Einführung in die Informatik
Inf, SAT, Wing, Mi

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

1. Grundlagen der Informatik

Zahlensysteme, Nachkommazahlen, Single-Format

2. Rechnerarchitekturen (Hard- und Softwaresysteme)

Rechnerarchitekturen, Betriebssysteme

3. Programmierung

Sprachen, Datentypen, Datenstrukturen, Programmiermodelle, Compiler, Programmierparadigmen, Domänenspezifische Sprachen, Algorithmen

4. Verteilte Systeme (Netzwerke)

5. Themenfelder der Informatik

Datenbanken, Big Data, Multimediaverarbeitung, Software Engineering, KI, Human Computing Interface, Robotics, VR/AR

Literatur

- Grundlagen und Konzepte der Informatik, Pearson Studium 2007

  Helmut Herold, Bruno Lurz, Jürgen Wohlrab

  ISBN-10: 3827373050 ISBN-13: 978-3827373052

- Gumm, H.P.; Sommer, M.: Einführung in die Informatik,
  8. Auflage, Oldenbourg 2008,

  ISBN-10: 3486587242

  ISBN-13: 978-3486587241

Vorlesung

- Vorlesung per Powerpoint
- Beispiele an der Tafel
- Beispiele auf der Homepage
- Musterklausur am Ende der Vorlesung,

  Ende Dezember

Zahlensysteme:

Neben dem Dezimalsystem sind alle weiteren ***Zahlensysteme*** die vor allem in der Informatik verwendet werden, sogenannte ***Stellenwertsysteme***.

- **Definition - Stellenwertsystem:**
- Jede reelle Zahl wird durch eine Folge von Ziffern beschrieben:
- Wie eine Ziffer zum Wert der Zahl beiträgt, hängt von ihrer Position bezüglich des Kommas ab. Sie wird dazu mit (B = Basis des Zahlensystems multipliziert. Für den Wert der Zahl ergibt sich damit:

$$X = \pm(Z_m B^m + Z_{m-1} B^{m-1} + \ldots + Z_1 B^1 + Z_0 B^0 + Z_{-1} B^{-1} + Z_{-2} B^{-2} \ldots)$$

$$X = \pm \sum_{\mu=-\infty}^{+\infty} Z_\mu B^\mu \quad \approx \pm \sum_{\mu=-m}^{+n} Z_\mu B^\mu$$

$$123{,}4 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1}$$

$$123{,}4 = 1 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 + 4 \cdot 8^{-1}$$

Zahlensysteme:

Bezüglich der Basis B und der zugelassenen Ziffern gilt folgende Übersicht, die in der Tabelle zusammengefasst ist:

Zahlensystem	**Zahlenbasis**	**Ziffern**	**Beispiel**
Dualsystem	B=2	0, 1	11111001100
Fünfersystem	B=5	0, 1, 2, 3, 4	30441
Siebenersystem	B=7	0, 1, 2, 3, 4, 5, 6	5551
Oktalsystem	B=8	0, 1, 2, 3, 4, 5, 6, 7	3714
Dezimalsystem	B=10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	1996
Hexadezimalsystem	B=16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	7CC

Zahlensysteme:

Besteht die Gefahr der Verwechselung, wird das benutzte Stellensystem durch Zusätze markiert. Folgende Notationsformen sind üblich:

Dualsystem:	11111001100_2
Oktalsystem:	3714_8
Dezimalsystem:	1996_{10}
Hexadezimalsystem:	$7CC_{16}$

Beispiele:

Dezimal-System	Dual-System
5	101
7	111
15	1111
8	?
12	?
?	1110
?	00011
?	11001

Beispiele:

Dezimal-System	Oktal-System	
5	5	
7	7	
15	?	
8	8	?
?	17	
12	?	
?	100	
?	77	
?	44	

Beispiele:

Dezimal-System	Hexadezimal-System
5	5
7	?
15	?
16	?
?	17
12	?
?	11
?	100
10	?
100	?
?	201

Umrechnungstabelle für die weitere Berechnung:

Dezimal	Dual	Oktal	Hexadezimal
0	0000	0	00
1	0001	1	01
2	0010	2	02
3	0011	3	03
4	0100	4	04
5	0101	5	05
6	0110	6	06
7	0111	7	07
8	1000	10	08
9	1001	11	09
10	1010	12	0A
11	1011	13	0B
12	1100	14	0C
13	1101	15	0D
14	1110	16	0E
15	1111	17	0F

Zahlenkonvertierung

a) Berechnen der Binärzahl von 12

$13_{10} = 8 + 4 + 1$

$13_{10} = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 =$

$= 1101$ Basis 2 $= 1101_2 =$

$= 1*8 + 1*4 + 0*2 + 1*1$ Basis 2 =

....wiederholtes Ausklammern ...

$= (1*4 + 1*2 + 0*1)*2 + 1*1$

$= ((1*2 + 1*1)*2 + 0*1)*2 + 1*1$

Zahlenkonvertierung

Beispiel - Umwandlung Dezimalzahl in Dualzahl: $45_{10} = X_2$?

45 : 2	= 22	Rest:	1,	also $45 = 22 \cdot 2 + 1$
22 : 2	= 11	Rest:	0,	also $45 = (11 \cdot 2 + 0) \cdot 2 + 1$
11 : 2	= 5	Rest:	1,	also $45 =$
				$= ((5 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1$
				$= 5 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
5 : 2	= 2	Rest:	1	
2 : 2	= 1	Rest:	0	
1 : 2	= 0	Rest:	1	

Ergebnis: $45_{10} = 101101_2$

Zahlenkonvertierung

Beispiel - Umwandlung Dezimalzahl in Dualzahl: $2005_{10} = X_2$?

2005 : 2	= 1002	Rest:	1,	also $2005 = 1002 \cdot 2 + 1$
1002 : 2	= 501	Rest:	0,	also $2005 = (501 \cdot 2 + 0) \cdot 2 + 1$
501 : 2	= 250	Rest:	1,	also $2005 =$
				$= ((250 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1$
				$= 250 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
250 : 2	= 125	Rest:	0, ...	
125 : 2	= 62	Rest:	1	
62 : 2	= 31	Rest:	0	
31 : 2	= 15	Rest:	1	
15 : 2	= 7	Rest:	1	
7 : 2	= 3	Rest:	1	
3 : 2	= 1	Rest:	1	
1 : 2	= 0	Rest:	1	

Ergebnis: $2005_{10} = 0111\ 1101\ 0101_2$

Zahlenkonvertierung

Kurz:
Die berechnete Zahl im anderen Zahlensystem ergibt sich aus dem Nacheinanderschreiben der „Rest“-Werte von „unten“ nach „oben“. Oben ist immer das Komma

Restwerte schreibt man auch mittels der „modulo“-Schreibweise:
X mod 3 := Rest von X bis auf Vielfache von 3, oder allgemein:
X mod Y := R mod Y, wenn $X = n*Y + R$ mit $0 <= R < Y$ und n,X,Y,R natürliche Zahlen (n, R ist dann eindeutig bestimmt).

