Darstellung Reeller Zahlen im Dezimalsystem

Reelle Zahlen entstehen entweder, wenn man ganze Zahlen teilt (rationale Zahlen) oder als Grenzwert eines Limesprozesses.

$$\frac{3}{4} = 0{,}75 \quad \text{oder} \quad \frac{1}{3} = 0{,}3333333333333333 \ldots$$

Im Englischen wird statt des Kommas meist ein Punkt verwendet: **0.75** oder **0.33** daher verwenden die meisten Programmiersprache den Dezimalpunkt und nennen die Zahlen dann Gleitpunktzahlen.

Bei rationalen Zahlen bricht die Folge der Nachkommastellen ab, $\frac{123}{12} = 10{,}25$
oder sie wird periodisch $\frac{123}{13} = 9{,}461538\ 461538\ 461538\ 461538$

Bei irrationalen Zahlen ist nur eine Näherung möglich.

$$\sqrt{2} = 1{,}4142135 \quad oder \quad \pi = 3{,}141592653589793238 \ldots$$

Von der Kreiszahl π sind heute nur die ersten 2,7 Billionenstellen bekannt.

In den letzten beiden Beispielen würde die exakte Darstellung unendlich viele Stellen benötigten – dies kann ein Rechner nicht liefern.
Je weiter fortgeschritten die PC-Entwicklung schreiten wird, desto genauer können die Berechnungen durchgeführt werden, aber im Resümee bleiben es immer Näherungen.

Exponentialdarstellung in technisch-wissenschaftlichen Anwendungsgebiet:

Sonnenmasse	= 1989100000000000000000000000000 kg	= $1{,}9891 \cdot 10^{30}$ kg
1 Lichtjahr	= 9460730472580,0 km	= $9{,}4607 \cdot 10^{12}$ km
el. Feldkonst.	= 0,000000000008854187781762 Fm^{-1}	= $8{,}8542 \cdot 10^{-12}$ Fm^{-1}
abs. Nullpunkt	= - 273,15 °C	= $-2{,}7315 \cdot 10^{2}$ °C

mögliche Varianten der Darstellung der Zahl absoluten Nullpunktes:
$-27315 \cdot 10^{-2} = -2731{,}5 \cdot 10^{-1} = -273{,}15 \cdot 10^{0} = -27{,}315 \cdot 10^{1} =$ **$-2{,}7315 \cdot 10^{2}$**

Die **normierte Darstellung** setzt das Komma bei einer Folge von Ziffern nach der ersten Ziffer ungleich 0.

Eine **Mantisse** ist eine Folge der Ziffern, beginnend mit der ersten Ziffer ungleich 0.

Darstellung im Computer:

Für eine Kommazahl müssen 3 Bestimmungsstücke gespeichert werden:

- Vorzeichen
- Exponent
- Mantisse

		Vorzeichen	Exponent	Mantisse
Sonnenmasse	$1{,}9891 \cdot 10^{30}$ kg	0	30	19891
1 Lichtjahr	$9{,}4607 \cdot 10^{12}$ km	0	12	94607
el. Feldkonst.	$8{,}8542 \cdot 10^{-12}$ Fm^{-1}	0	-12	88542
abs. Nullpunkt	$-2{,}7315 \cdot 10^{2}$ °C	1	2	27315
1/3		0	-1	

Die Speicherung der Daten hat noch das Problem, dass keine Möglichkeit für die Darstellung des Vorzeichens des Exponenten eingeplant wurde.
Als Kunstgriff werden die Exponenten individuell in den positive Bereich verschoben und diese feste „Verschiebekonstante“ = ***bias*** angegeben, dass wieder auf die richtigen Ergebnisse zurückgerechnet werden kann.

	Vorzeichen	Exponent	Mantisse	***bias***
Sonnenmasse	0	30	19891	50
1 Lichtjahr	0	12	94607	50
el. Feldkonst.	0	-12	88542	50
abs. Nullpunkt	1	2	27315	50
1/3	0	-1	3,3333	50

Das Format wäre dann:

- Vorzeichen Stelle
- Exponent Stellen
- Mantisse Stellen
- bias

Darstellung im Dezimalsystem:

$Z_{10} = 13{,}245_{(10)} = 1 \cdot 10^1 + 3 \cdot 10^0 + 2 \cdot 10^{-1} + 4 \cdot 10^{-2} + 5 \cdot 10^{-3}$

Umrechnung Dualzahl in Dezimalzahl:

$$Z_2 = 10{,}101_{(2)} = 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} =$$

$$= 1 \cdot 2 + 0 \cdot 1 + 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} =$$

$$= 2 + 0 + 0{,}5 + 0 + 0{,}125 =$$

$$= \mathbf{2{,}625}_{(10)}$$

Umrechnung Dezimalzahl in Dualzahl:

$Z_{10} = 45{,}6875_{(10)} =$ 101101,1011

① Ganze Zahl:

45 ☞ Dualzahl

45	: 2 =	22 R1
	: 2 =	11 R0
	: 2 =	5 R1
	: 2 =	2 R1
	: 2 =	1 R0
	: 2 =	0 R1

② Nachkommazahl:

0,6875 ☞ Dualzahl

0,6875	• 2 =	1,375 -> 1
0,375	• 2 =	0,75 -> 0
0,75	• 2 =	1,5 -> 1
0,5	• 2 =	1 -> 1

Standardisierte Formate des IEEE *(Institute of Electrical and Electronics Engineers)*:

	Vorzeichen	Exponent	Mantisse	bias
short real	1 Bit	8 Bit	23 Bit	127
long real	1 Bit	11 Bit	52 Bit	1023

