Zahlenbasen Umwandlung – in Theorie und Praxis

Alexander Hermann

28. April 2016

Inhaltsverzeichnis

Listings

Abbildungsverzeichnis

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[1] ab.

Da wir Menschen es gewohnt sind im Dezimalsystem[2] 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:

$$b\mathrm{x}v$$

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:

[1] auf der Basis $b = 2$
[2] 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** `28xNOR70KRANK`
- **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[3]:

- **binär:** $101110_{(2)}$
- **ternär:** $211201_{(3)}$
- **quinär:** $402314_{(5)}$
- **octal:** $576302_{(8)}$
- **dezimal:** $964_{(10)}$
- **tridezimal** $5A9C0B3_{(13)}$
- **hexadezimal:** $AFFE09_{(16)}$
- **oktovigesimal** $NOR70KRANK_{(28)}$
- **hexatridezimal:** $GIRAFFE0Z6A_{(36)}$

[3] 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:

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

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[2] 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.[3] Im „normalen“, dezimalen Zahlensystem von 0 bis 9 ist die 10 bereits *zweistellig*.

Merke: *das erste Zeichen ist immer 0!*

$$x_s = b^{s-1} * w \tag{2.1}$$

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.

[1] Basis 36 bei `ASCII`; bei `UTF-8` auch größer

[2] von rechts nach links

[3] Deswegen ist 2 auch die kleinstmögliche Zahlenbasis, weil bei nur einem Zeichen kein Unterschied mehr möglich ist.