Beispiele:
5 mod 3 = 2
11 mod 7 =
24 mod 17 =
33 mod 16 =
63 mod 16 =
94 mod 16 =

-2 mod 16 $\Rightarrow$ $-2 + a*16 = b$, gesucht a und b, $b \in 0..15$

Zahlenkonvertierung

Beispiel: Umwandlung Dezimalzahl in Zahl zur Basis 5:

$$1996_{10} = X_5 \ ?$$

1996	:	5	=	399	Rest: 1
399	:	5	=	79	Rest: 4
79	:	5	=	15	Rest: 4
15	:	5	=	3	Rest: 0
3	:	5	=	0	Rest: 3

Ergebnis: $1996_{10} = 30441_5$

Umwandlung Dezimalzahl in Zahl zur Basis 5:

$$2010_{10} = X_5 \ ?$$

Zahlenkonvertierung

211_{10}

11010011_2

323_8

$D3_{16}$

CA_{16}

202_{10}

3302_5

FE_{16}

254_{10}

AFFE 1010111111111110_2

Zahlenkonvertierung zwischen 2, 8 und 16er-System

Berechnung des Binärsystems:
$4711_{10} = 1001001100111_2$

Damit ist es trivial, aus der Binär-Lösung sofort die weiteren Zahlen zu berechnen. Man fasst drei oder vier Bits zusammen:

a) Dual nach Hexadezimal
- Berechnen der Dualzahl
- Zusammenfassen von 4 Bits, von rechts beginnend
- $0001001001100111_2 = 1\ 0010\ 0110\ 0111_2 = 0001\ 0010\ 0110\ 0111_2$
- Alle „vier Bits“ in eine hexadezimale Zahl umwandeln
- $1\ 2\ 6\ 7_{16}$

b) Dual nach Oktal
- Berechnen der Dualzahl
- Zusammenfassen von 3 Bits, von rechts beginnend
- $0001001001100111_2 = 1\ 001\ 001\ 100\ 111_2 = 001\ 001\ 001\ 100\ 111_2$
- Alle „drei Bits“ in eine hexadezimale Zahl umwandeln
- $1\ 1\ 1\ 1\ 4\ 7_8$

2. Nachkommastellen

Zahlen mit Nachkommastellen:

$123{,}456789_{10}$

$$
\begin{aligned}
&\quad 1*100 \\
&+\ 2*10 \\
&+\ 3*1 \\
&\\
&+\ 4*0{,}1 \\
&+\ 5*0{,}01 \\
&+\ 6*0{,}001 \\
&+\ 7*0{,}0001 \\
&+\ 8*0{,}00001 \\
&+\ 9*0{,}000001
\end{aligned}
$$

Konvertierung von Zahlen mit Nachkommastellen

Bei der Konvertierung von Nachkommastellen auf eine Darstellung zu einer neuen Basis geht man gerade umgekehrt vor: die Zahl wird mit der neuen Basis multipliziert und dann die Stelle vor dem Komma als neue Ziffer „abgespalten“ (sukzessive). Von oben nach unten notiert ergeben die Vorkommastellen gerade die neue Zifferndarstellung zur neuen Basis (durch „Ausklammern“ von Potenzen von 1/b).

Beispiel:

$0{,}6875_{10}$ = Dualzahldarstellung ?

0,6875 * 2 = 1,375 vorne Abspalten: 1
0,375 * 2 = 0,750 Abspalten: 0
0,750 * 2 = 1,500 Abspalten: 1
0,500 * 2 = 1,000 Abspalten: 1

Ergebnis: $0{,}6875_{10} = 0{,}1011_2$

Beispiel:

$0{,}3_{10}$ = Dualzahldarstellung ?

Beispiele:

Dezimal-System	Dual-System
3,5	11,100000
6,25	?
?	1101,101
33,046875	?
?	101,1001
0,1	?
5,2	?

3. Ganzzahl-Multiplikation und -Division

Beispiele:

$55_{10} = 110111_2$

$55 \cdot 2 = 110_{10} = 1101110_2$

$16_{10} = 10000_2$

$16/2 = 8_{10} = 1000_2$

Multiplikation und Division

Die binäre Multiplikation kann auf das Verschieben der Bitfolge nach links zurückgeführt werden.

$55_{10} = 110111_2$

$55 \cdot 2 = 110111 \text{ SHL } 1 = 1101110_2$
$55 \cdot 2 = 110111 << 1 = 1101110_2$

Vorteil:
Einfache und schnelle Implementierung

Multiplikation und Division

Die binäre Division kann auf das Verschieben der Bitfolge nach rechts zurückgeführt werden.

110_{10} = 1101110_2
110/2 = 1101110 SHR 1 = 110111_2
110/2 = 1101110 >> 1 = 110111_2

55_{10} = 110111_2
55/2 = 110111 SHR 1 = 11011_2 = 27

Gleitkommazahlen

Dieses Zahlenformat gestattet die Beherrschung praktisch unbegrenzter Wertebereiche mit befriedigender Genauigkeit.

Grundlage der Darstellung ist, dass eine reelle Zahl im Zahlensystem mit der Basis wie folgt darstellt werden kann:

$$X = \pm m \cdot B^{\pm e}$$

e ist ganzzahlig

m ist ganzzahlig

$$e = \pm |e| \qquad\qquad m = \pm |m|$$

Das Vorzeichen der Mantisse m ist gleich dem der Zahl .

Gleitkommazahlen

Normierung:
Bei Punktverschiebung (Kommaverschiebung) in der Mantisse m um eine Stelle nach rechts (links) bleibt der Wert von erhalten, wenn gleichzeitig der Exponent e um 1 erniedrigt (erhöht) wird.