Beispiele absoluter Nullpunkt als „short real“:

abs. Nullpunkt: **- 273,15** °C

Ganze Zahl , Nachkommazahl

Umrechnung der Dezimalzahl: **273 , 15**

in die Dualzahl: **0000 0001 0001 0001 , 0010 0110 0110 0110 0110 0110**

273	:	2	=	136	Rest	1
136	:	2	=	68	Rest	0
68	:	2	=	34	Rest	0
34	:	2	=	17	Rest	0
17	:	2	=	8	Rest	1
8	:	2	=	4	Rest	0
4	:	2	=	2	Rest	0
2	:	2	=	1	Rest	0
1	:	2	=	0	Rest	1
0	:	2	=	0	Rest	0
0	:	2	=	0	Rest	0
0	:	2	=	0	Rest	0
0	:	2	=	0	Rest	0
0	:	2	=	0	Rest	0
0	:	2	=	0	Rest	0
0	:	2	=	0	Rest	0

0,15	•	2	=	0,3	Rest	0
0,3	•	2	=	0,6	Rest	0
0,6	•	2	=	1,2	Rest	1
0,2	•	2	=	0,4	Rest	0
0,4	•	2	=	0,8	Rest	0
0,8	•	2	=	1,6	Rest	1
0,6	•	2	=	1,2	Rest	1
0,2	•	2	=	0,4	Rest	0
0,4	•	2	=	0,8	Rest	0
0,8	•	2	=	1,6	Rest	1
0,6	•	2	=	1,2	Rest	1
0,2	•	2	=	0,4	Rest	0
0,4	•	2	=	0,8	Rest	0
0,8	•	2	=	1,6	Rest	1
0,6	•	2	=	1,2	Rest	1
0,2	•	2	=	0,4	Rest	0
0,4	•	2	=	0,8	Rest	0
0,8	•	2	=	1,6	Rest	1
0,6	•	2	=	1,2	Rest	1
0,2	•	2	=	0,4	Rest	0
0,4	•	2	=	0,8	Rest	0
0,8	•	2	=	1,6	Rest	1
0,6	•	2	=	1,2	Rest	1
0,2	•	2	=	0,4	Rest	0

- 0000 0001 0001 0001 , 0010 0110 0110 0110 0110 0110

Exponentialdarstellung in technisch-wissenschaftlichen Anwendungsgebiet:

- 1   0001 0001 0010 0110 0110 0110 0110 0110 • 2

Weil es sich um Binärzahlen handelt, steht nach der Normierung vor dem Komma (Dezimalpunkt) immer eine **1**.

Daher wird diese **1** nicht gespeichert und daher meist auch als „**hidden bit**“ bezeichnet.

Der Exponent 8 wird mit *bias 127* zu $135_{(10)}$ ⇨ $10000111_{(2)}$

	Vorzeichen	**Exponent**	**Mantisse**	*bias*
short real	**1 Bit**	**8 Bit**	**23 Bit**	*127*

1   10000111   0001000100100110011001

Ungenauigkeiten:

Durch die notwendige Beschränkung der Mantisse treten bei der Umrechnung in Gleitkommazahlen Rundungsfehler auf.

Beispiel: 0,1 (10) in **short real** (float in Java)

0 , 0001 1001 1001 1001 1001 1001 (2) x2*-4

Der Exponent -4 wird mit *bias 127* zu 123 ⇨ 011111011 (2)

Vorzeichen	Exponent	Mantisse
1 Bit	8 Bit	23 Bit
0	01111011	10011001100110011001100

In der Realität erzeugt die CPU hier einen Fehler im letzten Bit, weil sie intern mit einem 80-Bit-Leitpunktformat rechnet und somit dann die letzte hier angezeigte Stelle rundet.

0 01111011 10011001100110011001101

Algebraische Rechenoperationen:

Zusätzliche Ungenauigkeiten entstehen bei algebraischen Rechenoperationen.

Beispiel Java: 0.1 + 0.2 = 0.30000000000000004

Diese Ungenauigkeit an der 17. Nachkommastelle ist meist unerheblich, in einem Taschenrechner würde man den Fehler nicht bemerken, weil die Anzeige nicht so viele Stellen hat.

Grundsätzlich sind alle Gleitkommazahlen und besonders nach Rechenoperationen kritisch zu betrachte.
Besonders deutlich wird dies beim Vergleich zweier Gleitkommazahlen.
Hier muss man einkalkulieren, dass z. B. die Antwort der Vergleichsabfrage von 0,3 zur Summe aus 0,1 + 0,2 ⇨ „FALSCH“ (*false*) sein kann.

Übersicht der wichtigsten Gleitpunktformate

Bit	Vorzeichen	Exponent	Mantisse	gültige Dezimalstellen	Bereich	
					von	bis
32	1 Bit	8 Bit	23 Bit	≈ 7	$\pm 1 \cdot 10^{-38}$	$\pm 3 \cdot 10^{38}$
64	1 Bit	11 Bit	52 Bit	≈ 15	$\pm 1 \cdot 10^{-308}$	$\pm 3 \cdot 10^{308}$
80	1 Bit	15 Bit	64 Bit	≈ 19	$\pm 1 \cdot 10^{-4932}$	$\pm 3 \cdot 10^{4932}$

Real-Zahlenbereich in Programmiersprachen

Bereich		Bytes	Delphi	Java
von	bis			
$\pm 2{,}9 \cdot 10^{-39}$	$\pm 1{,}7 \cdot 10^{38}$	6	Real	
$\pm 1{,}5 \cdot 10^{-45}$	$\pm 3{,}4 \cdot 10^{38}$	4	Single	float
$\pm 5{,}0 \cdot 10^{-324}$	$\pm 1{,}7 \cdot 10^{308}$	8	Double	double
$\pm 3{,}4 \cdot 10^{-4951}$	$\pm 1{,}1 \cdot 10^{4932}$	10	Extended	