



# 4. Daten

# Was ist Informatik?

Begriffsbestimmung (Gegenstand):

vgl. Kapitel 1

"Informatik ist die Wissenschaft ...  
der maschinellen Verarbeitung,  
Speicherung und Übertragung  
von Information."

*(Broy, Informatik, Teil I, Springer 1992).*

- Information verarbeiten (Algorithmen)
- Information repräsentieren (Daten ... Datenbanken)
- Informationen übertragen (Internet ... Kontoauszugsdrucker)

Programme = Daten + Algorithmen

# Programme lösen Probleme

... durch Algorithmen, die über Daten operieren

Programme = Daten + Algorithmen

# Problembereiche zu Daten

Programme = Daten + Algorithmen

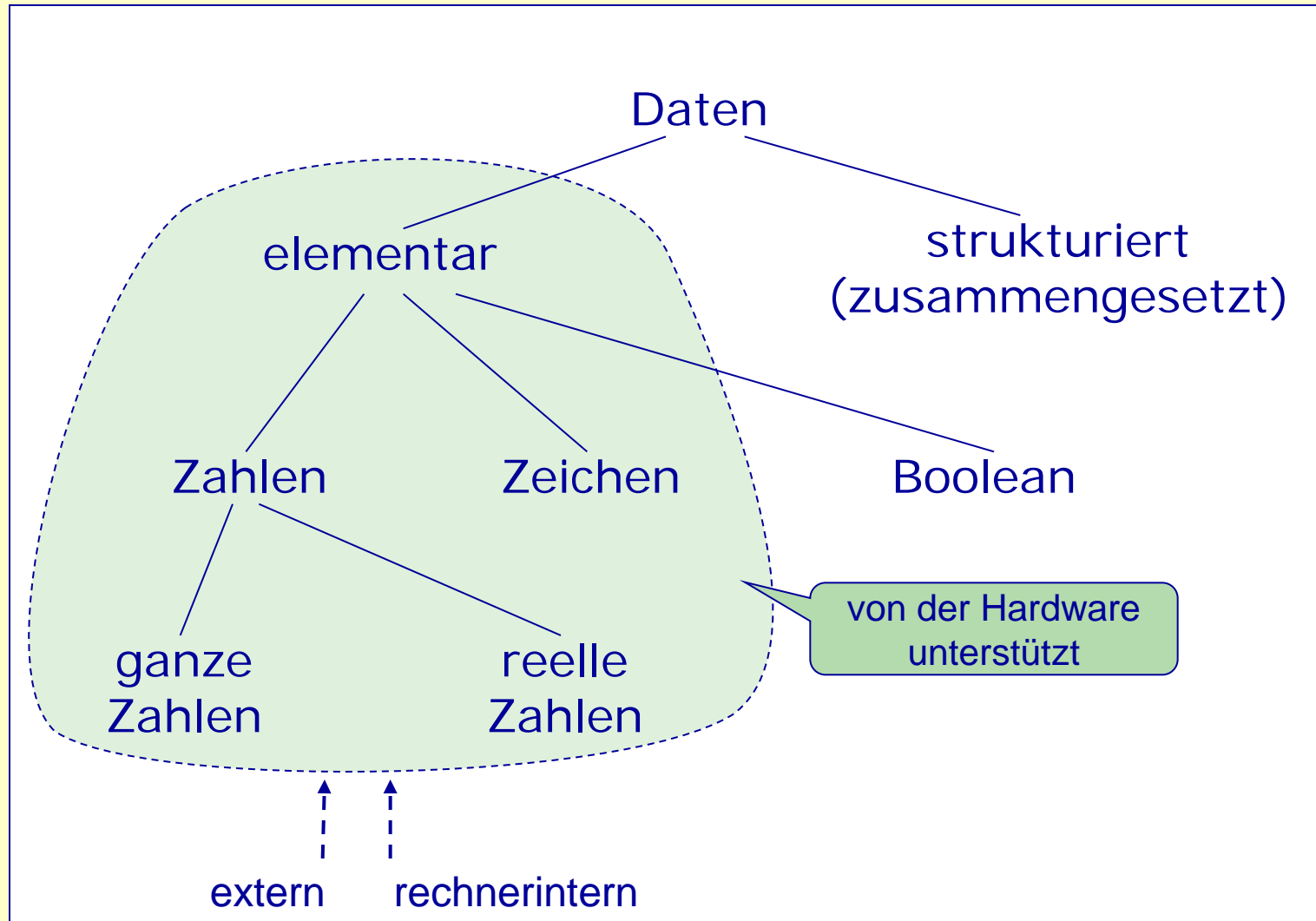
## Problembereiche:

- **Repräsentationen** (Darstellung):
  - extern (menschengerechte, lesbare Form, im Programm)
  - ↔ rechnerintern (Bitfolgen: 00110 ...)
- **Strukturierungsart:**
  - elementare Daten (Zahlen, Zeichen)
  - ↔ zusammengesetzte (strukturierte) Daten

jetzt: *elementare Daten in externer und rechnerinterner Repräsentation*

später (Teil II): *elementare und strukturierte Daten in Java*

# Klassifikation von Daten



# Rechnerinterne Darstellung von Daten bei Nutzung höherer Programmiersprachen wichtig?

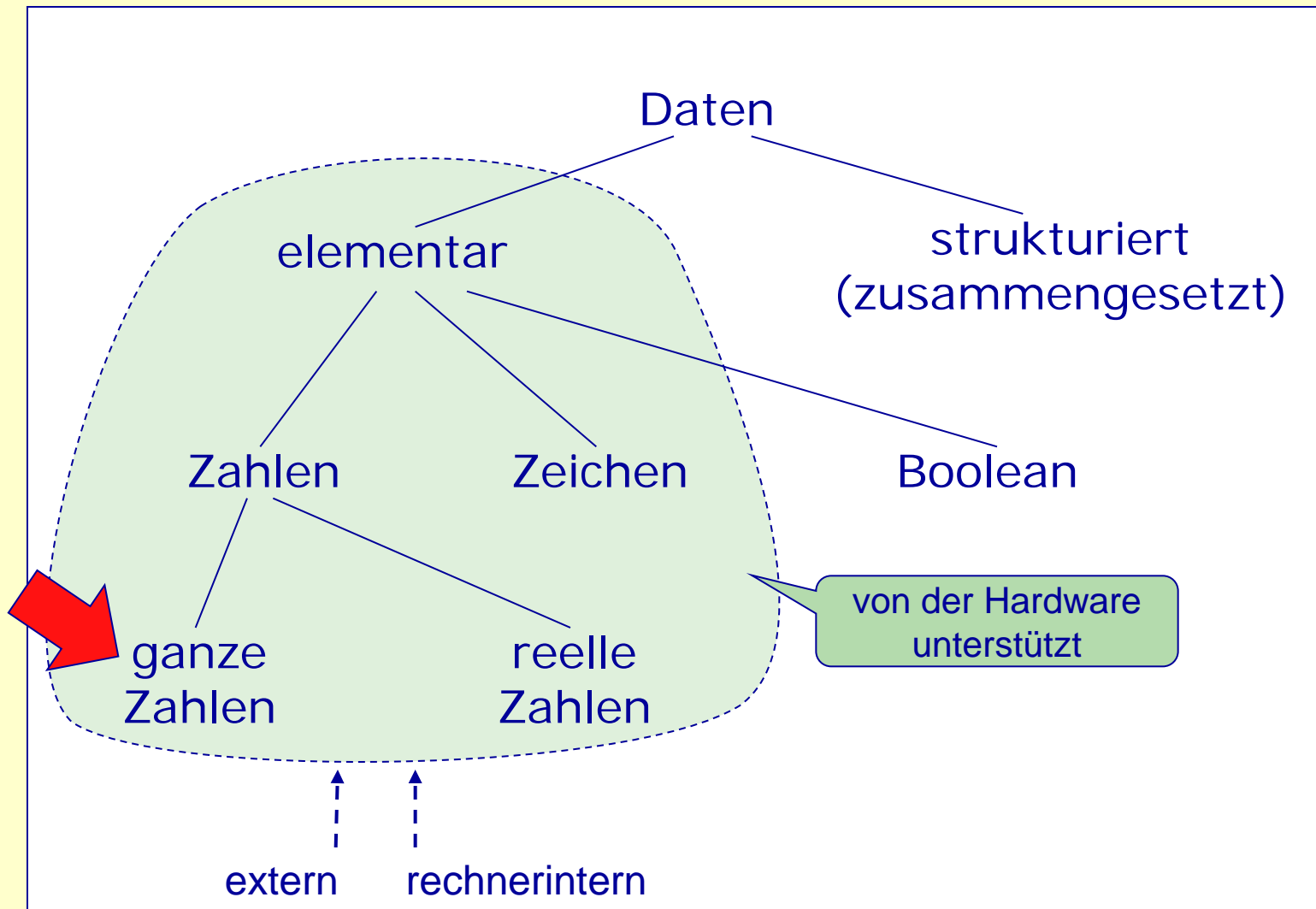
Adresse	Inhalt
60225	...
60226	00110011 01000110 01110000 10000001
60227	10110011 11100110 01110000 00011001
60228	00110011 00000110 01110000 11000001
60229	00000011 00000110 01110000 10000001
60230	10110011 00000110 01110000 01100001
60231	01110011 00000110 01110000 00000101
60232	01100111 00000001 01101011 01000101
60233	00111011 00000110 01110000 11100001
...	...
...	...
280461	00000000 00000000 00000000 00000001
...	...
...	...
440204	00000000 00000000 00000000 00000101
...	...