Beispiel:
$1996 = 19960 \cdot 10^{-1} = 199{,}6 \cdot 10^{1} = 1{,}996 \cdot 10^{3} = 0{,}1996 \cdot 10^{4}$

Steht, wie in den letzten beiden Darstellungsformen, die erste von Null verschiedene Mantissenziffer stets unmittelbar vor bzw. hinter dem Punkt (Komma), bezeichnet man als normiert. Damit gilt für die Gleitkommadarstellung folgende Festlegung:

$1_B \leq m < 10_B$ mit B=2,8,10,16

Gleitkommazahlen

Dabei ist zu beachten, dass die Zahl Null nicht normiert werden kann und damit eine Sonderbehandlung erfährt.

Gründe ?

Gleitkommazahlen / Floating Points

Aus dieser Darstellung folgt eine weitere Festlegung:

m = 1,*f* **Mantisse = 1,Fraction**

Für die weitere Betrachtung wird nur noch *f*, *die Fraktion* verwendet. *f* ist die um 1, reduzierte Mantisse m, mit der Festlegung, dass der Punkt (Komma) stets links von der Mantisse, dargestellt als *f* , definiert wird, aber nicht geschrieben wird.

Charakteristik:

Für den Exponenten genügt ein relativ kleiner Wertebereich (z.B.: bei B = 10), um extrem kleine bis extrem große Zahlenbeträge darstellen zu können.

Gleitkommazahlen

Die Vorzeichenbehandlung des Exponenten wird umgangen, indem mittels **Verschiebekonstante *K*** zur **Charakteristik *Ch*** übergewechselt wird und damit der Wertebereich an den Anfang des positiven Zahlenbereichs verschoben wird:

$$Ch = e + K$$

Mit dieser letzten Maßnahme wird das Vorzeichen des Exponenten in den positiven Bereich verschoben und wird nicht mehr (als negative Zahl) dargestellt.

Gleitkommazahlen

Für das Vorzeichen der Mantisse gelten die gleichen Vereinbarungen, wie für das Vorzeichen der Vorzeichenbetragzahlen:

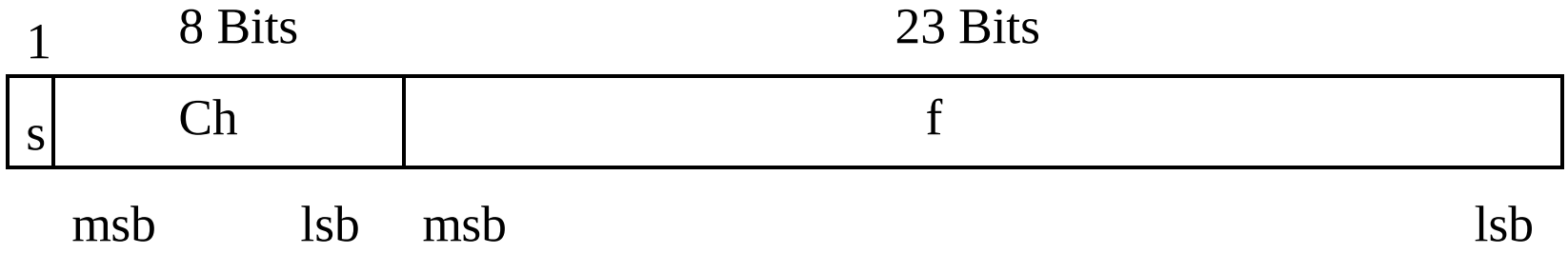
$s = 0 \quad \Rightarrow \quad z \geq 0$
$s = 1 \quad \Rightarrow \quad z < 0$

Damit sind alle Komponenten zur Zahlendarstellung eingeführt:

- das Vorzeichen der Mantisse s,
- die Abspaltung der Mantisse *f*
- die Charakteristik *Ch*

Gleitkommazahl Single:

k = 127
k = ?F



Single-Zahl hat 4 Byte =32 Bit

Falls $0 < Ch < 255$, $e = Ch - k$ $Ch = e+k$

$m = 1.f$ (Hidden-Bit)

$s = 0 \Leftrightarrow z \geq 0$

$s = 1 \Leftrightarrow z < 0$

$s=1; e = 0 ; f = 0,$ Zahl = -0.0 (signed zero)

$s=0; e = 0 ; f = 0,$ Zahl = 0.0 (unsigned zero)

$e = 0 ; f = 0,$ $Zahl = (-1)^s \cdot 2^{-126} \cdot 0.f$ (subnormal numbers)

$s=\{0,1\} ; Ch =255 ; f = 0,$ $Zahl = \pm INF$

$s=undef ; Ch =255 ; f \neq 0,$ Zahl = NAN (Not any Number)

msb = most significant bit lsb = least significant bit

Beispiel:

