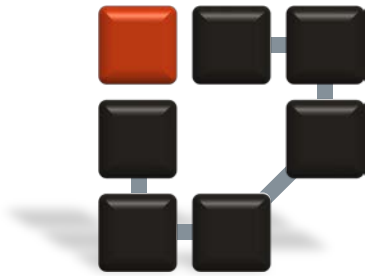
Informatik 1 für Nebenfachstudierende Grundmodul

Datendarstellung – Teil 1

Kai-Steffen Hielscher
Folienversion: 18. Oktober 2017



Informatik 7
Rechnernetze und
Kommunikationssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Inhaltsübersicht

- Kapitel 1 - Einführung und Übersicht
  - Was ist Informatik?
  - Grundbegriffe
  - **Datendarstellung**
  - Hardware von Computersystemen
  - Grundsoftware üblicher Computersysteme

Textdarstellung

- **Zur Erinnerung:** Die Repräsentanten von Informationen auf der Ebene der Nullen und Einsen nennen wir Daten.
- Im folgenden behandeln wir die Darstellung (Codierung) von Texten, logischen Werten, Zahlen und Programmen als Daten.
- Um Texte in einem Rechner darzustellen, codiert man Alphabet und Satzzeichen in Bitfolgen:
  - Bei einem Alphabet von 26 **Kleinbuchstaben**, 26 **Großbuchstaben**, **Satzzeichen** wie ".", ",", ";" und **Sonderzeichen** wie "+", "&" oder "%" besitzt eine Computertastatur (engl. keyboard) ca. 100 darstellbare Zeichen.
  - **Steuerinformationen**, wie z.B. **Zeilenumbruch** oder **Texteinrückungen** codiert man ebenfalls mit einem nichtdruckbaren "CR"- (für engl. carriage return = Wagenrücklauf) bzw. "TAB"-Zeichen (für Tabulator) .
  - Für die Darstellung aller Zeichen sind somit 7 Bit mit $2^7=128$ verschiedenen Möglichkeiten ausreichend.
  - Dabei ordnet man in einer Tabelle jedem Zeichen einen entsprechenden 7 -Bit-Code zu.

Der ASCII-Code

- Die meist verwendete **ASCII-Codierung** (für American Standard Code for Information Interchange) berücksichtigt zusätzliche Systematiken, wie:
  - die Kleinbuchstaben sind in der alphabetischen Reihenfolge durchnumeriert,
  - die Großbuchstaben sind in der alphabetischen Reihenfolge durchnumeriert,
  - die Ziffern 0 bis 9 stehen in der natürlichen Ordnungsrelation.
- Beispiele:

Zeichen	ASCII
a	97
A	65
b	98
B	66
z	122
Z	90
?	63

Zeichen	ASCII
0	48
1	49
9	57
CR	13
+	43
-	45
=	61

Der ASCII-Code

- Viele Rechensysteme, wie z.B. IBM-kompatiblen Personal-Computer, verwenden einen **Erweiterten ASCII-Code** mit zusätzlichen 128 Zeichen:
  - Somit sind 8 Bits zur Darstellung eines Zeichens notwendig.
  - Die zusätzlichen Codierungen verwendet man für **sprachspezifische Sonderzeichen**, z.B. deutsche Umlaute "ä", "ö", "ü" und "ß", **mathematische Sonderzeichen** oder Zeichen, mit denen man einfache grafische Darstellungen wie Rahmen und Schraffuren zusammensetzen kann.