Rechnerinterne Form wichtig auch bei Nutzung höherer Programmiersprachen:

- Verstehen von Speicherinhalten bei Laufzeitfehlern (Speicherdump)
- Größenbeschränkungen von Zahlen in Programmen
- Rundungsprobleme bei reellen Zahlen

# Ganze Zahlen

# Klassifikation von Daten





# Zahlendarstellungen: Ganze Zahlen - externe Form

- C, C++, JAVA u. a.: unterschiedliche Datentypen

<b>short:</b>	im Bereich $-2^{15} \dots 2^{15} - 1$	(2 Byte = 16 Bit)
<b>int:</b>	im Bereich $-2^{31} \dots 2^{31} - 1$	(4 Byte = 32 Bit)
<b>long:</b>	im Bereich $-2^{63} \dots 2^{63} - 1$	(8 Byte = 64 Bit)

z. B.: 100, -1, 0

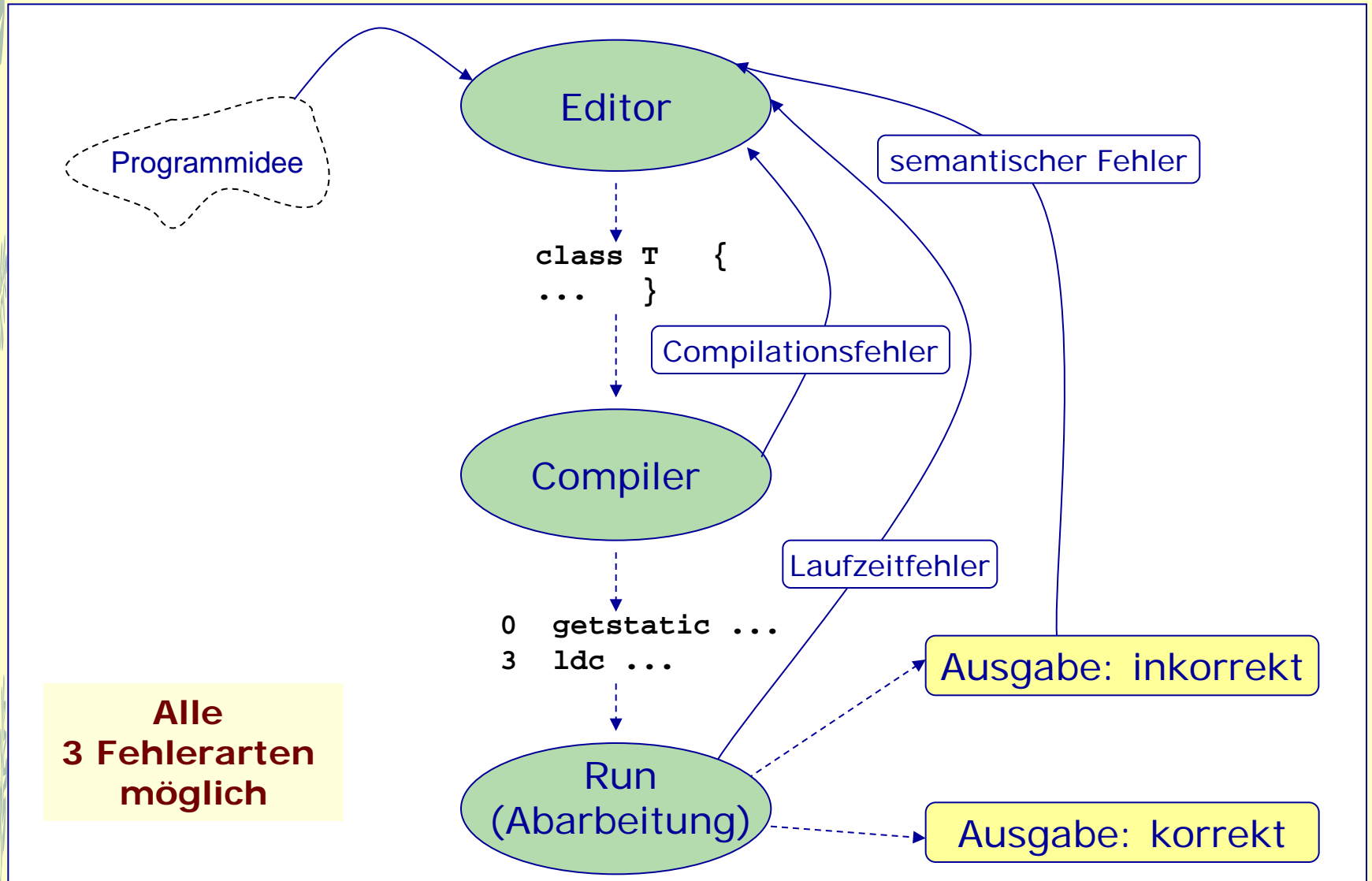
größte int: 2147483647  $(2^{31} - 1)$

kleinste int: -2147483648  $(-2^{31})$



Inhaltliche  
Begründung?

# Fehlerart bei Überschreitung der Zahlenbereiche?



# Fehler: zu große bzw. zu kleine Zahlen

Bereich int:

a) im Programmtext:  $x = 21474836470; (10 * \text{maxint})$

b) als Resultat einer Operation:  $y = 2147483647 - b$   
(zur Abarbeitungszeit:  $b < 0$ )

Welche Fehlerart:

a) → Compilationsfehler (lexikalischer Fehler)

b) → Laufzeitfehler (integer overflow) oder semantischer Fehler

## Achtung:

- Laufzeitfehler meist nicht angezeigt, sondern semantischer Fehler (abhängig von Rechnerarchitektur und z.T. von Programmiersprache)
- Damit: Resultat ist falsche Zahl, mit der weitergearbeitet wird  
→ falsche Ergebnisse erscheinen als semantische Fehler

*oft: größte int-Zahl + 1 → kleinste int-Zahl*

*2147483647 + 1 → -2147483648*

*d.h.  $(2^{31} - 1) + 1 \rightarrow (-2^{31})$*

Integer  
Overflow



Europäische  
Trägerrakete  
Ariane 5:  
Absturz 1996

# Positive ganze Zahlen: rechnerinterne Repräsentation

Binäre Darstellung (Dualzahl) mit zwei Ziffern: 0 und 1

$0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 10, 3 \rightarrow 11, 4 \rightarrow 100, 5 \rightarrow 101 \dots$

$24 = 11000_2$ , intern als Typ *int* mit 4 Byte:

00000000 00000000 00000000 00011000

Maschineninterne Operationen mit Dualzahlen:

$$\begin{array}{ccccccc} 24 & + & 16 & = & 40 & & \\ \downarrow & & \downarrow & & \uparrow & & \\ 11000 & + & 10000 & = & 101000 & & \end{array}$$

# Wert einer Dualzahl (Binärzahl)

Dualzahl:

00000000 00010100

$b_{n+1}b_n b_{n-1} \dots b_2 b_1$

Wert:

$$x = \sum_{i=1, \dots, n} b_i \cdot 2^{i-1}$$

(short: n=15, int: n=31, long: n=63)

z.B.  $10100_2 \rightarrow 0 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 = 20$

- 1. Bit "frei" für Vorzeichen (positiv: 0, neg.: 1)
- in 15 Bit: größte positive short-Zahl:  $2^{15} - 1$

01111111 11111111

# Negative ganze Zahlen: rechnerinterne Repräsentation

Dualzahl (Binärzahl):

$$\mathbf{b_{n+1}b_n b_{n-1} \dots b_2 b_1}$$

erstes Bit entscheidet: positiv oder negativ?