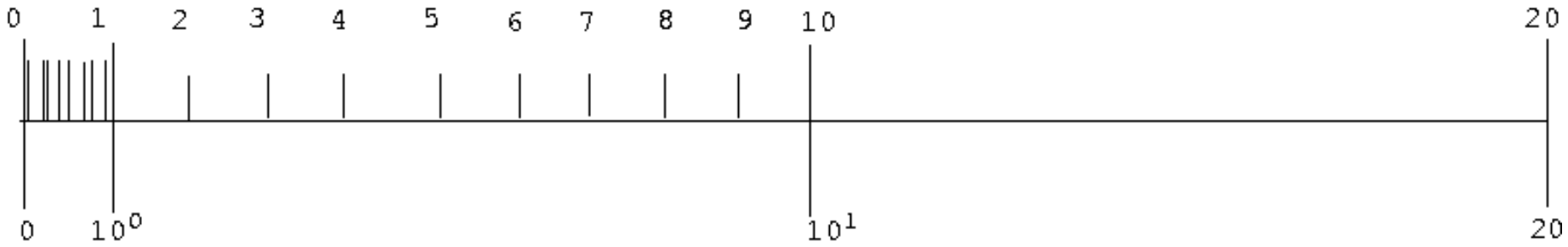
k = 127
Ch = 8 Bit

$d = -12{,}75_{10}$

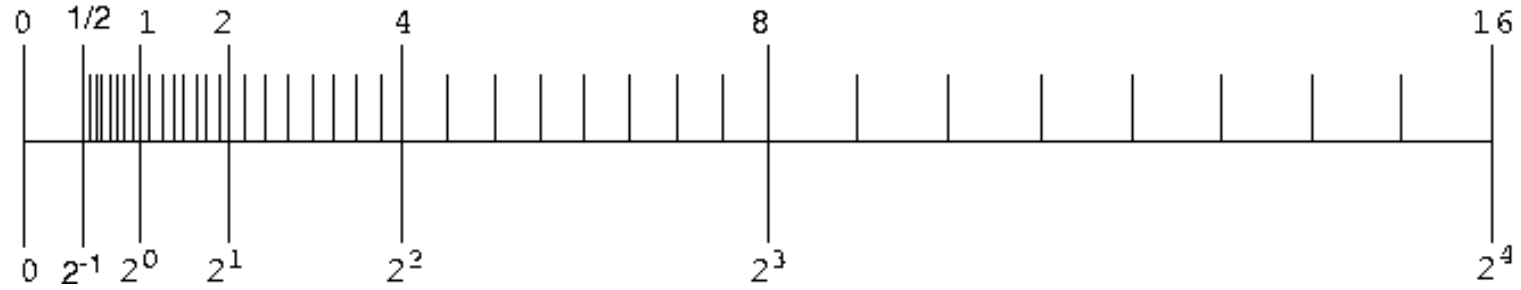
1) Umwandlung ins Binärsystem: $-1100{,}11_2$
2) Normierung auf $1{,}m \cdot 2^x$ $-1{,}100\ 11\quad 2^3$
3) Bestimmen von s, Abspalten von f und e, Ch
   s = 1
   m=1,f=1,10011
   e=3 => Ch = 127+3=130=82_{16}
4) Zusammenfassen: s Ch f
   1 1000 0010 10011 0000 0000 0000 0000 00
5) Hexadezimale Fassung
   1100 0001 0100 1100 0000 0000 0000 0000
   C 1 4 C 0 0 0 0

Zahlendarstellung

Decimal Representation:



Binary Representation:



Min. Single: 1.17549435e-38
00800000

Max. Single: 3.40282347e+38
7F7FFFFF

Zahlendarstellungen im Singleformat

x	**Nächste Zahl (x, +)**	**Lücke**
0.0	$1.4012985 \cdot 10^{-45}$	$1.4012985 \cdot 10^{-45}$
$1.1754944 \cdot 10^{-38}$	$1.1754945 \cdot 10^{-38}$	$1.4012985 \cdot 10^{-45}$
1.0	1.0000001	$1.1920929 \cdot 10^{-07}$
2.0	2.0000002	$2.3841858 \cdot 10^{-07}$
16.000000	16.000002	$1.9073486 \cdot 10^{-06}$
128.00000	128.00002	$1.5258789 \cdot 10^{-05}$
$1.0000000 \cdot 10^{+20}$	$1.0000001 \cdot 10^{+20}$	$8.7960930 \cdot 10^{+12}$
$9.9999997 \cdot 10^{+37}$	$1.0000001 \cdot 10^{+38}$	$1.0141205 \cdot 10^{+31}$

Beispiel Single: 1,0

x=1,0

Hexadezimal Darstellung: 3F80 0000

0011 1111 1000 0000 0000 0000 0000 0000

x=1,0 + x

Hexadezimal Darstellung: 3F80 0001

0011 1111 1000 0000 0000 0000 0000 0001

s=0

Ch=0111 1111 Ch=127 e=0

m=1, 000 0000 0000 0000 0000 0001

Differenz: $\Delta = 2^{-23} = 0{,}00000011920928955078125$

Beispiel Single: 2,0

x=2,0
Hexadezimal Darstellung: 4000 0000
0100 0000 0000 0000 0000 0000 0000 0000

x=2,0 + x

Hexadezimal Darstellung: 4000 0001

0 10000000 000 0000 0000 0000 0000 0001

s=0

Ch=1000 0000 Ch=128 e=1

m=1, 000 0000 0000 0000 0000 0001

X=10,00 0000 0000 0000 0000 0001
Differenz: $\Delta = 2^{-22} = 0{,}00000023841857910156 25$

Beispiele Single:

k = 127
Ch = 8 Bit

$d = 4711_{10} = 1267_{16}$ = 45 93 38 00
s = 0, Ch = 8B, f = 267

$d = 1_{10} = 1_{16}$ = 3F 80 00 00
s = 0, Ch = 7F, f = 0

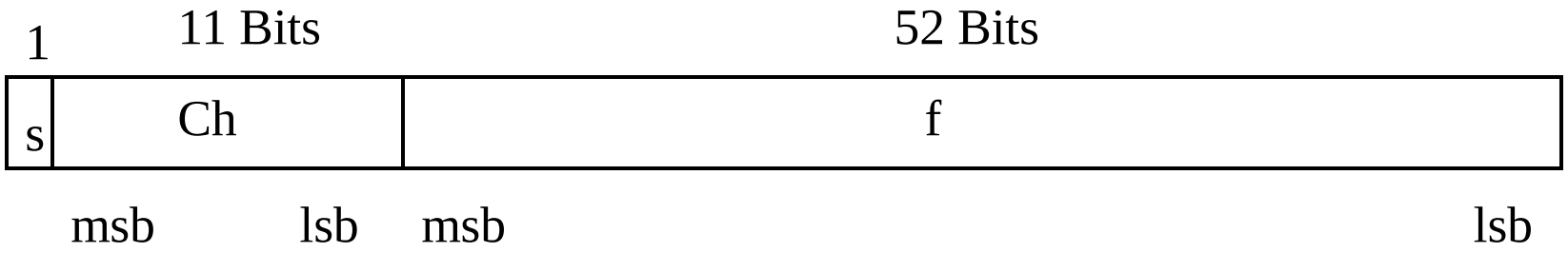
$d = -1_{10} = 1_{16}$ = BF 80 00 00
s = 1, Ch = 7F, f = 0

$d = 2_{10} = 2_{16}$ = 40 00 00 00
s = 0, Ch = 80, f = 0

Gleitkommazahl Double:

$k = 1023$

$k = 3FF_{16}$



$e = Ch - k$ $\quad Ch = e+k$ $\quad m = 1.f$

Double-Zahl hat 8 Byte =64 Bit

$0 < e < 2047$, $\quad (-1)^s \cdot 2^{-1023} \cdot 1.f$ (normal numbers)

$e = 0 \;\; ; f \neq 0$, $\quad (-1)^s \cdot 2^{-1022} \cdot 0.f$ (denormalized numbers)

$s=0 ; e = 0 \;\; ; f = 0$, $\quad$ Zahl = +0.0

$s=\{0,1\} ; Ch = 2047 ; f = 0$, $\quad$ Zahl = ±Infimum

$s=undef ; Ch = 2047 ; f \neq 0$, $\quad$ Zahl = NaN (Not any Number).