Der ASCII-Code

- Erweiterter ASCII-Code
  - ANSI-Zeichentabelle
  - z.B. Windows PCs

0	■	32		64	@	96	`	128	■	160		192	À	224	à
1	■	33	!	65	A	97	a	129	■	161	¡	193	Á	225	á
2	■	34	"	66	B	98	b	130	‚	162	¢	194	Â	226	â
3	■	35	#	67	C	99	c	131	ƒ	163	£	195	Ã	227	ã
4	■	36	$	68	D	100	d	132	„	164	¤	196	Ä	228	ä
5	■	37	%	69	E	101	e	133	…	165	¥	197	Å	229	å
6	■	38	&	70	F	102	f	134	†	166	¦	198	Æ	230	æ
7	■	39	'	71	G	103	g	135	‡	167	§	199	Ç	231	ç
8	■	40	(	72	H	104	h	136	ˆ	168	¨	200	È	232	è
9	■	41	)	73	I	105	i	137	‰	169	©	201	É	233	é
10	■	42	*	74	J	106	j	138	Š	170	ª	202	Ê	234	ê
11	■	43	+	75	K	107	k	139	‹	171	«	203	Ë	235	ë
12	■	44	,	76	L	108	l	140	Œ	172	¬	204	Ì	236	ì
13	■	45	-	77	M	109	m	141	■	173	-	205	Í	237	í
14	■	46	.	78	N	110	n	142	■	174	®	206	Î	238	î
15	■	47	/	79	O	111	o	143	■	175	¯	207	Ï	239	ï
16	■	48	0	80	P	112	p	144	■	176	°	208	Ð	240	ð
17	■	49	1	81	Q	113	q	145	‘	177	±	209	Ñ	241	ñ
18	■	50	2	82	R	114	r	146	’	178	²	210	Ò	242	ò
19	■	51	3	83	S	115	s	147	“	179	³	211	Ó	243	ó
20	■	52	4	84	T	116	t	148	”	180	´	212	Ô	244	ô
21	■	53	5	85	U	117	u	149	•	181	µ	213	Õ	245	õ
22	■	54	6	86	V	118	v	150	–	182	¶	214	Ö	246	ö
23	■	55	7	87	W	119	w	151	—	183	·	215	×	247	÷
24	■	56	8	88	X	120	x	152	˜	184	¸	216	Ø	248	ø
25	■	57	9	89	Y	121	y	153	™	185	¹	217	Ù	249	ù
26	■	58	:	90	Z	122	z	154	š	186	º	218	Ú	250	ú
27	■	59	;	91	[	123	{	155	›	187	»	219	Û	251	û
28	■	60	<	92	\	124	|	156	œ	188	¼	220	Ü	252	ü
29	■	61	=	93	]	125	}	157	■	189	½	221	Ý	253	ý
30	■	62	>	94	^	126	~	158	■	190	¾	222	Þ	254	þ
31	■	63	?	95	_	127	■	159	Ÿ	191	¿	223	ß	255	ÿ

■ Indicates that this character isn't supported by Windows.

™ Indicates that this character is available only in TrueType fonts.

Der ASCII-Code

- In Europa wird meist die ASCII-Erweiterung **Latin-1** verwendet, die durch die Norm **ISO 8859-1** (ISO: International Standardization Organisation) standardisiert ist.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80																
90																
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Der ASCII-Code

- ISO 8859-15 ist aktueller als ISO 8859-1
  - Eurozeichen
  - Sonderzeichen
    - französische,
    - estnische und
    - finnische Sprache

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80																
90																
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	€ 20AC	¥ 00A5	Š 0160	§ 00A7	š 0161	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	Ž 017D	µ 00B5	¶ 00B6	· 00B7	ž 017E	¹ 00B9	º 00BA	» 00BB	Œ 0152	œ 0153	Ÿ 0178	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Der Unicode

- Aktuelle Betriebssysteme und moderne Programmiersprachen wie Java bauen auf dem neueren Zeichensatz **Unicode** auf, der praktisch alle weltweit geläufigen Zeichensätze vereinigt.
  - Der Unicode verwendet für jedes Zeichen einen Codepunkt aus 17 Ebenen zu je 2^{16} = 65536 Zeichencodierungen. Somit ergeben sich insgesamt 1114112 unterschiedliche mögliche Zeichen. Dies ermöglicht die Codierung aller länderspezifischen Sonderzeichen, wie z.B. japanische, tibetische oder chinesische Schriftzeichen.
  - Interne Darstellung im Rechner im **Unicode Transformation Format** (**UTF**)
    - UTF-32 (32 Bits pro Zeichen)
    - UTF-16 (1 oder 2 16-Bit-Einheiten)
    - UTF-8 (variable Zeichenlänge, 1, 2, 3 oder 4 Bytes pro Zeichen)
  - Weitere Informationen über Unicode findet man unter http://www.unicode.org.