- $b_{n+1} = 0$  für positive Zahlen
- $b_{n+1} = 1$  für negative Zahlen

Verschiedene Varianten (abh. von Rechnerarchitektur)

# Variante 1: einfache Vorzeichenbit-Technik

Dualzahl (Binärzahl):  $b_{n+1}b_nb_{n-1}\dots b_2b_1$

$b_{n+1} = 0$  für positive Zahlen

$b_{n+1} = 1$  für negative Zahlen

sonst wie bisher

$24 = 11000_2$ , Typ short mit 2 Byte:

00000000 00011000

$-24 = -11000_2$ , Typ short mit 2 Byte:

10000000 00011000

# Variante 2: 1-er-Komplement

-x: bitweise Umkehrung der Dual-Ziffern

24 = 11000<sub>2</sub>, Typ short mit 2 Byte:

```
00000000 00011000
```

-24 = -11000<sub>2</sub>, Typ short mit 2 Byte:

```
11111111 11100111
```



# Variante 3: 2-er-Komplement

-x: Einerkomplement + 1

-24 = -11000<sub>2</sub>, Typ short mit 2 Byte:

+ 24

00000000 00011000

Einerkomplement:

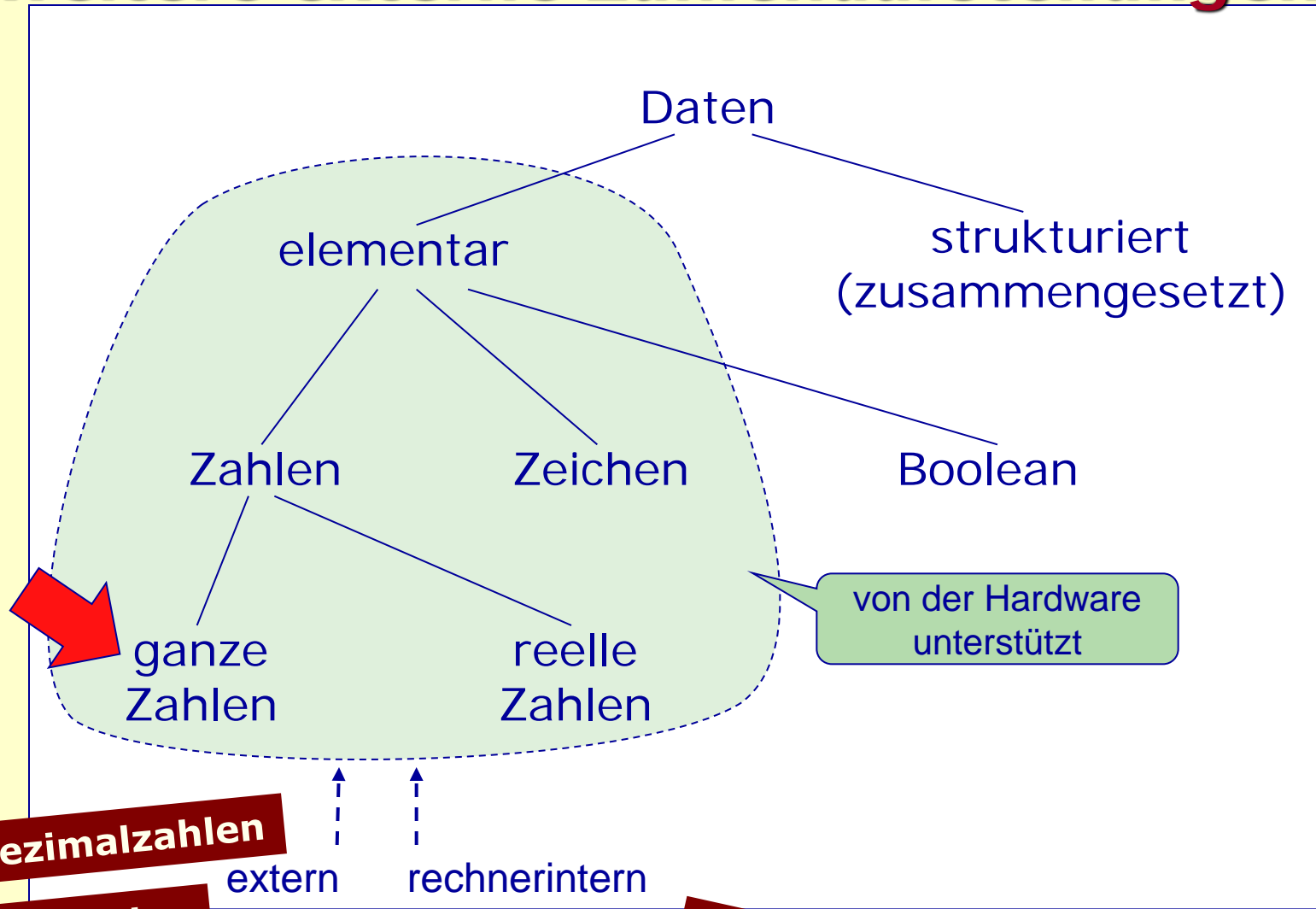
11111111 11100111

Zweierkomplement:

11111111 11101000

Heutige Rechner: meist 2-er-Komplement  
(Addition positiver und negativer Zahlen einfach)

# Ganze Zahlen: weitere externe Zahlendarstellungen



**Hexadezimalzahlen**

**Dezimalzahlen**

**Oktalzahlen**

**Binärzahlen**

# Ganze Zahlen: weitere externe Zahlendarstellungen

(bisher Zahlen mit Basis: 10, 2)

- a) Oktal-Zahl mit Ziffern 0, ..., 7  
(passen in 3 Bit)

Wert der Oktalzahl  $s_n s_{n-1} \dots s_2 s_1$

$$x = \sum_{i=1, \dots, n} s_i \cdot 8^{i-1}$$

z. B.  $14_8 = 12_{10}$

in C, Java u. a.: 014 (führende Null)

- b) Hexadezimal-Zahl  
mit Ziffern 0, ..., 9, A, ..., F  
(passen in 4 Bit)

Wert der Hexadezimalzahl  $s_n s_{n-1} \dots s_2 s_1$

$$x = \sum_{i=1, \dots, n} s_i \cdot 16^{i-1}$$

z. B.  $14_{16} = 20_{10}$ ,  $1A_{16} = 26_{10}$

in C, Java u. a.: 0x14, 0x1A,  
0x14, 0xCAFE

# Zusammenfassendes Beispiel

$$3E8_{16} = 1000_{10} = 1750_8 = 1111101000_2$$

→ Umrechnungsalgorithmen: Übungen

# Zum Nachdenken

- Warum werden überhaupt unterschiedliche Zahlensysteme verwendet?
- Es kann eine negative Zahl mehr (als bei positiven Zahlen) im 2er-Komplement dargestellt werden.
  - Wieso?
  - Beispiel: Wie sieht die interne Darstellung der kleinsten (short-, d.h. 16 Bit-)Zahl aus?  
→ größte Zahl:

**01111111 11111111**

→ kleinste Zahl: ?

# Warum verschiedene Zahlensysteme? (Basis 2, 8, 10, 16 u. a.)

- Dezimal: lesbar
- Binär: maschinenintern  
`0110 1111 0100 1100`
- Hexadezimal: komprimierte maschineninterne Form:  
Speicherdump: `6F4C`  
(1 Byte durch 2 Ziffern dargestellt)
- Oktal: passen in 3 Bit

# Interne Darstellung: kleinste ganze Zahl?

Interne Darstellung: größte Zahl  $Z_{\max}$

01111111 11111111

short:  $2^{15}-1$

Interne Darstellung: kleinste Zahl =  $-(Z_{\max} + 1)$  ?