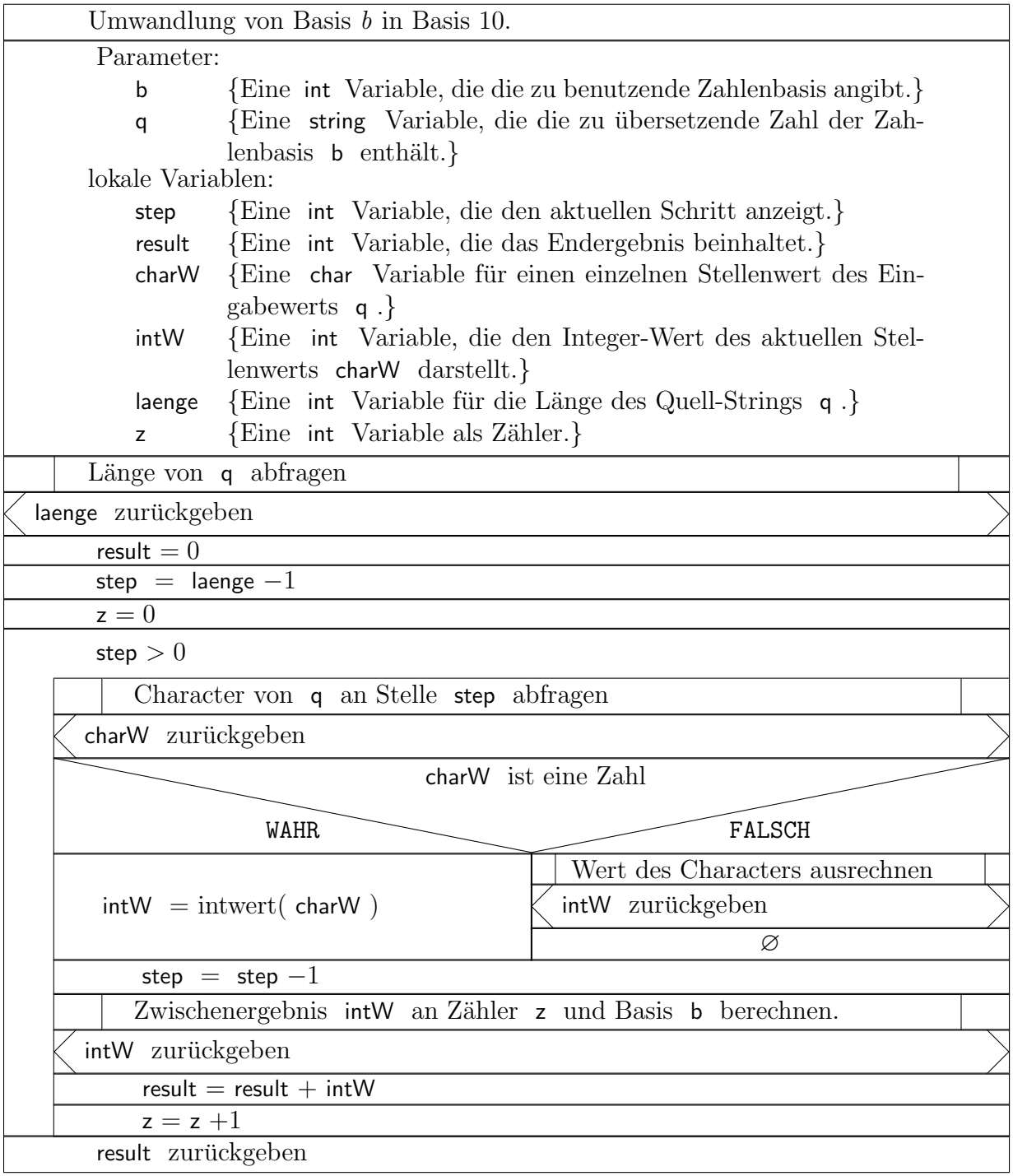

Abbildung 2.1: Struktogramm Umwandlung in das Dezimalsystem

Berechnung des Character-Werts
Parameter: c {Eine char Variable, die den auszuwertenden Character angibt.} lokale Variablen: result {Eine int Variable, die das Endergebnis beinhaltet.} z {Eine int Variable als Zwischenwert.}
Lese den ASCII oder UTF-8 Wert des Parameters c aus und schreibe es in das Zwischenergebnis z .
result = z + 9 – erste Buchstabenposition
result zurückgeben

Abbildung 2.2: Struktogramm Berechnung des Character-Werts

Berechnung des Zwischenergebnisses	
Parameter: b {Eine int Variable, die die zu benutzende Zahlenbasis angibt.} w {Eine int Variable, die den Eingabewert angibt.} p {Eine int Variable, die die zu benutzende Stellenposition enthält.} lokale Variablen: z {Eine int Variable als Zähler.} result {Eine int Variable für das Endergebnis}	
z = 0	
result = 1	
z < p	
	result = result ∗ b
	z = z +1
result = result ∗ w	
result zurückgeben	

Abbildung 2.3: Struktogramm Berechnung des Zwischenergebnisses

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

Eine der Anleitungen fand ich im Web[4]. 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.

$$\frac{s_n}{b} = q_n; r_n \tag{2.2}$$

2.2.2 Berechnungsablauf

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

$$s_n = q_{n-1} \tag{2.3}$$

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

$$g = r_0 * 10^0 + r_1 * 10^1 \cdots + r_n * 10^n \tag{2.4}$$

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

[4] www.arndt-bruenner.de [Brü15]