Der Unicode

- z.B. Balinesisch

Logische Werte

- Logische Aussagen besitzen die **Wahrheitswerte** "**wahr**" oder "**falsch**".
- Prinzipiell kann man diese Werte mit Hilfe von genau einem Bit darstellen.
- Wie im letzten Absatz ausgeführt, ist ein Byte aber die kleinste Gruppe von Bits, die ein Rechner verarbeitet.
- Daher wird in den meisten Computersystemen auch für die Darstellung von Wahrheitswerten ein ganzes Byte verwendet.
- Häufig findet man die englischen Bezeichnungen:
  - **logische Wahrheitswerte:**

0	falsch	false
1	wahr	true

  - **logische Operatoren:**

not	Negation
and	Und-Verknüpfung
or	Oder-Verknüpfung
xor	Exklusives Oder (entweder – oder)

Logische Operatoren

- 1-stellige **Negation**:
  Konvertiert logischen Wert von 0 nach 1 und umgekehrt
- **Und-Verknüpfung** zweier logischer Werte
  - genau dann 1, wenn beide Werte 1 sind
- **Oder-Verknüpfung** zweier logischer Werte
  - genau dann 0, wenn beide Werte 0 sind
- **Exklusive-Oder-Verknüpfung** zweier logischer Werte
  - genau dann 1, wenn beide Werte verschieden sind

NOT	
F	T
T	F

AND	F	T
F	F	F
T	F	T

	OR	Wert 2: F	Wert 2: T
Wert 1	**F**	F	T
	T	T	T

Ergebnis

XOR	F	T
F	F	T
T	T	F

Programme, Grafiken und Videos

- **Programme** sind Anweisungen, die einen Rechner veranlassen, bestimmte Dinge nach dem EVA-Prinzip auszuführen.
  - Zur Speicherung der **Anweisungsfolgen** im Rechner oder auf einem externen Medium benutzt man eine vorher festgelegte Codierung, die jedem **Befehl** eine bestimmte **Bitfolge** zuordnet.
  - Bei der Erstellung werden Programme als ASCII- oder Unicode-Text formuliert.
  - Ein **Compiler** übersetzt diesen Text in eine Reihe von Befehlen, die sogenannte **Maschinensprache**, die der Rechner „versteht“.
- Auch Komplexe **Multimediainformationen** wie Grafiken, Video- und Audioströme können als Daten in einem Computer verarbeitet und gespeichert werden.
  - Grafiken und Videos werden dabei in eine Folge von Rasterpunkten aufgelöst.
  - Bei ausreichender Vergrößerung (Lupenfunktion) sieht man, wie das Bild aus einzelnen Rasterpunkten zusammengesetzt ist.
  - Beispiele für standardisierte Codierungsverfahren sind **JPEG** für Grafiken und **MPEG** (**M**otion **P**icture **E**xpert **G**roup) für Audio- und Videodaten.