msb = most significant bit lsb = least significant bit

Beispiele Double:

k = 1023
Ch = 11 Bit

$d = 4711_{10} = 1267_{16}$ = 40B26700 00000000
s = 0, Ch = 40B, f = 267

$d = 1_{10} = 1_{16}$ = 3FF00000 00000000
s = 0, Ch = 3FF, f = 0

$d = -1_{10} = 1_{16}$ = BFF00000 00000000
s = 1, Ch = 3FF, f = 0

$d = 2_{10} = 2_{16}$ = 40000000 00000000
s = 0, Ch = 400, f = 0

Gleitkommazahl Extended:

k = 16383
k = 3FFF

Extended-Zahl hat 10 Byte =80 Bit

-23,5625

e = Ch – k

m = **j**.f

j=1 ; 0 < e < 32767, f ≠ 0	$(-1)^s \cdot 2^{-16383} \cdot 1.f$ (normal numbers)
j=0 ; e=0, f ≠ 0	$(-1)^s \cdot 2^{-16382} \cdot 0.f$ (denormalized)
s=0 ; e = 0 ; f = 0,	Zahl = +0.0
s={0,1} ; Ch = 32767 ; f = 0,	Zahl = ± Infimum
s=undef ; Ch = 32767 und f ≠ 0,	Zahl = NaN.

msb = most significant byte lsb = least significant byte

Beispiele Extended:

k = 16383=3FFF
Ch = 15 Bit

$d = 4711_{10} = 1267_{16}$ = 400B 9338 0000 0000 0000
s = 0, Ch = 400B, f = 9338 0000 0000 0000

$d = 1_{10} = 1_{16}$ = 3FFF 8000 0000 0000 0000
s = 0, Ch = 3FFF, f = 8000 0000 0000 0000

$d = -1_{10} = 1_{16}$ = BFFF 8000 0000 0000 0000
s = 1, Ch = 3FFF, f = 8000 0000 0000 0000

$d = 2_{10} = 2_{16}$ = 4000 8000 0000 0000 0000
s = 0, Ch = 4000, f = 8000 0000 0000 0000

Beispiele Extended:

k = 16383=3FFF
Ch = 15 Bit

$d = 4711_{10} = 1267_{16} = 1001001100111_2$

$s = 0,$

$1{,}001001100111_2 \cdot 2^{12}$

$e = 12$

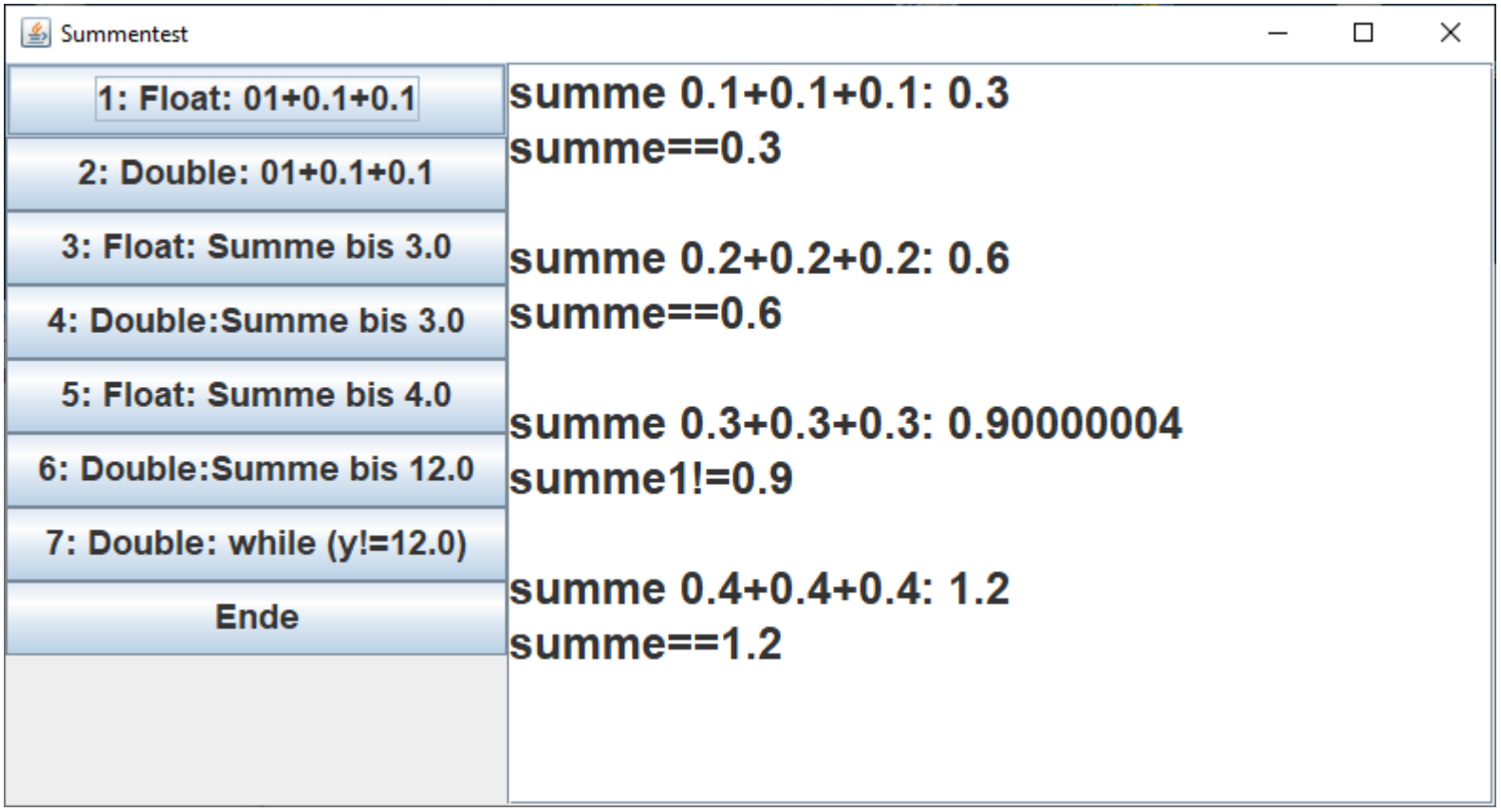
$Ch = k + e = 16383+12 = 16395 = 400B_{16}$

Tetrade f wird von links mit der 1, berechnet !!

