

Gleitkommazahlen



Die Typen float und double

Variablen

```
float x, y;    // 32 Bit groß  
double z;     // 64 Bit groß
```

Konstanten

```
3.14          // Typ double  
3.14f         // Typ float  
3.14E0        //  $3.14 * 10^0$   
0.314E1       //  $0.314 * 10^1$   
31.4E-1       //  $31.4 * 10^{-1}$   
.23  
1.E2          // 100
```

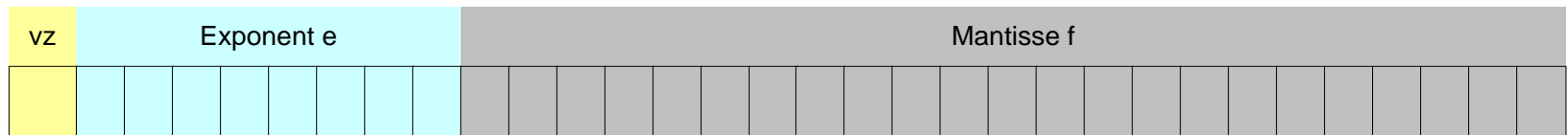
Syntax der Gleitkommakonstanten

FloatConstant	= [Digits] "." [Digits] [Exponent] [FloatSuffix].
Digits	= Digit {Digit}.
Exponent	= ("e" "E") ["+" "-"] Digits.
FloatSuffix	= "f" "F" "d" "D".



Gleitkommazahlen: Zahlendarstellung

- Darstellung von Gleitkommazahlen nach dem IEEE-Standard 754-1985
- Kodierung jeder float-Zahl binär in
 - 1 Bit Vorzeichen
 - 8 Bit Exponent
 - 23 Bit Mantisse



- Wert ergibt sich aus

$$(-1)^{vz} \times 2^{e-127} \times 1.f$$

Kodierung Dezimal → Float

- Gegeben sei eine Dezimalzahl x
 - Gesucht die binäre Darstellung als float
1. Bestimme den größten Exponenten d für den gilt $2^d \leq x$
 2. Setze $f = (x - 2^d) / 2^d$
Damit gilt $x = (1 + f) \times 2^d$ mit $0 \leq f < 1$
 3. Berechne Exponenten $e = d + 127$ und kodiere e in 11 Dualzahl-Bits
siehe Übungsaufgabe
 4. Kodiere f in 23 Dualzahl-Bits folgend

```
for (i = 0; i < 23; i++) {  
    f = f * 2.0;  
    if (f >= 1.0) {  
        print ('1');  
        f = f - 1.0;  
    } else {  
        print ('0');  
    }  
}
```



Beispiel: Berechnung der harmonischen Reihe

$$\text{sum} = 1/1 + 1/2 + 1/3 + \dots + 1/n$$

```
class HarmonicSequence {  
  
    public static void main (String[] arg) {  
        float sum = 0;  
        int n = In.readInt();  
        for (int i = n; i > 0; i--)  
            sum += 1f / i  
        Out.println("sum = " + sum);  
    }  
}
```

Was würden statt `1f / i` folgende Ausdrücke liefern?

<code>1 / i</code>	0 (weil ganzzahlige Division)
<code>1.0 / i</code>	einen double-Wert



Zuweisungskompatibilität

double \supseteq float \supseteq long \supseteq int \supseteq short \supseteq byte

```
float f; int i;  
f = i; // erlaubt  
i = f; // verboten  
i = (int)f; // erlaubt: schneidet Nachkommastellen ab; falls zu groß: maxint, minint  
f = 1.0; // verboten, weil 1.0 vom Typ double ist
```

Erlaubte Operationen

- Arithmetische Operationen (+, -, *, /)
- Vergleiche (==, !=, <, <=, >, >=)

Achtung: Gleitkommazahlen sollte man nicht auf Gleichheit prüfen



Typen von Gleitkommaausdrücken

Der "kleinere" Operandentyp wird in der "größeren" konvertiert, zumindest aber in int.

```
double  $\supseteq$  float  $\supseteq$  long  $\supseteq$  int  $\supseteq$  short  $\supseteq$  byte
```

```
double d; float f; int i; short s;
```

```
...
```

```
d + i    // double
```

```
f + i    // float
```

```
s + s    // int
```

Ein- / Ausgabe von Gleitkommazahlen

```
double d = In.readDouble();
```

```
float f = 3.14f;
```

```
// es gibt in der Klasse In kein readFloat
```

```
Out.println("d = " + d + ", f = " + f);
```



Operatoren für Gleitkommazahlen

Addition: $+$: Gleitkomma \times Gleitkomma \rightarrow Gleitkomma

Subtraktion: $-$: Gleitkomma \times Gleitkomma \rightarrow Gleitkomma

Multiplikation: $*$: Gleitkomma \times Gleitkomma \rightarrow Gleitkomma

Division: $/$: Gleitkomma \times Gleitkomma \rightarrow Gleitkomma

Inkrement um 1.0: $++$: Gleitkomma \rightarrow Gleitkomma

Dekrement um 1.0: $--$: Gleitkomma \rightarrow Gleitkomma



- Mit `Math` steht eine umfangreiche Funktionsbibliothek für mathematische Operationen zur Verfügung.
- Diese Operationen ruft man folgend auf

`Math.operator(argumente)`

sie liefern den entsprechenden Wert und können in Ausdrücken verwendet werden.

Beispiel: Sinusberechnung

```
double x;  
double sinusX;  
  
x = 30;  
sinusX = Math.sin(x);
```



Funktionen in Math (Auszug)

- `abs(double a)` Absolutbetrag von a
- `ceil(double a)` Kleinste Ganzzahl $\geq a$
- `floor(double a)` Größte Ganzzahl $\leq a$
- `max(double a, double b)` Maximum von a und b
- `min(double a, double b)` Minimum von a und b
- `log(double a)` natürlicher Logarithmus von a
- `pow(double a, double b)` a^b
- `round(double a)` Rundung auf `long`
- `cos(double a)` Kosinus
- `sin(double a)` Sinus
- `tan(double a)` Tangens
- `atan(double a)` Arkustangens
- ... und viele mehr