Zahlendarstellungen

- Dieser Abschnitt behandelt Zahlensysteme, Zahlendarstellungen und ihre Arithmetik.
  - Vor allem die **Gleitkommaarithmetik**, die im wissenschaftlich-technischen Bereich eine große Bedeutung hat, wird näher vorgestellt.
  - Gleitkommazahlen sind gut geeignet, um sehr große und sehr kleine numerische Werte darzustellen.
  - Zahlen stellt man durch eine Folge von Zeichen (die sogenannten Ziffern) dar.
  - Wir gehen im folgenden immer von der sogenannten **Radixschreibweise** aus, bei der der Beitrag einer Ziffer als Produkt aus dem Zahlenwert der Ziffer und der der Stelle der Ziffer entsprechenden Potenz der Basis B gebildet wird.
- Beispiele:
  - $(2017)_{10} = 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 7 \cdot 10^0$
  - $(101)_5 = 1 \cdot 5^2 + 0 \cdot 5^1 + 1 \cdot 5^0$
  - $(10110)_2 = 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^1$

Polyadische Zahlensysteme

- In einem **polyadischen** oder **B-adischen Zahlensystem** mit Basis $B \geq 2$ wird eine Zahl Z dargestellt durch

$$Z = \sum_{-\infty}^{n} b_i B^i$$

- Die b_i sind die Ziffern mit $0 \leq b_i < B$. Anstatt obiger Potenzschreibweise verwendet man normalerweise die Stellenschreibweise
  $Z = b_n\ b_{n-1}\ b_{n-2}\ \ldots\ b_2\ b_1\ b_0\ \mathbf{,}\ b_{-1}\ b_{-2}\ \ldots$
  (Beachte das Komma!)
- Für ganze Zahlen gilt $b_i = 0$ für $i < 0$.
- Bei gebrochenen Zahlen setzt man zwischen b_0 und b_{-1} ein Komma oder einen Punkt.

Polyadische Zahlensysteme

- Für die Informatik sind die wichtigsten Zahlensysteme das
  - **Dualsystem** (mit den Ziffern 0, 1)
  - **Dezimalsystem** (mit den Ziffern 0, 1, ..., 9)
  - **Oktalsystem** (mit den Ziffern 0, ..., 7)
  - Sedezimal- oder **Hexadezimalsystem** (mit den Ziffern 0, 1, ..., 9, A, B, ..., F)
- Ein B-adisches Zahlensystem zur Basis B benötigt B Ziffern 0, 1,..., B-1.

Polyadische Zahlensysteme

- Bisher haben wir nur ganze Zahlen betrachtet. Wie ist es aber für rationale und reelle Zahlen? Vom Dezimalsystem kennen wir folgende Aussagen, die auch für beliebige B-adische Zahlensysteme gelten:
  - **Rationale Zahlen** liefern einen abbrechenden oder einen periodischen Bruch.
  - **Reelle Zahlen** lassen sich durch einen unendlichen Bruch darstellen.
- **Beispiele:**
  - $7/5 = (1.4)_{10}$ $\quad 3/4 = (0.75)_{10}$
  - $7/5 = (1.01100110...)_2$ $\quad 3/4 = (0.11)_2$

Polyadische Zahlensysteme

- Wie am Beispiel von 7/5 zu erkennen ist, kann beim Übergang von einem Zahlensystem in ein anderes aus einem endlichen Bruch ein periodischer Bruch werden. Es gilt die folgende Regel:

  Ein abbrechender B-adischer Bruch entsteht aus dem echten Bruch p/q nur, wenn in q nur solche Primfaktoren enthalten sind, die auch in der Basis B vorkommen.

- Das heißt, dass jede endliche Dualzahl auch als endliche Dezimalzahl darstellbar ist. Die Umkehrung gilt nicht, wie das Beispiel 7/5 zeigt.

Polyadische Zahlensysteme

- Das **Dualsystem** hat die größte Bedeutung:
  - Formale Hilfsmittel sind zweiwertige Logik und Boolesche Algebra.
  - Rechnen im Dualsystem ist einfach, d.h. man erhält auch einfachere elektronische Schaltungen und Schaltwerke.
  - Die technische Realisierung von zweiwertigen Systeme ist meist nicht so komplex wie die von n-wertigen Systemen.
  - Es sind nur zwei Zustände eines Schalt- oder Speicherelementes zu unterscheiden: Spannung/keine Spannung, Magnetisierung +/-,...