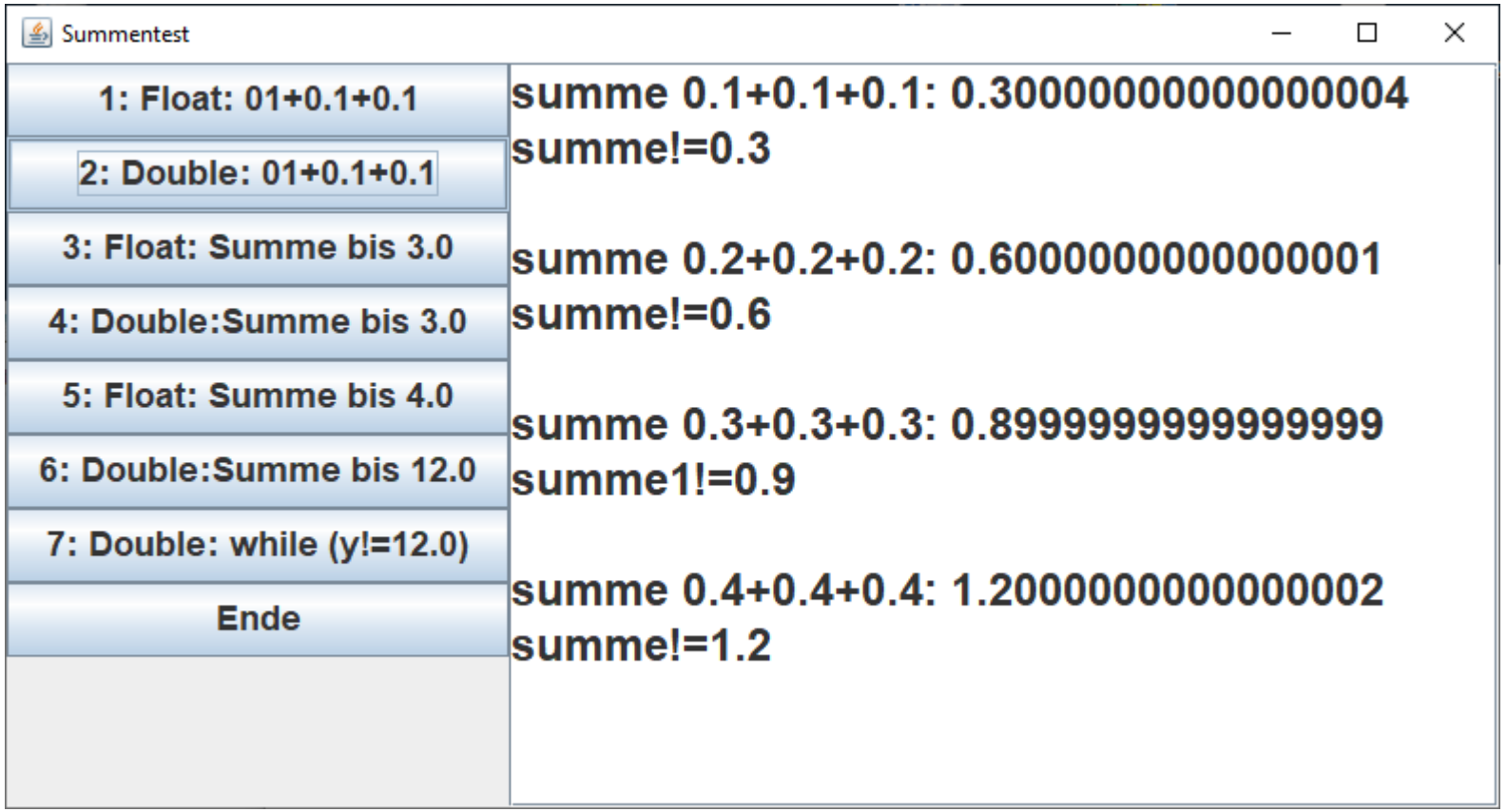
$jf = 1001\ \ 0011\ 0011\ 1_2$

$jf = 9338_{16}$

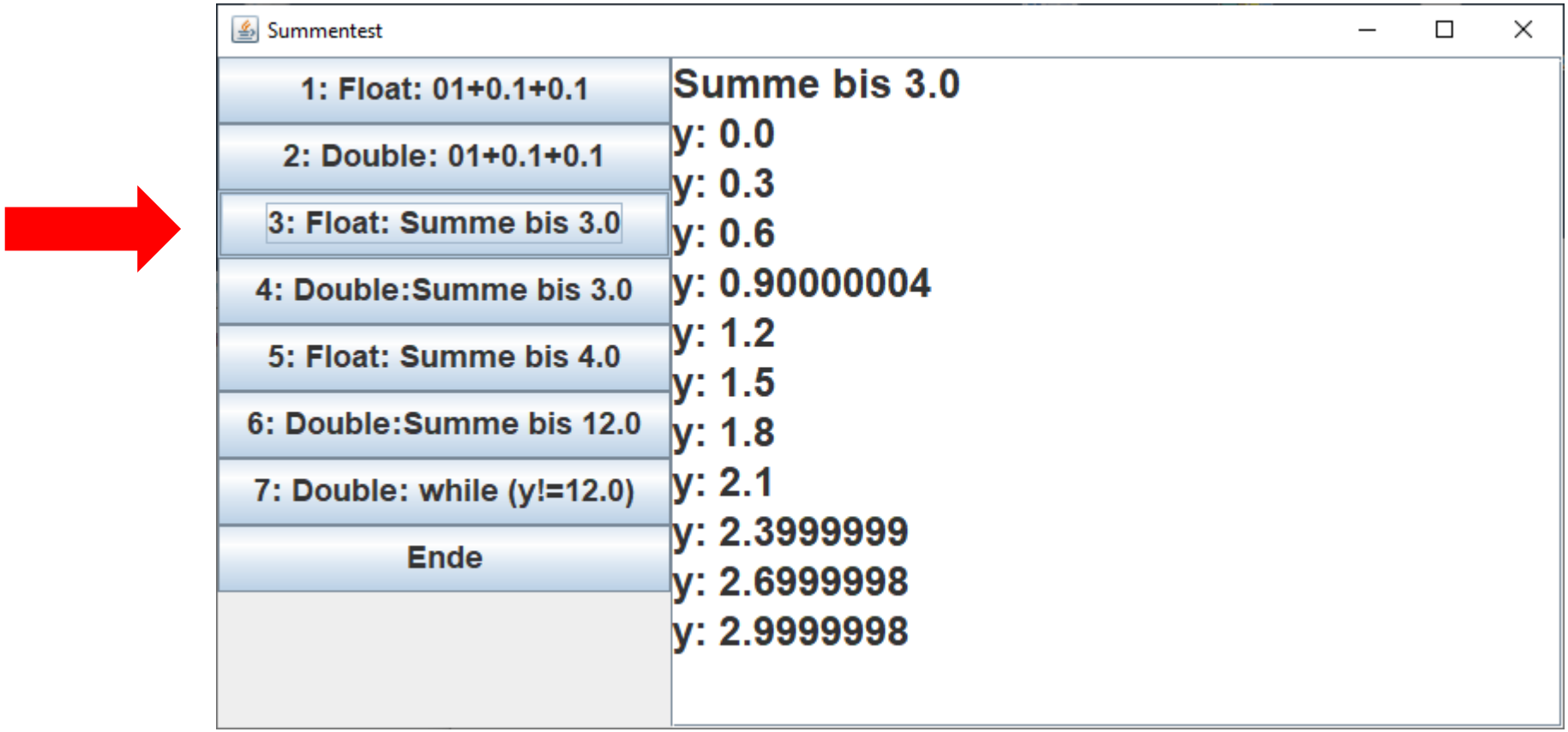
$4711_{10} = 40\ 0B\ 93\ 38\ 00\ 00\ 00\ 00\ 00\ 00$