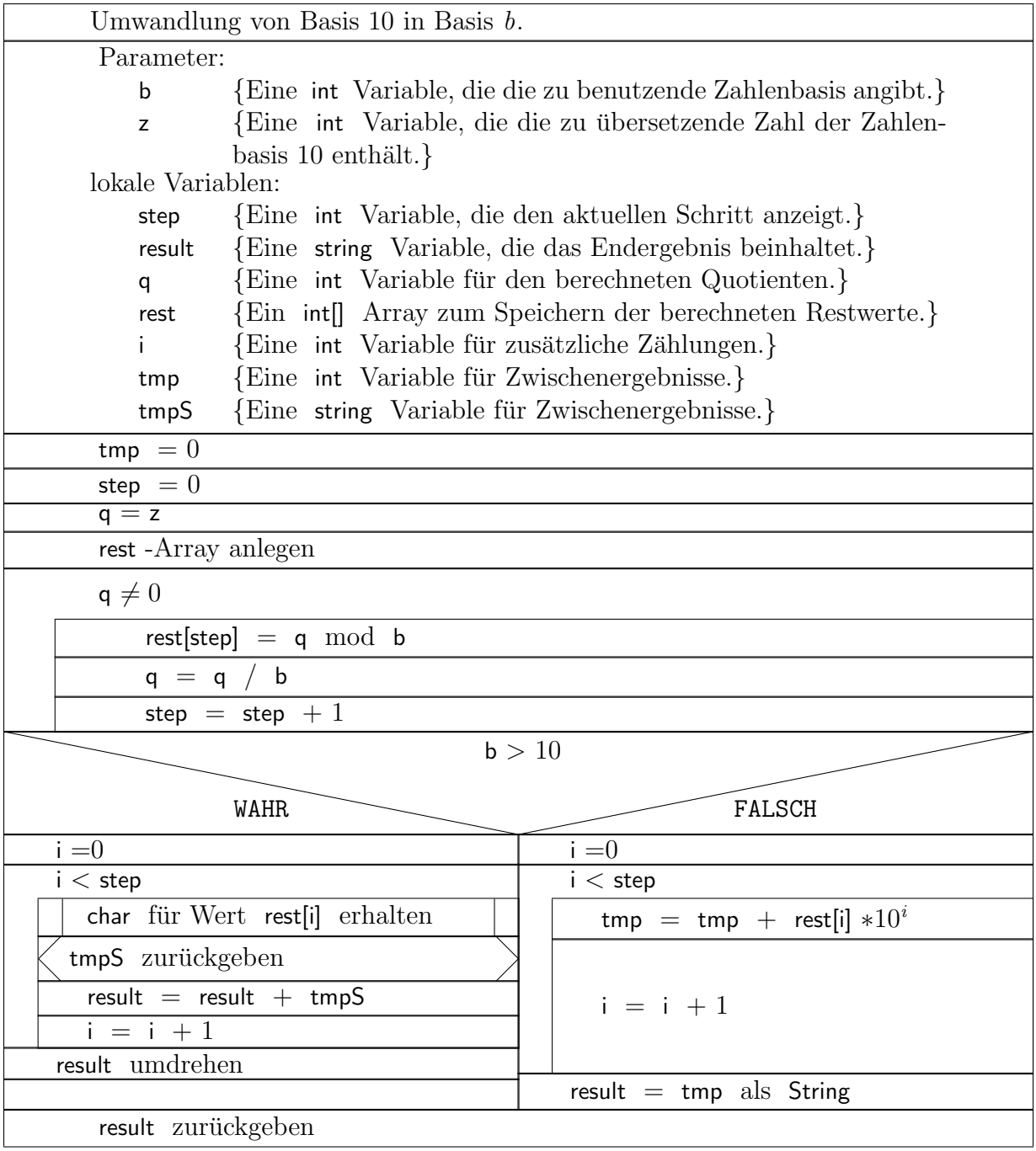

Abbildung 2.4: Struktogramm Umwandlung vom Dezimalsystem

Berechnung eines Character-Werts
Parameter: x {Eine int Variable, die den auszuwertenden wert angibt.} lokale Variablen: result {Eine char Variable, die das Endergebnis beinhaltet.} z {Eine int Variable als Zwischenwert.}
Lese den ASCII oder UTF-8 Wert des Buchstabens **A** aus und schreibe es in das Zwischenergebnis z .
result = x - 9 + z
result 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 $b = 2$:
$$g = r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2 + r_3 * 10^3 + r_4 * 10^4$$
$$+r_5 * 10^5 + r_6 * 10^6$$
$$g = 1 * 1 + 0 * 10 + 1 * 100 + 1 * 1000 + 0 * 10000$$
$$+0 * 100000 + 0 * 1000000 + 1 * 10000000$$
$$= 10001101$$

$10x141 = 2x10001101$

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 $b = 3$:
$$g = r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2$$
$$g = 1 * 1 + 1 * 10 + 1 * 100 = 111$$

$10x13 = 3x111$

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 $b = 3$:
$$g = r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2 + r_3 * 10^3 + r_4 * 10^4$$
$$g = 0 * 1 + 2 * 10 + 0 * 100 + 2 * 1000 + 1 * 10000$$
$$= 12020$$

$10x141 = 3x12020$

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 $b = 8$:
$$g = r_0 * 10^0 + r_1 * 10^1$$
$$g = 5 * 1 + 1 * 10 = 15$$

$10x13 = 8x15$

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 $b = 8$:
$$g = r_0 * 10^0 + r_1 * 10^1 + r_2 * 10^2$$
$$g = 5 * 1 + 1 * 10 + 2 * 100$$
$$= 215$$

$10x141 = 8x215$

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“[1]