Polyadische Zahlensysteme

- Als Konsequenz besitzen auch das Oktalsystem und das Hexadezimalsystem große Bedeutung. Die Umwandlung von Dual- in Oktal- oder Hexadezimalzahlen und umgekehrt ist besonders einfach.
- **Beispiele:**
  - (Oktalzahl → Dualzahl): jede Oktalziffer wird durch drei Dualziffern ersetzt:
    $(523)_8 = (101\ 010\ 011)_2$
  - (Dualzahl → Oktalzahl): drei Dualziffern werden durch eine Oktalziffer ersetzt:
    $(10111.01)_2 = (010\ 111.010)_2 = (27.2)_8$

Polyadische Zahlensysteme

- Für die Umwandlung zwischen Dual- und Hexadezimalsystem sind analog jeweils vier Dualziffern in eine Hexadezimalziffer umzuwandeln.
- Wir haben gesehen, dass die Konvertierung zwischen Dual-, Oktal- und Hexadezimalsystem sehr einfach ist. Da 2 der einzige Primfaktor für die Basen B = 2, 8, 16 ist, bleiben endliche B-adische Brüche endlich und periodische Brüche bleiben periodisch.
- **ACHTUNG!**
  Schwieriger ist der allgemeine Fall.

Zahlenkonvertierung und Konvertierungsfehler

- Eine Zahl Z von einer Darstellung im B-adischen System in eine C-adische Darstellung zu konvertieren bedeutet, Koeffizienten $0 \leq c_c < C$ zu finden, so dass gilt

$$Z = \sum_{-\infty}^{n} b_i B^i = \sum_{-\infty}^{m} c_i C^i$$

- Da wir bei Realisierung auf einem Rechner nur endlich viele Stellen zur Speicherung einer Zahl zur Verfügung haben, können wir uns auf die Betrachtung abbrechender B-adischer Brüche beschränken:

$$Z = \sum_{-k}^{n} b_i B^i = \sum_{-l}^{m} c_i C^i$$

Zahlenkonvertierung und Konvertierungsfehler

- Für ganze Zahlen (d.h.: k=l=0) ist es immer möglich, diese Beziehung zu erfüllen. Die Konvertierung einer endlichen gebrochenen Zahl muss dagegen nicht notwendig wieder zu einer endlich-stelligen Zahl führen. Es kann wegen der begrenzten Stellenzahl zu einem **Konvertierungsfehler** kommen.

Zahlenkonvertierung und Konvertierungsfehler

- Für die wichtige Umwandlung vom Dezimal- ins Dualsystem und zurück bedeutet dies folgendes:
  - Oft wird ein Dezimalbruch im Dualsystem nur approximativ, d.h. mit einem gewissen Fehler, darstellbar sein:

$$\sum_{-k}^{n} b_i 10^i \approx \sum_{-l}^{m} c_i 2^i$$

  - Ein Dualbruch kann immer exakt in einen Dezimalbruch konvertiert werden (Grund: 2 ist Primfaktor von 10).
- Man muss also schon vor Beginn einer numerischen Berechnung unter Umständen einen Fehler in Kauf nehmen!
- Wie man Dezimalzahlen in Dualzahlen umwandelt, ist im folgenden exemplarisch für ganze Zahlen und echt gebrochene Zahlen dargestellt. Gebrochene Zahlen kann man konvertieren, indem man den ganzen und den gebrochenen Anteil jeweils separat konvertiert.