---

$Z_{\max} + 1$ :

(0) 10000000 00000000

falls nicht nur 16 Bit

1er Komplement:

(1) 01111111 11111111

2er Komplement: + 1

10000000 00000000

short:  $-2^{15}$

# Überschreitung von Zahlenbereichen: semantischer Fehler entsteht

value = ... größte int-Zahl

01111111 11111111

short:  $2^{15}-1$

value = value + 1; ... kleinste int-Zahl

10000000 00000000

short:  $-2^{15}$

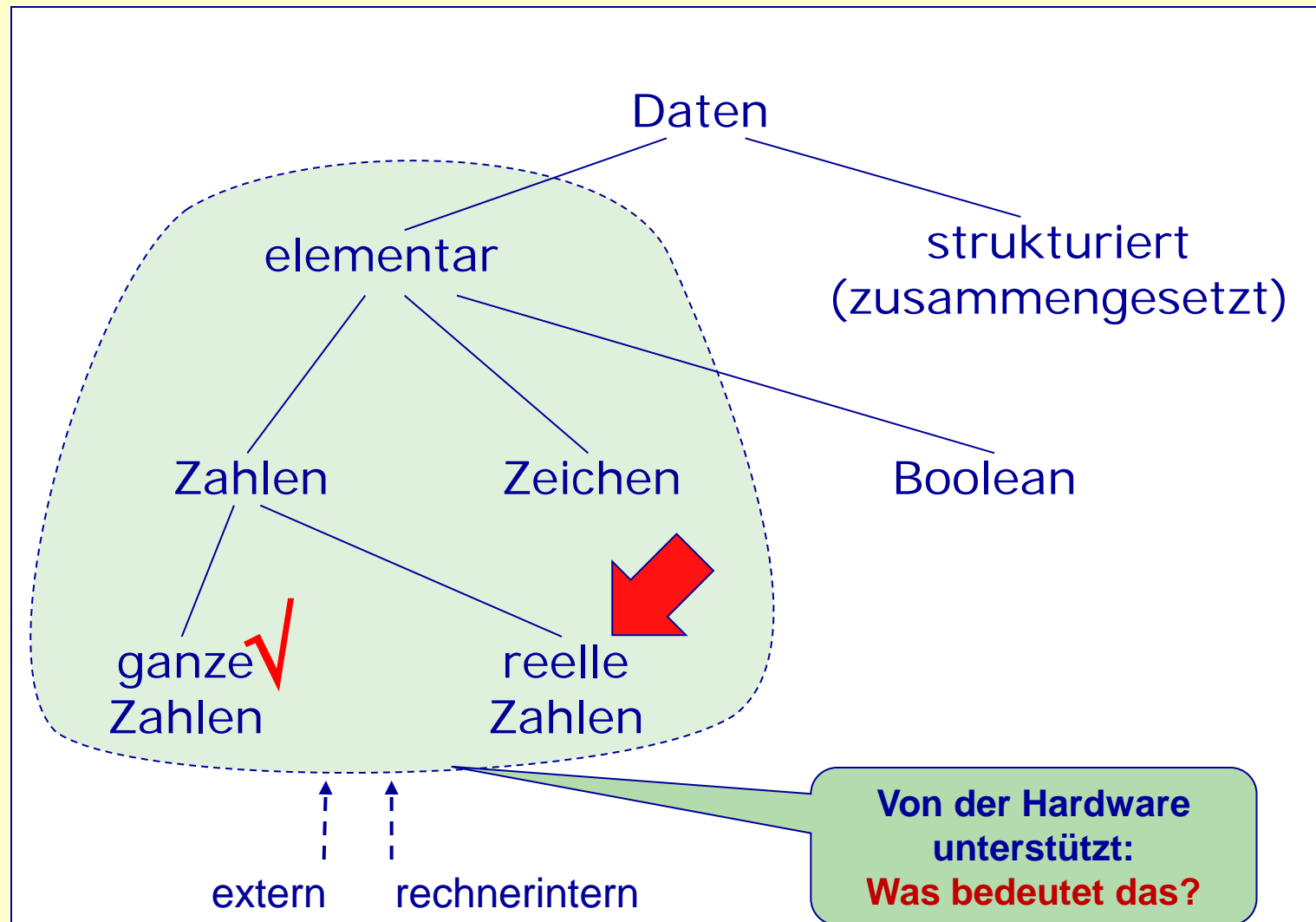


# **Reelle Zahlen**

**Gleitkommazahlen**

**Floating-point numbers**

# Klassifikation von Daten



# Reelle Zahlen: externe Form

C, C++, JAVA u. a.: 2 Typen (mit 2 Bereichen)

float (4 Byte) im Bereich

$$-3.40282347 * 10^{38} \dots +3.40282347 * 10^{38}$$

double (8 Byte) im Bereich

$$-1.79769313486231570 * 10^{308} \dots +1.79769313486231570 * 10^{308}$$

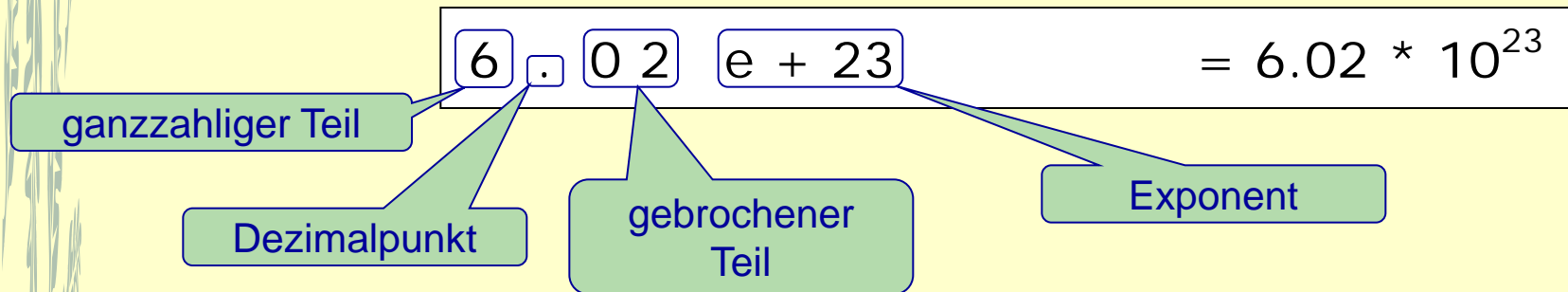
- Begründung für 'krumme' Bereiche:  
interne Repräsentation
- Nicht jede reelle Zahl ist repräsentierbar – warum?

In jedem endlichen Intervall  $[a, b]$  liegen unendlich viele reelle Zahlen, aber nur endlich viele davon können in  $n$  Bytes ( $n = 4$  oder  $8$ ) repräsentiert werden!

→ Rundungsfehler normal, z.B. Zahl Pi

# Reelle Zahlen: externe Form (cont.)

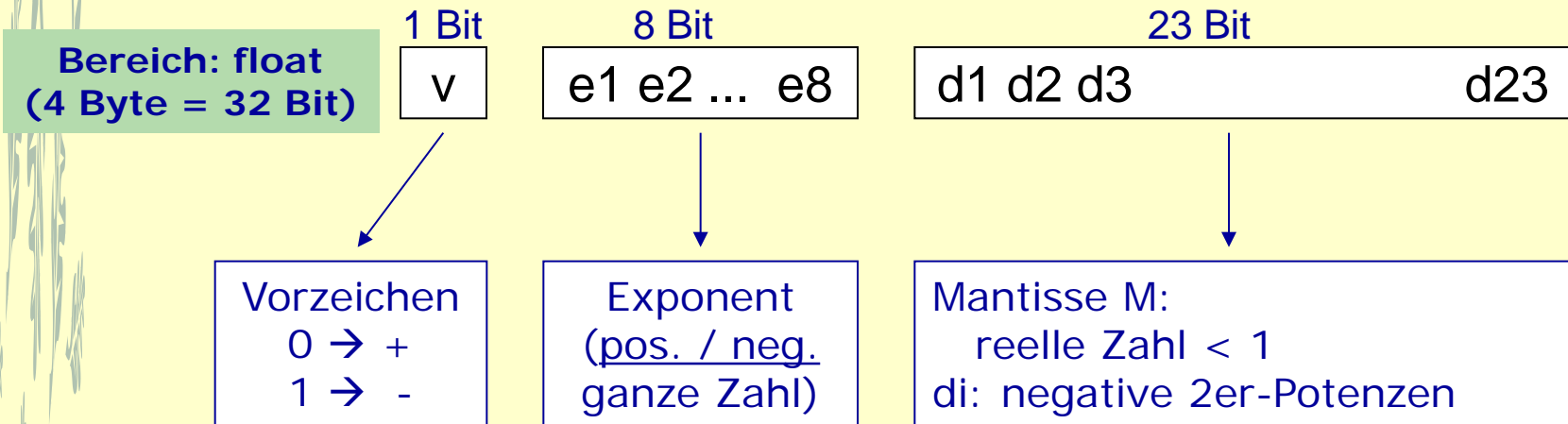
(C, C++, Java, Pascal)



