Zahlen- und Zeichendarstellung

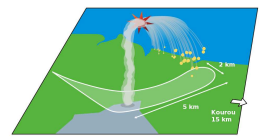
Inhalt:

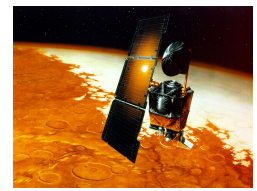
- Zahlensysteme
- ASCII
- Zahlentypen
- Genauigkeit

Motivation — I

Absturz einer Ariane 5 mit vier Satelliten am 04.06.1996 wegen fehlerhafter Steuersoftware: Bei der Umwandlung einer 64–Bit–Gleitkommazahl in eine 16–Bit–Ganzzahl kam es zu einem arithmetischen Überlauf. Die Software wurde ungeprüft von der Ariane 4 kopiert, diese war aber schwächer und langsamer. Schaden knapp 300 Mio. Euro.

1999 verglühte der Mars Climate Orbiter in der Marsatmosphäre. Bei Parametern wurden unterschiedliche physikalische Einheiten verwendet.





Motivation — II

```
#include <stdio.h>
int main (void)
{
    float f1 = 123456789.0;
    float f2;
    f2 = f1 + 1.0;
    f2 = f2 - f1;
    printf("%e\n", f2);
    return 0;
}

oergel@pool07:~> ./a.out
0.000000e+00
oergel@pool07:~>
```


Informationsdarstellung in Digitalrechnern

Das **Bit** — die kleinste Informationseinheit kann zwei Zustände annehmen, die wir mit **0** und **1** bezeichnen.
Das **Byte** besteht aus 8 Bit und ist heute i.d.R. die kleinste adressierbare Speichereinheit. Es kann $2^8 = 256$ verschiedene Zustände (Bitmuster) annehmen.

Bitnr.	7	6	5	4	3	2	1	0
Byte	1	1	0	0	1	1	0	1
8 Bit	MSB							LSB

MSB — *most significant bit* **LSB** — *least significant bit*

Zahlensysteme

Dual
Basis: 2
Werte: 0 und 1

Dezimal
Basis: 10
Werte: 0 bis 9

Hexadezimal
Basis: 16
Werte: 0 – 9, A – F

$$
\begin{aligned}
0110\ 1010_2 &= 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\
106_{10} &= 1 \cdot 10^2 + 0 \cdot 10^1 + 6 \cdot 10^0 \\
6A_{16} &= 6 \cdot 16^1 + 10 \cdot 16^0
\end{aligned}
$$

$$0110\ 1010_2 = 106_{10} = 6A_{16}$$

Ein Byte wird durch genau zwei Hexadezimalziffern dargestellt!

Darstellung eines Wertes in verschiedenen Zahlensystemen

dezimal	binär	hex	dezimal	binär	hex
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Darstellung von Zeichen

ASCII — *American Standard Code of Information Interchange*

Code	...0	...1	...2	...	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	...	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	...	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	...	’	(	)	*	+	,	-	.	/
3...	0	1	2	...	7	8	9	:	;	¡	=	¿	?
4...	@	A	B	...	G	H	I	J	K	L	M	N	O
5...	P	Q	R	...	W	X	Y	Z	[	\	]	^	_
6...	‘	a	b	...	g	h	i	j	k	l	m	n	o
7...	p	q	r	...	w	x	y	z	{	—	}		DEL

Darstellung von Zahlen im Rechner

Natürliche Zahlen

Ein **Byte** kann einen von 256 möglichen Zuständen annehmen. Natürliche Zahlen haben immer ein positives Vorzeichen, d.h. in einem Byte können die Zahlen $0000\ 0000_2$ – $1111\ 1111_2$ dargestellt werden.