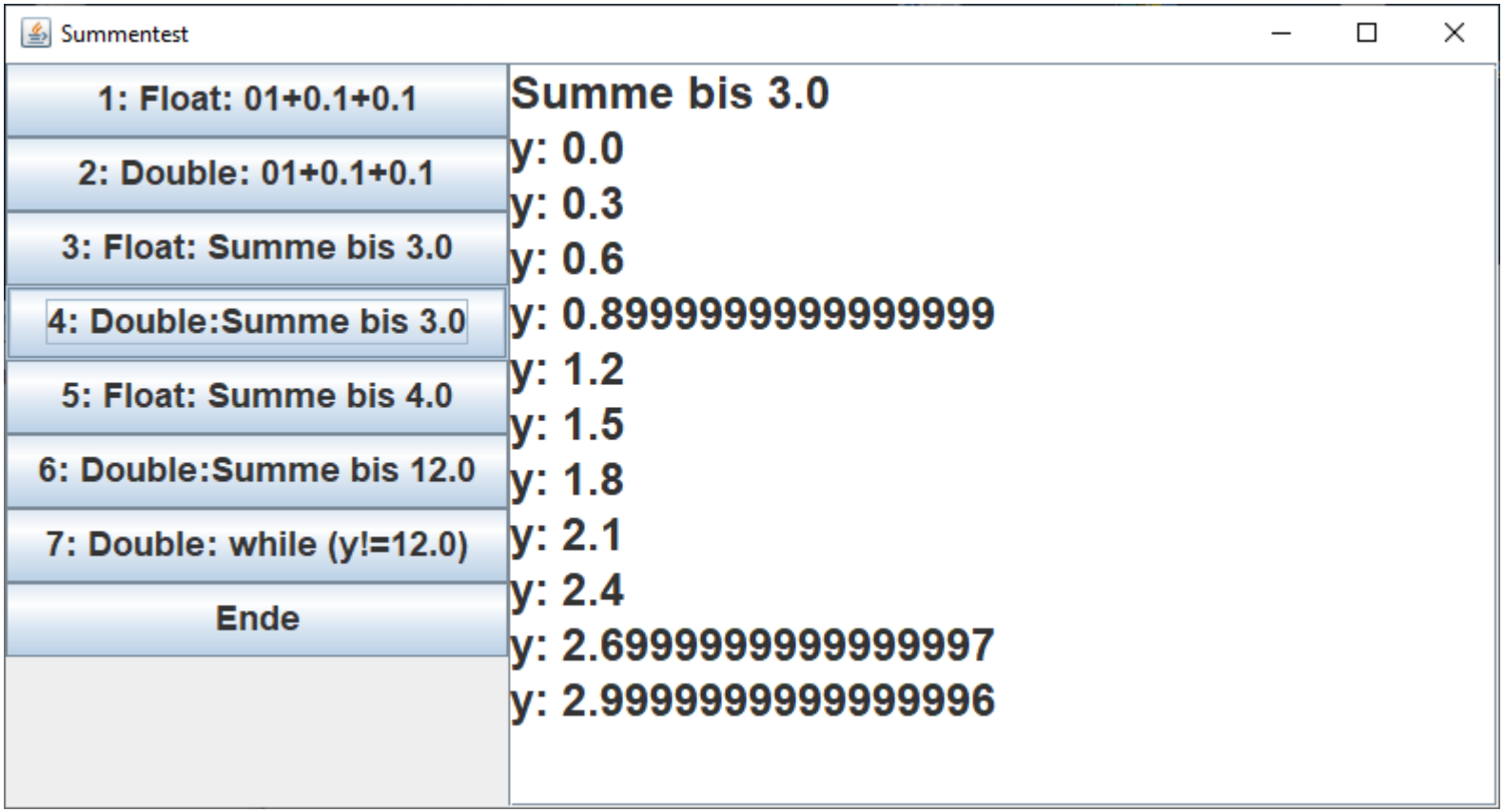
- float summe1=0.1f+0.1f+0.1f;
- float summe2=0.2f+0.2f+0.2f;
- float summe3=0.3f+0.3f+0.3f;
- float summe4=0.4f+0.4f+0.4f;