Teile optional:

- 1 e -2 (kein gebrochener Teil)
- 3.14 (kein Exponent)
- .3 (kein ganzzahliger Teil)
- 2. (kein gebrochener Teil)
- 6.02 e 2 (kein Vorzeichen im Exponenten)
- 2** keine reelle Zahl, sondern anderer Zahlenbereich:  
ganze Zahl

# Reelle Zahlen: rechnerinterne Form



Für float/4 Byte: v, e1, e2, ... e8, d1, d2 ... d23:

**Exponent  $exp$  zwischen  $2^7-1$  und  $-2^7$  (127 und -128)**

$$M = d1 * 2^{-1} + d2 * 2^{-2} + \dots + d23 * 2^{-23}$$

$$\text{Dargestellte reelle Zahl } r = [+/-] 2^{exp} (1+M)$$

Andere Varianten beim Exponenten möglich: exp immer positiv interpretiert:  $2^{exp-127}$

# Beispiel

1	00000010	01000000 00000000 00000000
---	----------	----------------------------

$v = -1$

$\text{exp} = 2$

Mantisse  $M = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3}$

$$r = (-1) \cdot 2^2 \cdot (1 + 1 / 4) = -4 \cdot 5 / 4 = \mathbf{-5}$$

**Andere Variante bei Interpretation des Exponenten:**

$$\mathbf{2^{2-127} = 2^{-125}}$$

$$r = (-1) \cdot 2^{-125} \cdot (1 + 1 / 4) = -2^{-125} \cdot 5 / 4 = \mathbf{-2^{-127} \cdot 5}$$

Und die reelle Zahl 0.0 ?

Dargestellte reelle Zahl  $r = [+/-] 2^{\text{exp}} (1+M)$

# Repräsentation der reellen Zahl 0.0

Spezielle Festlegungen: bisher 0.0 nicht repräsentierbar

$$r = [+/-] 2^{\text{exp}} (1+M)$$

Hardware: keine einheitliche Regelung

ein Beispiel: **IEEE 754 floating-point standard**

(Hennessy, Patterson: Computer Architecture)

Mantisse M: 0

Exponent exp: kleinste mögliche negative Zahl im Exponentenfeld

0	10000000	00000000	00000000	00000000	00000000
---	----------	----------	----------	----------	----------



# Ganze Zahl $\leftrightarrow$ reelle Zahl

Notation im Programm: -5.0 oder -5

11111111 11111111 11111111 11111011

anders repräsentiert  
(anderer Zahlenbereich)

1 | 00000010 | 01000000 00000000 00000000

$$v = -1$$

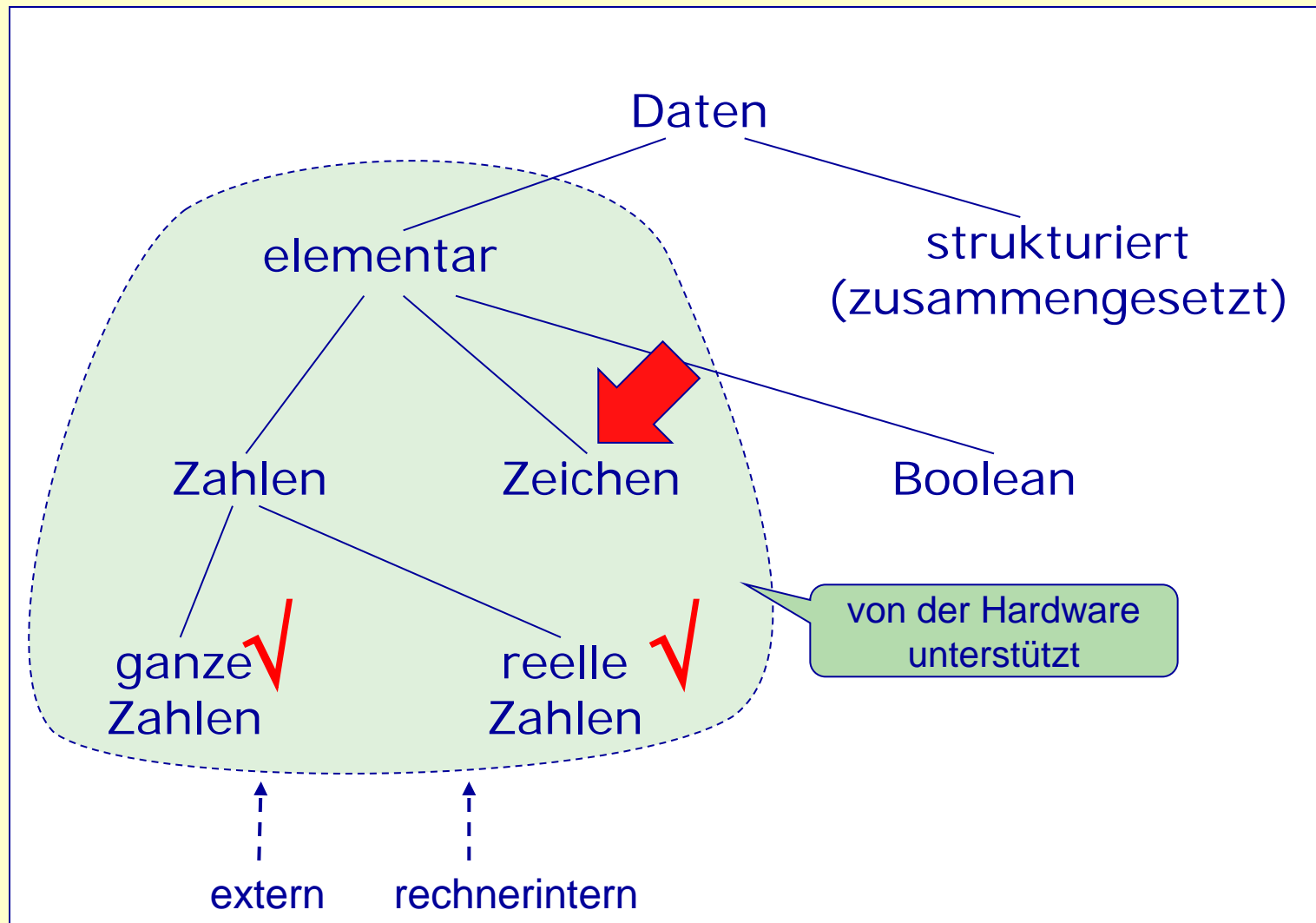
$$\text{exp} = 2$$

$$\text{Mantisse } M = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3}$$

$$r = (-1) \cdot 2^2 \cdot (1 + 1/4) = -4 \cdot 5/4 = -5$$

# Zeichen

# Klassifikation von Daten



# Zeichen (char)

Schreibmaschinenzeichen und weitere Zeichen,  
u. a. Steuerzeichen (CTRL ..)

z.B.

CR = \carriage return" (Wagenrücklauf)

LF = \line feed" (neue Zeile)

---

---

## Wichtige Zeichensätze:

### 1. ASCII Zeichensatz

(American Standard Code for Information Interchange)

7-bit-Code (1 Byte mit führendem Bit = 0)

→  $2^7 = 128$  Zeichen

→ ausreichend für "normale" Programmierung

(mit lateinischen Zeichen, Ziffern, Operatoren, Spezielles)

# Zeichen (char)

## 2. LATEIN1 (ISO-8859-1) Zeichensatz (umfasst ASCII)

8-bit-Code (1 Byte)

→  $2^8 = 256$  Zeichen

## 3. Unicode Zeichensatz (umfasst ISO-8859-1)

16-Bit-Code (2 Byte - Basisversion)

→  $2^{16} = 65536$  Zeichen

Schriftzeichen der wichtigsten Sprachen codiert:

Griechisch, Chinesisch, Kyrillisch, Arabisch ...