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();
```

[1] 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     * Hoechstm   gliches 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 $rechenweg (Optional) Gibt an, ob der Rechenweg angezeigt werden
             soll oder nicht; ohne Angabe standardmaessig FALSE.
119      * @return string Ergebnis in Basis bX
120      */
121     public static function Base10toBaseX($b10, $bX, $rechenweg=false)
122     {
123       $targetBase = new Base(intval($bX));
124       $result = array();
125       $restArray = array();
126       $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
              $rest [" . $targetBase->GetSign($rest) . "]\n";
142         $restArray[] = $rest;
143         $quotient = intval($quotient / $bX);
144       }
145       if ($rechenweg) echo "--------------------\n";
146       $cnt = count($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
              '%3\$s'\n", $b10, $targetBase->GetName(), $rOut);
161         echo "</pre><!-- ende Rechenweg -->\n\n";
162       }
```

```
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 Zahlensysteme
4  {
5      /// <summary>
6      /// Interface einer Zahlenbasis: Grundlagen.
7      /// </summary>
8      public interface IBase
9      {
10         /// <summary>
11         /// Der Name der Zahlenbasis.
12         /// </summary>
13         /// <value>Der Name der Zahlenbasis.</value>
14         string Name { get; }
15         /// <summary>
16         /// Die Anzahl der moeglichen Zeichen.
17         /// </summary>
18         /// <value>Das Zahlensystem.</value>
19         uint System { get; }
20         /// <summary>
21         /// Das hoechstmoegliche Zeichen.
22         /// </summary>
23         /// <value>Die maximale Anzahl an Zeichen.</value>
24         Char MaxSign { get; }
25         /// <summary>
26         /// Gibt das entsprechende Zeichen fuer die uebergebene Zahl zurueck.
27         /// </summary>
28         /// <returns>Das Zeichen fuer die uebergebene Zahl.</returns>
29         /// <param name="number">Die zu uebersetzende Zahl.</param>
30         Char GetSign(uint number);
31         /// <summary>
32         /// Gibt die Zahl des uebergebenen Zeichens zurueck.
33         /// </summary>
34         /// <returns>Die Zahl fuer das uebergebene Zeichen.</returns>
35         /// <param name="sign">Das zu uebersetzende Zeichen.</param>
36         uint GetNumber(char sign);
37     }
38 }
```

B.3.2 Exception

Dann eine spezialisierte Exception, falls hier irgendetwas grandios daneben geht[2]

Listing B.5: C-Sharp ZahlensystemException

```
1  using System;
2
3  namespace Zahlensysteme
4  {
5     /// <summary>
6     /// Exception fuer Zahlensysteme.
7     /// </summary>
8     public class ZahlensystemException : Exception
9     {
10       /// <summary>
11       /// Das Zahlensystem als uint. Gleichzusetzen mit tryBase.System.
12       /// </summary>
13       private uint system;
14       /// <summary>
15       /// Die auszuprobierende Basis.
16       /// </summary>
17       private IBase tryBase;
18
19       /// <summary>
20       /// Konstruktor mit Angabe einer Zahlenbasis. Siehe <see cref="Zahlensysteme.
             ZahlensystemException"/>.
21       /// </summary>
22       /// <param name="tb">Test-Basis.</param>
23       public ZahlensystemException(IBase tb)
24          : base(String.Format("Das kleinstmoegliche Zahlensystem ist 2; {0} ist zu
               klein!!", tb.System))
25       {
26          this.tryBase = tb;
27          this.system = tb.System;
28       }
29
30       /// <summary>
31       /// Konstruktor mit Angabe einer Zahlenbasis. Siehe <see cref="Zahlensysteme.
             ZahlensystemException"/>.
32       /// </summary>
33       /// <param name="s">Zahlenbasis als Zahl.</param>
34       public ZahlensystemException(uint s)
35          : base(String.Format("Das kleinstmoegliche Zahlensystem ist 2; {0} ist zu
               klein!!", s))
36       {
37          this.tryBase = null;
38          this.system = s;
39       }
40
41       /// <summary>
42       /// Konstruktor mit Uebergabe der verwendeten Zahlenbasis, so wie einer falschen
             Zahl ausserhalb der Zahlenbasis.
43       /// Siehe <see cref="Zahlensysteme.ZahlensystemException"/>.
44       /// </summary>
45       /// <param name="tb">verwendeten Zahlenbasis.</param>
46       /// <param name="s">falschen Zahl.</param>
47       public ZahlensystemException(IBase tb, uint s)
48          : base(String.Format("{0} {1} ist ausserhalb der Basis {2}.", tb, s, UeUetb.
               System))
49       {
50          this.tryBase = tb;
```