Sukzessive Division mit Rest

- Das folgende Verfahren heißt „**sukzessive Division mit Rest**“ und basiert auf dem **Hornerschema**, das von der Darstellung von Polynomen her bekannt ist.
- Umwandlung einer ganzen Dezimalzahl Z in eine Dualzahl: Beobachtung:

$$Z = c_n \cdot 2^n + c_{n-1} \cdot 2^{n-1} + \ \ldots + c_1 \cdot 2^1 + c_0$$
$$= 2 \cdot (c_n \cdot 2^{n-1} + c_{n-1} \cdot 2^{n-2} + \ \ldots + c_1) + c_0$$

$$Z : 2 = q_o \ \mathbf{Rest}\ c_0$$
$$q_0 : 2 = q_1 \ \mathbf{Rest}\ c_1$$
$$q_1 : 2 = q_2 \ \mathbf{Rest}\ c_2$$
$$\ldots$$
$$q_{n-1} : 2 = 0 \ \mathbf{Rest}\ c_n$$

Stelle niedrigster Wertigkeit entsteht zuerst

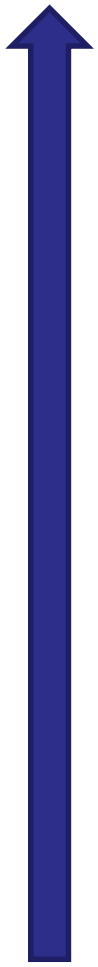
Ergebnis: $Z = (c_n\ c_{n-1}\ \ldots\ c_1\ c_0)_2$

gebräuchliche Stellenschreibweise

Sukzessive Division mit Rest

- **Beispiel:** $(4711)_{10} \rightarrow (1001001100111)_2$

4711	div 2 =	2355	Rest → 1
2355	div 2 =	1177	Rest → 1
1177	div 2 =	588	Rest → 1
588	div 2 =	294	Rest → 0
294	div 2 =	147	Rest → 0
147	div 2 =	73	Rest → 1
73	div 2 =	36	Rest → 1
36	div 2 =	18	Rest → 0
18	div 2 =	9	Rest → 0
9	div 2 =	4	Rest → 1
4	div 2 =	2	Rest → 0
2	div 2 =	1	Rest → 0
1	div 2 =	0	Rest → 1



Sukzessive Multiplikation mit Überlauf

- Umwandlung einer echt gebrochenen Dezimalzahl Z in eine Dualzahl:

$$Z \cdot 2 = q_0 \text{ Überlauf } c_{-1}$$
$$q_0 \cdot 2 = q_1 \text{ Überlauf } c_{-2}$$
$$q_1 \cdot 2 = q_2 \text{ Überlauf } c_{-3}$$
$$\ldots$$

- Falls q_i irgendwann den Wert 0 annimmt, haben wir einen endlichen Dualbruch erhalten, andernfalls müssen wir nach einer gewissen Zahl von Schritten (entsprechend der Stellenzahl) abbrechen und uns mit einer **Approximation** begnügen.
- **Beispiel:** Konvertierung von $(0.2)_{10}$ in das Dualsystem
  0.2 * 2 = 0.4 Überlauf 0
  0.4 * 2 = 0.8 Überlauf 0
  0.8 * 2 = 1.6 Überlauf 1
  0.6 * 2 = 1.2 Überlauf 1
  0.2 * 2 = 0.4 Periode !!
- $(0.2)_{10} = (0.00110011\ldots)_2$

Sukzessive Multiplikation mit Überlauf

- Nehmen wir an, wir hätten nur acht Bits zur Speicherung der Nachkommastellen zur Verfügung, so ergebe sich aufgrund der Rechnung $(0.2)_{10} \approx (0.00110011)_2$.
- Die Probe durch Re-Konvertierung ins Dezimalsystem liefert aber

  $(0.2)_{10} \approx (0.00110011)_2 =$

  $1/8 + 1/16 + 1/128 + 1/256 =$

  $(32 + 16 + 2 + 1)/256 =$

  $51/256 = 0.1992187...$

  ⇨ $0.1992187... < 0.2$
- Die Differenz zwischen diesen Werten heißt **Konvertierungsfehler.**
- **Konvertierungsfehler:**
  Differenz zwischen korrektem Wert und Approximation