*Unicode Version 6.0 (Okt. 2010): 109.242 Zeichen (32 Bit)*

JAVA verwendet Unicode

# ASCII-Code

Anwendung:  
Hexadezimale  
Zahlendarstellung  
(je Byte: 2 Hex-Ziffern)

Zeichen „/“:  
47 → 2F = 0010 1111

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

# Beispiel: ASCII-Code

binär:

0010 1100	0100 0011	0101 1100	0110 0001
-----------	-----------	-----------	-----------

hexadezimal:

2C

43

5C

61

als Zeichenfolge (dekodiert nach ASCII-Tabelle):

, C \ a

# Unicode: dargestellte Schriftsätze

## The Unicode Character Code Charts By Script

[SYMBOLS AND PUNCTUATION](#) | [NAME INDEX](#) | [HELP AND LINKS](#)

European Alphabets	African Scripts	Indic Scripts	East Asian Scripts	Central Asian Scripts
(see also <b>Comb. Marks</b> )	<b>Ethiopic</b>	Bengali	<b>Han Ideographs</b>	Kharoshthi
<b>Armenian</b>	Ethiopic	Devanagari	<b>Unified CJK Ideographs</b> (5MB)	Mongolian
Armenian	Ethiopic Supplement	Gujarati	CJK Ideographs Ext. A (2MB)	Phags-Pa (5.0)
<i>Armenian Ligatures</i>	Ethiopic Extended	Gurmukhi	CJK Ideographs Ext. B (13MB)	Tibetan
<b>Coptic</b>	<b>Other African scripts</b>	Kannada	Compatibility Ideographs (.5MB)	
Coptic	N'Ko (5.0)	Limbu	Compatibility Ideo. Suppl. (.5MB)	
<i>Coptic in Greek block</i>	Tifinagh	Malayalam	Kanbun	
<b>Cyrillic</b>	<b>Middle Eastern Scripts</b>	Oriya	(see also <b>Unihan Database</b> )	<b>Ancient Scripts</b>
Cyrillic	<b>Arabic</b>	Sinhala	<b>Radicals and Strokes</b>	<b>Ancient Greek</b>
Cyrillic Supplement	Arabic	Syloti Nagri	CJK Radicals	Ancient Greek Numbers
<b>Georgian</b>	Arabic Supplement	Tamil	KangXi Radicals	Ancient Greek Musical
Georgian	Arabic Presentation Forms A	Telugu	CJK Strokes	<b>Cuneiform</b>
Georgian Supplement	Arabic Presentation Forms B		Ideographic Description	Cuneiform (5.0)
<b>Greek</b>	<b>Hebrew</b>	<b>Philippine Scripts</b>	<b>Chinese-specific</b>	Cuneiform Numbers (5.0)
Greek	Hebrew	Buhid	Bopomofo, Extended	Old Persian
Greek Extended	<i>Hebrew Presentation Forms</i>	Hanunoo	<b>Japanese-specific</b>	Ugaritic
(see also <b>Ancient Greek</b> )	<b>Other ME Scripts</b>	Tagalog	Hiragana	<b>Linear B</b>
<b>Latin</b>	Syriac	Tagbanwa	Katakana,	Linear B Syllabary
Basic Latin	Thaana		Katakana Phonetic Ext.	<b>Linear B Ideograms</b>
Latin-1	<b>American scripts</b>	<b>South East Asian</b>	<i>Halfwidth Katakana</i>	<b>Other Ancient Scripts</b>
Latin Extended A	Canadian Syllabics	Buginese	<b>Korean-specific</b>	Aegean Numbers
Latin Extended B	Cherokee	Balinese (5.0)	Hangul Syllables (4MB)	Counting Rod Num. (5.0)
Latin Extended C (5.0)	Deseret	Khmer	Hangul Jamo	Cypriot Syllabary
Latin Extended D (5.0)		Khmer Symbols	Hangul Compatibility Jamo	Gothic
Latin Extended Additional	<b>Other Scripts</b>	Lao	<i>Halfwidth Jamo</i>	Old Italic
<i>Latin Ligatures</i>	Shavian	Myanmar	<b>Yi</b>	Ogham
<i>Fullwidth Latin Letters</i>	Osmanya	New Tai Lue	YI (.6MB)	Runic
Small Forms	Glagolitic	Tai Le	Yi Radicals	Phoenician (5.0)
(see also <b>Phonetic Symbols</b> )		Thai		



# Unicode: Basic Latin (ASCII) und Latin-1

Zu lesen:

- Alles hexadezimal

- z.B. Spalte 004, Zeile 1  
→ Gesamt 2 Byte 0041  
→ 'A' repräsentiert

C0 Controls and Basic Latin

	000	001	002	003	004	005	006	007
0	NUL	DLE	SP	0	@	P	~	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOQ	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

C1 Controls and Latin-1 Supplement

	008	009	00A	00B	00C	00D	00E	00F
0	XXX	DCS	NB SP	°	À	Đ	à	đ
1	XXX	PU1	¡	±	Á	Ñ	á	ñ
2	BPH	PU2	¢	²	Â	Ò	â	ò
3	NBH	STS	£	³	Ã	Ó	ã	ó
4	IND	CCH	¤	´	Ä	Ô	ä	ô
5	NEL	MW	¥	µ	Å	Õ	å	õ
6	SSA	SPA	¦	¶	Æ	Ö	æ	ö
7	ESA	EPA	§	·	Ç	×	ç	÷
8	HTS	SOS	¨	¸	È	Ø	è	ø
9	HTJ	XXX	©	¹	É	Ù	é	ù
A	VTS	SCI	ª	º	Ê	Ú	ê	ú
B	PLD	CSI	«	»	Ë	Û	ë	û
C	PLU	ST	¬	¼	Ì	Ü	ì	ü
D	RI	OSC	½	½	Í	Ý	í	ý
E	SS2	PM	®	¾	Î	Þ	î	þ
F	SS3	APC	¸	¿	Ï	ß	ï	ÿ

# Unicode: Kyrillisch

	Cyrillic															
	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	040A	040B	040C	040D	040E	040F
0	È	А	Р	а	р	è	Ѡ	Ѳ	Ѵ	Г	К	У	І	Ă	З	Û
1	Ë	Б	С	б	с	ë	ѡ	ѳ	ѵ	г	к	у	Ѣ	ă	з	Û
2	Ђ	В	Т	в	т	ђ	Ѣ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ
3	Ѓ	Г	У	г	у	ѓ	ѣ	ѧ	ѩ	ѭ	ѯ	ѱ	ѳ	ѵ	ѷ	ѹ
4	Є	Д	Ф	д	ф	є	Ѫ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ	Ѽ
5	Є	Д	Ф	д	ф	є	Ѫ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ	Ѽ
6	І	Ж	Ц	ж	ц	і	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
7	Ї	З	Ч	з	ч	ї	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
8	Ј	И	Ш	и	ш	ј	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
9	Љ	Й	Щ	й	щ	љ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
A	Њ	К	Ъ	к	ъ	њ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
B	Ѣ	Л	Ы	л	ы	ѣ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
C	Ќ	М	Ь	м	ь	ќ	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
D	Ў	Н	Э	н	э	ў	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
E	Ў	О	Ю	о	ю	ў	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ
F	Ѧ	П	Я	п	я	Ѧ	Ѩ	Ѭ	Ѯ	Ѱ	Ѳ	Ѵ	Ѷ	Ѹ	Ѻ	Ѽ

# Unicode: Arabisch

	Arabic															
	0600	0601	0602	0603	0604	0605	0606	0607	0608	0609	060A	060B	060C	060D	060E	060F
0	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط
1	ظ	ع	ف	ق	ك	ل	م	ن	هـ	و	ز	ح	ج	ب	ا	آ
2	أ	إ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
3	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
4	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
5	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
6	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
7	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
8	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
9	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
A	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
B	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
C	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
D	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
E	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ
F	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ	أ