[2] z.B.: wenn jemand versucht eine Zahlenbasis unter 2 anzulegen...

```
51            this.system = tb.System;
52        }

54    }
55 }
```

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;


6  namespace Zahlensysteme
7  {
8     /// <summary>
9     /// Zahlenbasis; implementiert <see cref="Zahlensysteme.IBase"/>.
10    /// </summary>
11    public class Base : IBase
12    {
13       /// <summary>
14       /// Der Name der Zahlenbasis.
15       /// </summary>
16       private string name;
17       /// <summary>
18       /// Die Zahlenbasis; die Anzahl der moeglichen Zeichen.
19       /// </summary>
20       private uint system;
21       /// <summary>
22       /// Das hoechstmoegliche Zeichen.
23       /// </summary>
24       private Char maxSign;
25       /// <summary>
26       /// Die Position von 'A'.
27       /// </summary>
28       protected const uint A_POS=65;
29       /// <summary>
30       /// Die Position von '0'.
31       /// </summary>
32       protected const uint ZERO_POS=48;

34       /// <summary>
35       /// Konstruktor ohne Parameter.
36       /// </summary>
37       public Base()
38       {
39          this.system = 10;
40          this.SetName();
41          this.maxSign = this.GetSign(9);
42       }

44       /// <summary>
45       /// Konstruktor mit Uebergabe einer Basis-Zahl.
46       /// </summary>
47       /// <param name="System">Basis-Zahl.</param>
48       public Base(uint System)
49       {
50          this.system = System;
51          this.SetName();
52          this.maxSign = this.GetSign(System-1);
```

```
54            if (System < 2)
55            {
56               throw new ZahlensystemException(this);
57            }
58         }

60         /// <summary>
61         /// Konstruktor mit Uebergabe einer Basis-Zahl und einem Namen der Zahlenbasis.
62         /// </summary>
63         /// <param name="System">Basis-Zahl.</param>
64         /// <param name="Name">Name der Zahlenbasis.</param>
65         public Base(uint System, string Name)
66         {
67            this.system = System;
68            this.name = "Basis " + Name;
69            this.maxSign = this.GetSign(System-1);

71            if (System < 2)
72            {
73               throw new ZahlensystemException(this);
74            }
75         }

77         /// <summary>
78         /// Setzt den Namen diese Zahlenbasis.
79         /// </summary>
80         protected void SetName()
81         {
82            this.name = "Basis " + this.system.ToString();
83         }

85         #region IBase Members
86         /// <summary>
87         /// Der Name der Zahlenbasis.
88         /// </summary>
89         /// <value>Der Name der Zahlenbasis.</value>
90         public string Name { get{ return this.name; } }

92         /// <summary>
93         /// Die Anzahl der moeglichen Zeichen.
94         /// </summary>
95         /// <value>Das Zahlensystem.</value>
96         public uint System { get { return this.system; } }

98         /// <summary>
99         /// Das hoechstmoegliche Zeichen.
100        /// </summary>
101        /// <value>Die maximale Anzahl an Zeichen.</value>
102        public char MaxSign { get { return this.maxSign; } }

104        /// <summary>
105        /// Gibt das entsprechende Zeichen fuer die uebergebene Zahl zurueck.
106        /// </summary>
107        /// <returns>Das Zeichen fuer die uebergebene Zahl.</returns>
108        /// <param name="number">Die zu uebersetzende Zahl.</param>
109        public char GetSign(uint number)
110        {
111           char result;

113           if (number < this.system)
114           {
115              result = GetSignByNumber(number);
116           }
117           else
```