Anzahl von Bytes	1	2	3	4
größte nat. Zahl	255_{10}	$65\ 535_{10}$	$16\ 777\ 215_{10}$	$\approx 4.3 \cdot 10^9$

Ganze Zahlen — Integer

Darstellung mit Vorzeichenbit

- Reservierung eines Bits für das Vorzeichen. Bit 7 wird mit 1 belegt, wenn die Zahl negativ ist.
- Darstellungsbereich: -127_{10} ... $+127_{10}$
- Problem: -0 und $+0$ $\Rightarrow$ Zweierkomplement

Ganze Zahlen — Integer

Darstellung im Zweierkomplement

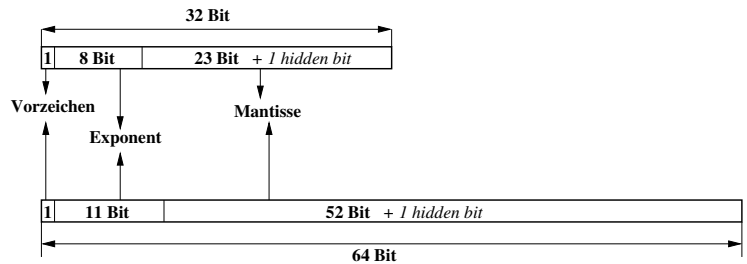
- Negative Zahlen haben auch im MSB eine 1, aber es gibt nur eine Darstellung für die 0
- Bildung einer negativen Zahl im Zweierkomplement
  1. Negieren aller Bits der positiven Zahl
  2. Addition von 1
- Beispiel: $+106_{10} = 0110\ 1010_2$
  $-106_{10} = 1001\ 0110_2$
  1. 0110 1010 *negieren* $\rightarrow$ 1001 0101
  2. 1001 0101 + 0000 0001 = 1001 0110

Reelle Zahlen — Real

32 Bit–Darstellung

- **Vorzeichen:** das *most signifikant bit* (Nr. 31) dient zur Darstellung des Vorzeichens
- **Mantisse:** 24 Bit (inkl. einem *hidden bit*)
  - normalisiert auf 1.0_2 – $1.11\ldots1_2$
  - Dualbrüche: $1.10_2 = 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} = 1.50_{10}$
    $1.01_2 = 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 1.25_{10}$
- **Exponent:** 8 Bit

Real nach IEEE (PC) — 32 Bit $\Leftrightarrow$ 64 Bit



$\Rightarrow$ **Beachte:** Unterschiedliche Bedeutung der Bits bei verschiedenen Speicherformaten möglich!

Genauigkeit der internen Zahlendarstellung

Kreiszahl π in unterschiedlicher Darstellung:

$$0.0314159 \cdot 10^{+2} \qquad 0.3141593 \cdot 10^{+1} \qquad 3.1415927 \cdot 10^{0}$$

- Die letzte Darstellungsform bietet das höchste Maß an Genauigkeit
- Mantisse wird stets auf $1.0\ldots$ bis $9.9\ldots$ normalisiert
- Anzahl der Nachkommastellen ist begrenzt $\Rightarrow$ ungenaue Werte

Genauigkeit der internen Zahlendarstellung

32 Bit Darstellung

24 Bit Mantisse bedeutet:
Nach 1.0_2 folgt $1.0\ldots01_2$ mit der 1 an Position 2^{-23}

$\rightarrow \varepsilon = 2^{-23} \approx 1.2 \cdot 10^{-7}$

64 Bit Darstellung

53 Bit Mantisse bedeutet:
Nach 1.0_2 folgt $1.0\ldots01_2$ mit der 1 an Position 2^{-52}

$\rightarrow \varepsilon = 2^{-52} \approx 2.2 \cdot 10^{-16}$

BCD–Kodierung

- Binär Dezimal Code — *binary coded decimal* wird direkt von vielen Prozessorbefehlssätzen unterstützt

Dezimal-ziffer	BCD	Dezimal-ziffer	BCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

- Beispiel: $145\ 933\ 911_{10} \rightarrow$ 5 Byte Speicherbedarf

0	1	4	5	9	3	3	9	1	1
0000	0001	0100	0101	1001	0011	0011	1001	0001	0001

Schlußfolgerung

Für die Auswertung einer Bitfolge im Arbeitsspeicher ist es aufgrund der unterschiedlichen internen Darstellungen von Bedeutung, wie ihr Inhalt zu interpretieren ist.

Beispiel von 32 Bit Speicherinhalt:

0010 0001 0100 1000 0011 1100 0111 1000

als Fließkommazahl	$\rightarrow$	$+6.78... \cdot 10^{-19}$
als ganze Zahl	$\rightarrow$	$558\ 382\ 200_{10}$
als Zeichenkette	$\rightarrow$	'!H¡x'