# Unicode: Chinesisch (Han) 82 Tabellen, ca. 20.000 Zeichen

	CJK Unified Ideographs															
	4E00	4E01	4E02	4E03	4E04	4E05	4E06	4E07	4E08	4E09	4E0A	4E0B	4E0C	4E0D	4E0E	4E0F
0	一 4E00	丐 4E01	北 4E02	丰 4E03	乂 4E04	乐 4E05	习 4E06	买 4E07	龟 4E08	亏 4E09	冫 4E0A	京 4E0B	什 4E0C	仝 4E0D	仵 4E0E	仰 4E0F
1	丁 4E10	丑 4E11	兩 4E12	卯 4E13	乂 4E14	禾 4E15	乡 4E16	乱 4E17	乾 4E18	云 4E19	亡 4E1A	恒 4E1B	仁 4E1C	仑 4E1D	仝 4E1E	伶 4E1F
2	亏 4E20	刃 4E21	丢 4E22	串 4E23	乂 4E24	兵 4E25	虺 4E26	盗 4E27	亂 4E28	互 4E29	亢 4E2A	亲 4E2B	仝 4E2C	令 4E2D	仝 4E2E	仲 4E2F
3	七 4E30	专 4E31	卯 4E32	弗 4E33	乃 4E34	兵 4E35	纟 4E36	乳 4E37	粼 4E38	亅 4E39	亅 4E3A	毫 4E3B	仝 4E3C	仓 4E3D	代 4E3E	仝 4E3F
4	上 4E40	且 4E41	兩 4E42	临 4E43	乂 4E44	乔 4E45	乏 4E46	𠂇 4E47	五 4E48	交 4E49	竟 4E4A	仄 4E4B	仔 4E4C	令 4E4D	月 4E4E	
5	丁 4E50	丕 4E51	严 4E52	犖 4E53	久 4E54	厶 4E55	乏 4E56	乳 4E57	丁 4E58	井 4E59	亥 4E5A	衰 4E5B	仗 4E5C	仕 4E5D	以 4E5E	作 4E5F
6	厂 4E60	世 4E61	並 4E62	丶 4E63	从 4E64	乖 4E65	书 4E66	𠂇 4E67	了 4E68	三 4E69	亦 4E6A	亼 4E6B	他 4E6C	仆 4E6D	件 4E6E	
7	万 4E6F	卅 4E70	喪 4E71	丩 4E72	毛 4E73	乘 4E74	𠂇 4E75	𠂇 4E76	尔 4E77	𠂇 4E78	产 4E79	廉 4E7A	仇 4E7B	仗 4E7C	夫 4E7D	价 4E7E
8	丈 4E7F	丘 4E80	丨 4E81	丸 4E82	么 4E83	乘 4E84	乱 4E85	𠂇 4E86	予 4E87	亅 4E88	亨 4E89	彡 4E8A	仆 4E8B	付 4E8C	仁 4E8D	伏 4E8E
9	三 4E8F	丙 4E90	𠂇 4E91	丹 4E92	义 4E93	乙 4E94	𠂇 4E95	𠂇 4E96	争 4E97	瓦 4E98	亩 4E99	亼 4E9A	仇 4E9B	仙 4E9C	仕 4E9D	伴 4E9E
A	上 4EA0	业 4EA1	个 4EA2	为 4EA3	𠂇 4EA4	𠂇 4EA5	𠂇 4EA6	𠂇 4EA7	𠂇 4EA8	𠂇 4EA9	𠂇 4EAA	人 4EAB	今 4EAC	𠂇 4EAD	仪 4EAE	𠂇 4EAF
B	下 4EB0	丛 4EB1	丫 4EB2	主 4EB3	之 4EB4	一 4EB5	𠂇 4EB6	𠂇 4EB7	𠂇 4EB8	𠂇 4EB9	𠂇 4EBA	𠂇 4EBB	介 4EBC	任 4EBD	仝 4EBE	任 4EBF
C	丌 4EC0	东 4EC1	丩 4EC2	井 4EC3	乌 4EC4	乂 4EC5	𠂇 4EC6	𠂇 4EC7	二 4EC8	𠂇 4EC9	京 4ECA	亼 4ECB	欠 4ECC	仝 4ECD	仝 4ECE	仝 4ECF
D	不 4ED0	丝 4ED1	中 4ED2	丽 4ED3	乍 4ED4	九 4ED5	𠂇 4ED6	𠂇 4ED7	𠂇 4ED8	𠂇 4ED9	亭 4EDA	亼 4EDB	仍 4EDC	仝 4EDD	仝 4EDE	仝 4EDF
E	与 4EE0	丞 4EE1	𠂇 4EE2	举 4EE3	乎 4EE4	乞 4EE5	𠂇 4EE6	乾 4EE7	于 4EE8	亞 4EE9	亮 4EEA	亼 4EEB	从 4EEC	仝 4EED	仝 4EEE	仝 4EEF
F	丐 4EF0	丢 4EF1	𠂇 4EF2	𠂇 4EF3	乏 4EF4	也 4EF5	𠂇 4EF6	亂 4EF7	亏 4EF8	𠂇 4EF9	盲 4EFA	亿 4EFB	仝 4EFC	仝 4EFD	仝 4EFE	仝 4EFF

# Unicode: Griechisch und Koptisch

0370

Greek and Coptic

03FF

	037	038	039	03A	03B	03C	03D	03E	03F
0			í 0398	Π 03A6	ú 03B8	π 03C0	β 03D0	ϗ 03E0	κ 03F4
1			Α 0391	Ρ 03A1	α 03B1	ρ 03C1	θ 03D1	ϛ 03E1	ρ 03F1
2			Β 0392		β 03B2	ς 03C2	Υ 03D2	Ϝ 03E2	Ϙ 03F2
3			Γ 0393	Σ 03A3	γ 03B3	σ 03C3	Υ 03D3	ϝ 03E3	ι 03F3
4	´ 0374	´ 0384	Δ 0394	Τ 03A4	δ 03B4	τ 03C4	ÿ 03D4	Ϟ 03E4	θ 03F4
5	´ 0375	ˆ 0385	Ε 0395	Υ 03A5	ε 03B5	υ 03C5	φ 03D5	ϙ 03E5	ε 03F5
6			Ά 0396	Ζ 0396	Φ 03A6	ζ 03B6	φ 03C6	π 03D6	ϛ 03E6
7		· 0387	Η 0397	Χ 03A7	η 03B7	χ 03C7	ϝ 03D7	Ϟ 03E7	Ϙ 03F7
8		Έ 0398	Θ 0398	Ψ 03A8	θ 03B8	ψ 03C8	Ϟ 03D8	ϙ 03E8	β 03F8
9		Ή 0399	Ι 0399	Ω 03A9	ι 03B9	ω 03C9	ϙ 03D9	ϙ 03E9	Ϙ 03F9
A	˘ 037A	˘ 038A	Ϊ 039A	Κ 03A9	ϊ 03B9	κ 03C9	ς 03D9	ϙ 03E9	Μ 03FA
B			Λ 039B	ÿ 03AB	λ 03BB	ü 03CB	ς 03DB	ϙ 03EB	Ϙ 03FB
C			Ό 039C	Μ 039C	ά 03AC	μ 03BC	ό 03CC	Ϟ 03DC	ρ 03FC
D			Ν 039D	έ 03AD	ν 03BD	ύ 03CD	Ϟ 03DD	ϙ 03ED	Ϙ 03FD
E	˘ 037E	Ύ 039E	Ξ 039E	ή 03AE	ξ 03BE	ώ 03CE	ϙ 03DE	† 03EE	Ϙ 03FE
F			Ό 039F	Ο 039F	ί 03AF	ο 03BF		ϙ 03DF	† 03EF