Zahlendarstellung im Rechner

- Eine Folge von N Bits, die als Ganzes adressierbar und manipulierbar sind, bezeichnet man als **Wort**, die Zahl N heißt **Wortlänge**.
- Lange Zeit betrug die typische Wortlänge 16 Bit, im PC-Bereich ist eine Wortlänge von 32 oder 64 Bit inzwischen Standard.
- Zwischen der Wortlänge eines Rechners und der Zahlendarstellung besteht ein unmittelbarer Zusammenhang, da i.A. für die Darstellung einer Zahl ein Wort verwendet wird (manchmal auch ein Halbwort oder ein Doppelwort).
  - der darstellbare Zahlenbereich ist beschränkt und es können nur rationale Zahlen - und davon auch nur eine Teilmenge - dargestellt werden.
  - bei einer Wortlänge von 16 Bit können im Dualsystem nur die ganzen Zahlen von 0 bis 2^{16}-1 (0 ... 65535) dargestellt werden, im Dezimalsystem, unter Verwendung der **Tetradenschreibweise**, bei der jede Dezimalziffer durch 4 Bit dargestellt werden, sogar nur 0 bis 9999
- Um negative und gebrochene Zahlen darzustellen, müssen auch Vorzeichen und Komma berücksichtigt werden.

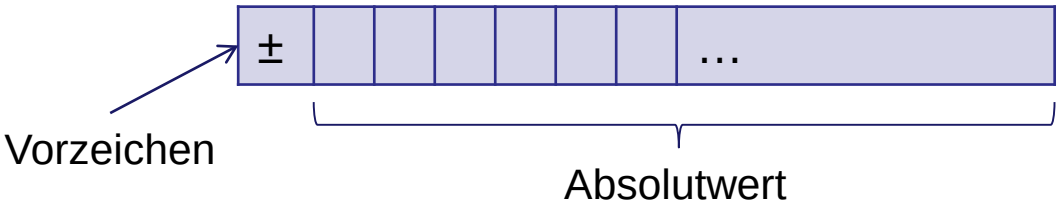
Zahlendarstellung im Rechner

- Hinsichtlich der Kommabehandlung unterscheidet man zwischen der **Festkomma-** und der **Gleitkommadarstellung**.
  - Bei der Festkommadarstellung wird das Komma nicht explizit dargestellt, es steht immer an der gleichen Stelle.
  - Durch einen **Maßstabsfaktor** MF wird das als ganze Zahl aufgefasste Wort in die darzustellende Zahl transformiert.
- **Beispiele:** (Wortlänge 20 Bit = 5 Tetraden à 4 Bit)
  - Z = 732510 7 3 2 5 1 MF = 10^{1}
  - Z = 4.37 0 0 4 3 7 MF = 10^{-2}
- Die **Vorzeichendarstellung** kann durch ein Vorzeichenbit gelöst werden, das an den Anfang oder das Ende eines Worts gestellt ist.
  - Eine 0 bedeutet z.B. ein positives, eine 1 ein negatives Vorzeichen.
  - Bei der Tetradendarstellung wird dementsprechend eine Tetrade (= 4 Bit) für das Vorzeichen reserviert.
- Um diese Sonderbehandlung des Vorzeichens zu umgehen, verwendet man zur Zahldarstellung oft die sog. **Komplementdarstellung (nächster Foliensatz)**.

Zahlendarstellung im Rechner

Darstellung ganzer Zahlen

- Ganze Zahlen ::= Natürliche Zahlen + Negative (‘nat.‘) Zahlen
  { ... -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, ... }
- Kodierung des absoluten Zahlenwertes und Vorzeichen´+´ bzw. ´-´, d.h. für diese 2 Möglichkeiten genau 1 weiteres Bit Information nötig
- sogen. **Vorzeichendarstellung**:
  - Erstes Bit für Vorzeichen: **0** für "**+**" und **1** für "**-**"
  - weitere Bits für den Absolutwert
- Beispiel:
  - Wortlänge sei 4 Bit
  - Ein Bit für Vorzeichen
  - Drei Bit für den Absolutwert
  - Zahlen von -7 bis +7 darstellbar



Z = ± | Absolutwert |