuotient, X, quotient, rest,
173                                 targetBase.GetSign((uint)rest)));
174     }
175
176     resultVal = targetBase.GetSign((uint)rest).ToString() + resultVal;
177 }
178
179 if (Rechenweg) result.Add(String.Format("Das Ergebnis der Umwandlung von {0}
180     der Basis 10 in die Basis {1} ist '{2}'", valB10, X, resultVal));
181 result[0] = resultVal;
182
183 return result;
184 }
185
186 public static List<string> BaseXtoBase10(string valBX, uint X, bool Rechenweg)
187 {
188     List<string> result;
189     IBase sourceSystem = new Base(X);
190
191     int step = valBX.Length - 1;
192     char[] valBXC = valBX.ToCharArray();
193     char charW;
194     uint intW = 0;
195     uint tmp;
196     int zaehler = 0;
197     string[] fmt = new string[2];
198     string tmpFmt;
199
200     fmt[0] = "Wert an Position {0}: '{1}' = {2} int; {3}^{0} * {2} = {4} * {2} =
201         {5}";
202     fmt[1] = "Wert an Position {0}: {1}; {3}^{0} * {2} = {4} * {2} = {5}";
203
204     if (!Rechenweg)
205     {
206         result = new List<string>(1);
207         result.Add(String.Empty);
208     }
209     else
210     {
211         result = new List<string>();
212         result.Add(String.Empty);
213         result.Add("Rechenweg:");
214     }
215
216     while (step >= 0)
217     {
218         charW = valBXC[step];
219
220         tmp = sourceSystem.GetNumber(charW);
221
222         if (Char.IsDigit(charW))
223         {
224             tmpFmt = fmt[1];
225         }
226         else
227         {
228             tmpFmt = fmt[0];
229         }
230
231         if (Rechenweg) result.Add(String.Format(tmpFmt, zaehler, charW, tmp, X, (
232             uint)Math.Pow(X, zaehler), tmp * (uint)Math.Pow(X, zaehler)));

```

```

229         intW += tmp * (uint)Math.Pow(X, zaehler);

231         zaehler++;
232         step--;
233     }
234     result[0] = intW.ToString();

236     if (Rechenweg) result.Add(String.Format("Das Ergebnis der Umwandlung von '{0}'
        der Basis {1} in die Basis 10 ist {2}", valBX, X, result[0]));
237     return result;
238 }

240 public static List<string> BaseXtoBaseY(string valX, uint X, uint Y, bool
    Rechenweg)
241 {
242     List<string> result;
243     uint tmp;
244     int l;
245     string tmpS;

247     if (X!=10)
248     {
249         result = Base.BaseXtoBase10(valX, X, Rechenweg);
250         tmp = uint.Parse(result[0]);
251         l = result.Count;
252     }
253     else
254     {
255         tmp = uint.Parse(valX);
256         l = 0;
257         result = new List<string>();
258     }

260     if (l > 0)
261     {
262         if (Y!=10)
263         {
264             result.AddRange(Base.Base10toBaseX(tmp, Y, Rechenweg));
265         }
266         else
267         {
268             result.Add(tmp.ToString());
269         }
270     }
271     else
272     {
273         if (Y!=10)
274         {
275             result = Base.Base10toBaseX(tmp, Y, Rechenweg);
276         }
277         else
278         {
279             result = new List<string>();
280             result.Add(tmp.ToString());
281         }
282     }
283     tmpS = String.Format("Das Ergebnis der Umwandlung von '{0}' der Basis {1} in
        die Basis {3} ist '{2}'", valX, X, result[l], Y);
284     if (Rechenweg) result.Add(tmpS);
285     result.Add(result[l]);
286     return result;
287 }
288 }
289 }

```



# Literaturverzeichnis

- [Brü15] BRÜNNER, ARNDT: *Umrechnung von Zahlensystemen*, 12 2015.
- [Stö99] STÖCKER, HORST: *Taschenbuch mathematischer Formeln und moderner Verfahren*. Wissenschaftlicher Verlag Harri Deutsch GmbH, Frankfurt am Main, 1999.

# Index

binär, 9  
Binärsystem, 9

dezimal, 9

Hardware, 9  
hexadezimal, 9

octal, 9

Software, 9

Zahlen, 9  
-basen, 9  
-system, 9  
-systeme, 9