# Unicode: Mathematische Operatoren

2200

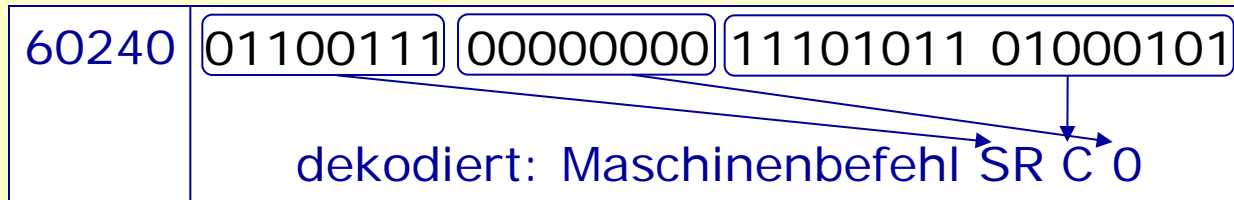
Mathematical Operators

22FF

	220	221	222	223	224	225	226	227	228	229	22A	22B	22C	22D	22E	22F
0	$\nabla$ 2200	$\sqcup$ 2201	$\angle$ 2202	$\mathbb{F}$ 2203	$\wr$ 2204	$\doteq$ 2205	$\neq$ 2206	$\nlessdot$ 2207	$\nlessgtr$ 2208	$\sqsupset$ 2209	$\boxtimes$ 220A	$\wp$ 220B	$\wedge$ 220C	$\in$ 220D	$\nlessdot$ 220E	$\cdot$ 220F
1	$\complement$ 2201	$\Sigma$ 2211	$\Delta$ 2212	$\mathbb{F}$ 2213	$\nlessdot$ 2214	$\doteq$ 2215	$\equiv$ 2216	$\nlessdot$ 2217	$\nlessgtr$ 2218	$\sqsupset$ 2219	$\square$ 221A	$\wp$ 221B	$\nlessdot$ 221C	$\nlessdot$ 221D	$\nlessdot$ 221E	$\cdot$ 221F
2	$\partial$ 2202	$-$ 2212	$\nlessdot$ 2222	$\mathbb{F}$ 2232	$\nlessdot$ 2242	$\doteq$ 2252	$\nlessdot$ 2262	$\nlessdot$ 2272	$\nlessdot$ 2282	$\sqsupset$ 2292	$\sqsupset$ 22A2	$\nlessdot$ 22B2	$\nlessdot$ 22C2	$\nlessdot$ 22D2	$\nlessdot$ 22E2	$\nlessdot$ 22F2
3	$\exists$ 2203	$\nlessdot$ 2213	$\nlessdot$ 2223	$\mathbb{F}$ 2233	$\nlessdot$ 2243	$\doteq$ 2253	$\nlessdot$ 2263	$\nlessdot$ 2273	$\nlessdot$ 2283	$\sqsupset$ 2293	$\sqsupset$ 22A3	$\nlessdot$ 22B3	$\nlessdot$ 22C3	$\nlessdot$ 22D3	$\nlessdot$ 22E3	$\nlessdot$ 22F3
4	$\nlessdot$ 2204	$\nlessdot$ 2214	$\nlessdot$ 2224	$\nlessdot$ 2234	$\nlessdot$ 2244	$\doteq$ 2254	$\nlessdot$ 2264	$\nlessdot$ 2274	$\nlessdot$ 2284	$\sqsupset$ 2294	$\sqsupset$ 22A4	$\nlessdot$ 22B4	$\nlessdot$ 22C4	$\nlessdot$ 22D4	$\nlessdot$ 22E4	$\nlessdot$ 22F4
5	$\emptyset$ 2205	$/$ 2215	$\parallel$ 2225	$\nlessdot$ 2235	$\nlessdot$ 2245	$\doteq$ 2255	$\nlessdot$ 2265	$\nlessdot$ 2275	$\nlessdot$ 2285	$\nlessdot$ 2295	$\nlessdot$ 22A5	$\nlessdot$ 22B5	$\nlessdot$ 22C5	$\nlessdot$ 22D5	$\nlessdot$ 22E5	$\nlessdot$ 22F5
6	$\Delta$ 2206	$\nlessdot$ 2216	$\nlessdot$ 2226	$\nlessdot$ 2236	$\nlessdot$ 2246	$\doteq$ 2256	$\nlessdot$ 2266	$\nlessdot$ 2276	$\nlessdot$ 2286	$\nlessdot$ 2296	$\nlessdot$ 22A6	$\nlessdot$ 22B6	$\nlessdot$ 22C6	$\nlessdot$ 22D6	$\nlessdot$ 22E6	$\nlessdot$ 22F6
7	$\nlessdot$ 2207	$\nlessdot$ 2217	$\nlessdot$ 2227	$\nlessdot$ 2237	$\nlessdot$ 2247	$\doteq$ 2257	$\nlessdot$ 2267	$\nlessdot$ 2277	$\nlessdot$ 2287	$\nlessdot$ 2297	$\nlessdot$ 22A7	$\nlessdot$ 22B7	$\nlessdot$ 22C7	$\nlessdot$ 22D7	$\nlessdot$ 22E7	$\nlessdot$ 22F7
8	$\in$ 2208	$\circ$ 2218	$\nlessdot$ 2228	$\nlessdot$ 2238	$\nlessdot$ 2248	$\doteq$ 2258	$\nlessdot$ 2268	$\nlessdot$ 2278	$\nlessdot$ 2288	$\nlessdot$ 2298	$\nlessdot$ 22A8	$\nlessdot$ 22B8	$\nlessdot$ 22C8	$\nlessdot$ 22D8	$\nlessdot$ 22E8	$\nlessdot$ 22F8
9	$\nlessdot$ 2209	$\cdot$ 2219	$\nlessdot$ 2229	$\nlessdot$ 2239	$\nlessdot$ 2249	$\doteq$ 2259	$\nlessdot$ 2269	$\nlessdot$ 2279	$\nlessdot$ 2289	$\nlessdot$ 2299	$\nlessdot$ 22A9	$\nlessdot$ 22B9	$\nlessdot$ 22C9	$\nlessdot$ 22D9	$\nlessdot$ 22E9	$\nlessdot$ 22F9
A	$\in$ 220A	$\sqrt{\quad}$ 221A	$\cup$ 222A	$\nlessdot$ 223A	$\nlessdot$ 224A	$\nlessdot$ 225A	$\nlessdot$ 226A	$\nlessdot$ 227A	$\nlessdot$ 228A	$\nlessdot$ 229A	$\nlessdot$ 22AA	$\nlessdot$ 22BA	$\nlessdot$ 22CA	$\nlessdot$ 22DA	$\nlessdot$ 22EA	$\nlessdot$ 22FA
B	$\nlessdot$ 220B	$\sqrt[3]{\quad}$ 221B	$\int$ 222B	$\nlessdot$ 223B	$\nlessdot$ 224B	$\nlessdot$ 225B	$\nlessdot$ 226B	$\nlessdot$ 227B	$\nlessdot$ 228B	$\nlessdot$ 229B	$\nlessdot$ 22AB	$\nlessdot$ 22BB	$\nlessdot$ 22CB	$\nlessdot$ 22DB	$\nlessdot$ 22EB	$\nlessdot$ 22FB
C	$\nlessdot$ 220C	$\sqrt[4]{\quad}$ 221C	$\mathbb{F}$ 222C	$\nlessdot$ 223C	$\nlessdot$ 224C	$\nlessdot$ 225C	$\nlessdot$ 226C	$\nlessdot$ 227C	$\nlessdot$ 228C	$\nlessdot$ 229C	$\nlessdot$ 22AC	$\nlessdot$ 22BC	$\nlessdot$ 22CC	$\nlessdot$ 22DC	$\nlessdot$ 22EC	$\nlessdot$ 22FC
D	$\nlessdot$ 220D	$\alpha$ 221D	$\mathbb{F}$ 222D	$\nlessdot$ 223D	$\nlessdot$ 224D	$\nlessdot$ 225D	$\nlessdot$ 226D	$\nlessdot$ 227D	$\nlessdot$ 228D	$\nlessdot$ 229D	$\nlessdot$ 22AD	$\nlessdot$ 22BD	$\nlessdot$ 22CD	$\nlessdot$ 22DD	$\nlessdot$ 22ED	$\nlessdot$ 22FD
E	$\blacksquare$ 220E	$\infty$ 221E	$\mathbb{F}$ 222E	$\nlessdot$ 223E	$\nlessdot$ 224E	$\nlessdot$ 225E	$\nlessdot$ 226E	$\nlessdot$ 227E	$\nlessdot$ 228E	$\nlessdot$ 229E	$\nlessdot$ 22AE	$\nlessdot$ 22BE	$\nlessdot$ 22CE	$\nlessdot$ 22DE	$\nlessdot$ 22EE	$\nlessdot$ 22FE
F	$\nlessdot$ 220F	$\nlessdot$ 221F	$\mathbb{F}$ 222F	$\nlessdot$ 223F	$\nlessdot$ 224F	$\nlessdot$ 225F	$\nlessdot$ 226F	$\nlessdot$ 227F	$\nlessdot$ 228F	$\nlessdot$ 229F	$\nlessdot$ 22AF	$\nlessdot$ 22BF	$\nlessdot$ 22CF	$\nlessdot$ 22DF	$\nlessdot$ 22EF	$\nlessdot$ 22FF

# Speicherbelegung: ein Beispiel

- Maschinenbefehl: SR C 0



- ganze Zahl: 1728113477

intern dargestellt als:

01100111 00000000 11101011 01000101

- reelle Zahl:

$$2^{-50} * (1 + 2^{-8} + 2^{-9} + 2^{-10} + 2^{-12} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-21} + 2^{-23})$$

intern dargestellt als:

01100111 00000000 11101011 01000101

- 4 Zeichen (erw.) ASCII: g NUL δ E

intern dargestellt als:

01100111 00000000 11101011 01000101

Welcher Inhalt wird denn nun wirklich durch das 4-Byte-Wort  
01100111 00000000 11101011 01000101 repräsentiert ?

Steuerwerk  
(Prozessor)