```
118             {
119                 throw new ZahlensystemException(this, number);
120             }
121
122             return result;
123         }
124
125         /// <summary>
126         /// Gibt die Zahl des uebergebenen Zeichens zurueck.
127         /// </summary>
128         /// <returns>Die Zahl fuer das uebergebene Zeichen.</returns>
129         /// <param name="sign">Das zu uebersetzende Zeichen.</param>
130         public uint GetNumber(char sign)
131         {
132             uint result = GetNumberBySign(sign);
133
134             if (result >= this.system)
135             {
136                 throw new ZahlensystemException(this, result);
137             }
138
139             return result;
140         }
141         #endregion
142
143         /// <summary>
144         /// Gibt das entsprechende Zeichen fuer die uebergebene Zahl zurueck.
145         /// </summary>
146         /// <returns>Das Zeichen fuer die uebergebene Zahl.</returns>
147         /// <param name="number">Die zu uebersetzende Zahl.</param>
148         public static char GetSignByNumber(uint number)
149         {
150             char result = ' ';
151             uint tmpI;
152
153             if (number < 10)
154             {
155                 tmpI = number + ZERO_POS;
156             }
157             else
158             {
159                 tmpI = number - 10 + A_POS;
160             }
161             result = (char)tmpI;
162
163             return result;
164         }
165
166         /// <summary>
167         /// Gibt die Zahl des uebergebenen Zeichens zurueck.
168         /// </summary>
169         /// <returns>Die Zahl fuer das uebergebene Zeichen.</returns>
170         /// <param name="sign">Das zu uebersetzende Zeichen.</param>
171         public static uint GetNumberBySign(char sign)
172         {
173             uint result = 0;
174             uint tmp;
175
176             if (Char.IsDigit(sign))
177             {
178                 result = uint.Parse(sign.ToString());
179             }
180             else
181             {
182                 tmp = (uint)sign;
```

```
183             result = tmp - A_POS + 10;
184         }
185
186         return result;
187     }
188
189     /// <summary>
190     /// Gibt grundlegende Informationen ueber diese Zahlenbasis zurueck.
191     /// </summary>
192     /// <returns>A <see cref="System.String"/> that represents the current <see cref
            ="Zahlensysteme.Base"/>.</returns>
193     public override string ToString()
194     {
195         StringBuilder result = new StringBuilder(this.name);
196         result.Append("; MaxSign: ");
197         result.Append(this.maxSign);
198
199         return result.ToString();
200     }
201
202     /// <summary>
203     /// Statische Methode zur Umwandlung einer Zahl der Basis 10 zu der Basis X.
204     /// </summary>
205     /// <returns>Eine <see cref="System.Collections.Generic.List<String>"/>, die an
            erster Stelle das Ergebnis enthaelt.</returns>
206     /// <param name="valB10">Wert der Basis 10.</param>
207     /// <param name="X">Ziel-Basis.</param>
208     /// <param name="Rechenweg">Rechenweg mit <c>true</c> in die Ergebnis-Liste
            packen, oder nicht.</param>
209     public static List<string> Base10toBaseX(ulong valB10, uint X, bool Rechenweg)
210     {
211         List<string> result;
212         string resultVal = String.Empty;
213         IBase targetBase = new Base(X);
214         ulong quotient = valB10;
215         ulong tmpQuotient;
216         uint rest = 0;
217         List<uint> restList = new List<uint>();
218         string[] fmt = new string[2];
219         string tmpFmt;
220
221         fmt[0] = "{0} : {1} = {2} Rest {3}";
222         fmt[1] = "{0} : {1} = {2} Rest {3} [{4}]";
223
224         if (!Rechenweg)
225         {
226             result = new List<string>(1);
227             result.Add(String.Empty);
228         }
229         else
230         {
231             result = new List<string>();
232             result.Add(String.Empty);
233             result.Add("Rechenweg:");
234         }
235
236         while (quotient > 0)
237         {
238             tmpQuotient = quotient;
239             rest = (uint)(quotient % X);
240             quotient = quotient / X;
241             restList.Add(rest);
242             if (Rechenweg)
243             {
244                 if (rest > 9)
```

```
245                 {
246                     tmpFmt = fmt[1];
247                 }
248                 else
249                 {
250                     tmpFmt = fmt[0];
251                 }
252                 result.Add(String.Format(tmpFmt, tmpQuotient, X, quotient, rest,
                        targetBase.GetSign((uint)rest)));
253             }

255             resultVal = targetBase.GetSign((uint)rest).ToString() + resultVal;
256         }

258         if (Rechenweg) result.Add(String.Format("Das Ergebnis der Umwandlung von {0}
                der Basis 10 in die Basis {1} ist '{2}'", valB10, X, resultVal));
259         result[0] = resultVal;

