Floating-Point

- Neben ganzen Zahlen sind **Gleitkommazahlen** wichtiger elementarer Zahlentyp
- Im Englischen Dezimalpunkt statt Komma, daher **Floating-Point**
- Approximation der reellen Zahlen
  - Aber nur **beschränkte Genauigkeit**
    - Enthalten auch spezielle „Zahlen“, siehe später
- **Darstellung** einer Floating-Point-Zahl *z*

$$z = (-1)^{v} \cdot \text{Mantisse} \cdot 2^{\text{Exponent}}$$

Floating-Point

- Floating-Point-Zahlen nach **IEEE 754-1985**
  - 32 Bit float
  - 64 Bit double

float	v	30 - Exponent - 23	22 - Mantisse - 0
	1 Bit	8 Bit	23 Bit
double	v	62 - Exponent - 52	51 - Mantisse - 0
	1 Bit	11 Bit	52 Bit

- Float
  - Exponententeil von 2^-126 bis 2^127 entspricht etwa 10^-38 bis 10^38
  - Mantisse mit 7 Stellen Genauigkeit
- Double
  - Exponententeil 10^308
  - Mantisse mit 16 Stellen Genauigkeit

Floating-Point

- Spezielle Floating-Point-„Zahlen“:Es gibt Bitmuster für
  - $+\infty$ (positiv Unendlich)
  - $-\infty$ (negativ Unendlich)
  - NaN („Not a Number“)
    - Beispielsweise als Ergenis für
      - $0/0$
      - $+\infty + -\infty$
    - Aber: $+\infty + 5 = +\infty$

Floating-Point

- Im Gegensatz zur Ganzzahlarithmetik kann es bei Floating-Point-Operationen zu **Rundungsfehlern** kommen
  - Multiplikationen:
    erzeugt längere Mantissen, die wieder auf Standardformat gerundet werden müssen
  - Addition:
    eine Mantisse muss so verschoben werden, dass beide Zahlen mit dem gleichen (dem größeren) Exponenten dargestellt sind
    Einige und im Extremfall alle Bits der Mantisse eines Summanden können aus dem Darstellungsbereich herausfallen

Floating-Point

- Beispiele:
  - Der Einfachheit halber Dezimal
  - Mantisse 3 Stellen
- Multiplikation

1.34e0*3.45e2=4.623e2→
4.62e2

- Addition

1.34e0+3.45e2

0.0134e2
3.4500e2

Floating-Point

- Bei längeren Berechnungen können sich diese Rundungsfehler schnell akkumulieren
- Insbesondere wenn sowohl sehr kleine als auch sehr große Zahlen
- Verschiedene Berechnungsverfahren für dieselbe Funktion kann zu verschiedenen Ergebnissen führen

Floating-Point

- Bei Konversion von Dezimal in Dual kann es bei Floating-Point-Zahlen zu **Konversionsfehlern** kommen
- Endlicher Dezimalbruch kann unendlicher Dualbruch sein
  - Beispiel: $0{,}1_{10} = 0{,}0001100110011001\ldots_2$

- Algorithmus zur Konversion

Konversion von Gleitkommazahlen dezimal $\rightarrow$ dual

1. Multipliziere z mit 2 und nenne das Ergebnis wieder z.
2. Die Vorkommastelle von z ist nun die nächste duale Nachkommastelle.
3. Falls $z \geqslant 1$, so ziehe 1 von z ab und nenne das Ergebnis wieder z.
4. Weiter bei 1. $\square$

Beispiel 2.5.3. Sei wieder $z = 0,3_{10}$. Wir erhalten für z der Reihe nach die Werte $0,6$; $1,2$; $0,4$; $0,8$; $1,6$; $1,2 \ldots$ und daher $d = 0,010011\ldots$. Offensichtlich wiederholt sich das Ganze jetzt periodisch, es ist also $0,3_{10} = 0,0\overline{1001}_2$. ❖