261         return result;
262     }

264     /// <summary>
265     /// Statische Methode zur Umwandlung einer Zahl der Basis X zu der Basis 10.
266     /// </summary>
267     /// <returns>Eine <see cref="System.Collections.Generic.List<String>"/>, die an
            erster Stelle das Ergebnis enthaelt.</returns>
268     /// <param name="valBX">Wert der Basis X.</param>
269     /// <param name="X">Quell-Basis.</param>
270     /// <param name="Rechenweg">Rechenweg mit <c>true</c> in die Ergebnis-Liste
            packen, oder nicht.</param>
271     public static List<string> BaseXtoBase10(string valBX, uint X, bool Rechenweg)
272     {
273         List<string> result;
274         IBase sourceSystem = new Base(X);

276         int step = valBX.Length -1;
277         char[] valBXC = valBX.ToCharArray();
278         char charW;
279         uint intW = 0;
280         uint tmp;
281         int zaehler = 0;
282         string[] fmt = new string[2];
283         string tmpFmt;

285         fmt[0] = "Wert an Position {0}: '{1}' = {2} int; {3}^{0} * {2} = {4} * {2} =
                {5}";
286         fmt[1] = "Wert an Position {0}: {1}; {3}^{0} * {2} = {4} * {2} = {5}";

288         if (!Rechenweg)
289         {
290             result = new List<string>(1);
291             result.Add(String.Empty);
292         }
293         else
294         {
295             result = new List<string>();
296             result.Add(String.Empty);
297             result.Add("Rechenweg:");
298         }

300         while (step >= 0)
301         {
302             charW = valBXC[step];

304             tmp = sourceSystem.GetNumber(charW);
```

```
306             if (Char.IsDigit(charW))
307             {
308                 tmpFmt = fmt[1];
309             }
310             else
311             {
312                 tmpFmt = fmt[0];
313             }
315             if (Rechenweg) result.Add(String.Format(tmpFmt, zaehler, charW, tmp, X, (
                    uint)Math.Pow(X, zaehler), tmp * (uint)Math.Pow(X, zaehler)));
316             intW += tmp * (uint)Math.Pow(X, zaehler);
318             zaehler++;
319             step--;
320         }
321         result[0] = intW.ToString();
323         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]));
324         return result;
325     }
327     /// <summary>
328     /// Statische Methode zur Umwandlung einer Zahl der Basis X zu der Basis Y.
329     /// </summary>
330     /// <returns>Eine <see cref="System.Collections.Generic.List<String>"/>, die an
            erster Stelle das Ergebnis enthaelt.</returns>
331     /// <param name="valX">Wert der Basis X.</param>
332     /// <param name="X">Quell-Basis.</param>
333     /// <param name="Y">Ziel-Basis</param>
334     /// <param name="Rechenweg">Rechenweg mit <c>true</c> in die Ergebnis-Liste
            packen, oder nicht.</param>
335     public static List<string> BaseXtoBaseY(string valX, uint X, uint Y, bool
            Rechenweg)
336     {
337         List<string> result;
338         uint tmp;
339         int l;
340         string tmpS;
342         if (X!=10)
343         {
344             result = Base.BaseXtoBase10(valX, X, Rechenweg);
345             tmp = uint.Parse(result[0]);
346             l = result.Count;
347         }
348         else
349         {
350             tmp = uint.Parse(valX);
351             l = 0;
352             result = new List<string>();
353         }
355         if (l > 0)
356         {
357             if (Y!=10)
358             {
359                 result.AddRange(Base.Base10toBaseX(tmp, Y, Rechenweg));
360             }
361             else
362             {
363                 result.Add(tmp.ToString());
364             }
```

```
365             }
366             else
367             {
368                 if (Y!=10)
369                 {
370                     result = Base.Base10toBaseX(tmp, Y, Rechenweg);
371                 }
372                 else
373                 {
374                     result = new List<string>();
375                     result.Add(tmp.ToString());
376                 }
377             }
378             tmpS = String.Format("Das Ergebnis der Umwandlung von '{0}' der Basis {1} in
                    die Basis {3} ist '{2}'", valX, X, result[l], Y);
379             if (Rechenweg) result.Add(tmpS);
380             result.Add(result[l]);
381             return result;
382         }
383     }
384 }
```

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