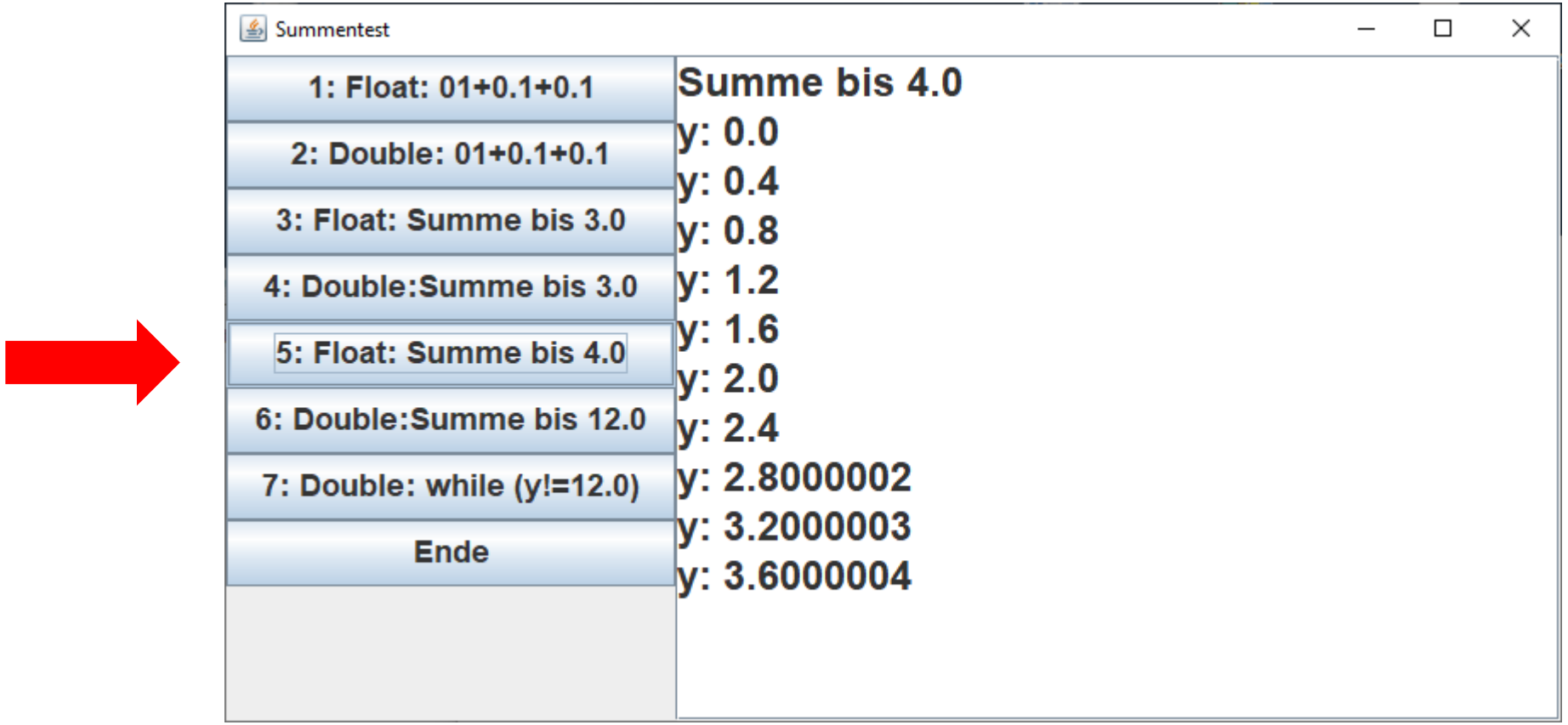
- double summe1=0.1+0.1+0.1;
- double summe2=0.2+0.2+0.2;
- double summe3=0.3+0.3+0.3;
- double summe4=0.4+0.4+0.4;



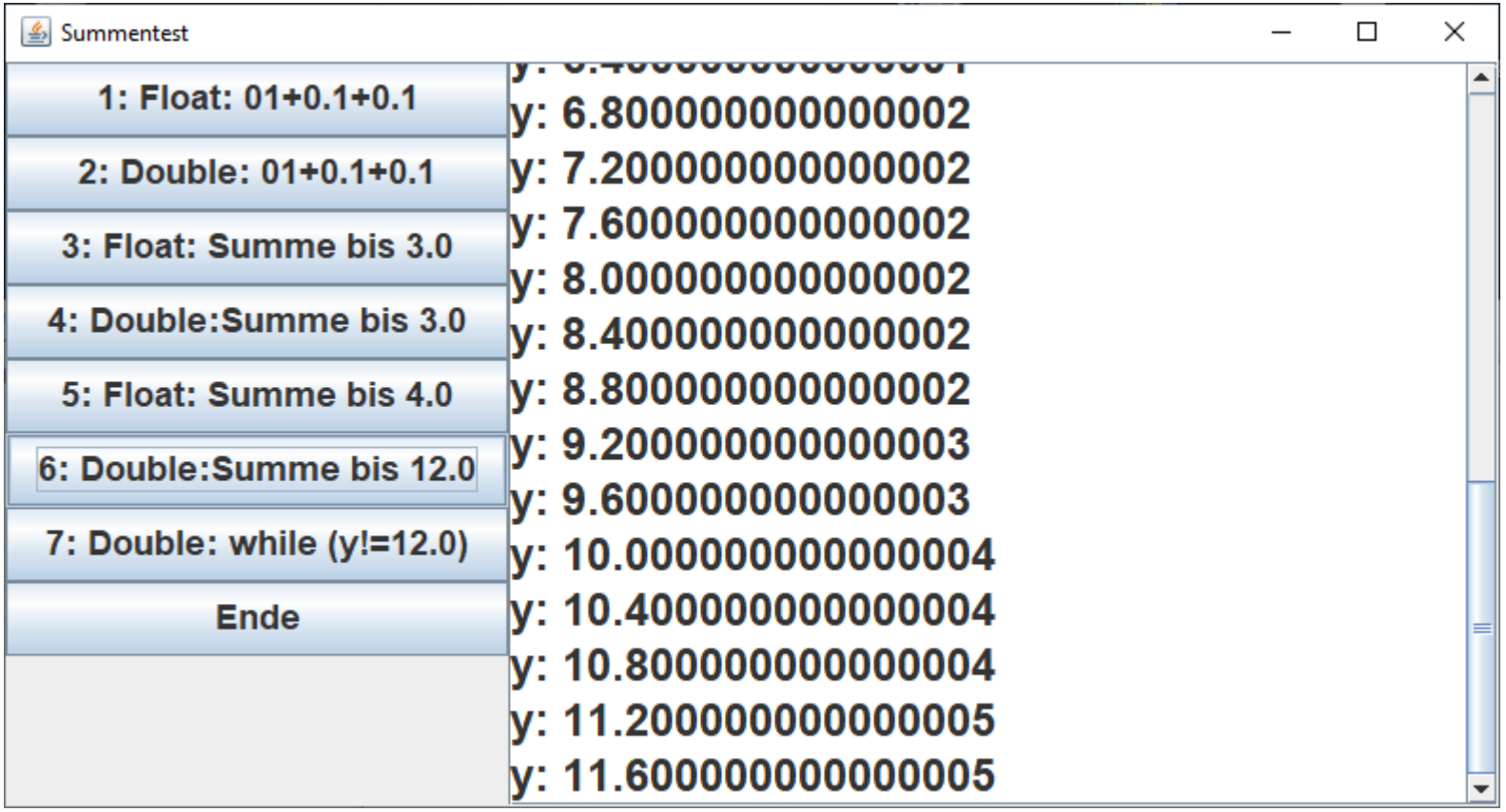

```
float y=0.0f;
editor.setText("Summe bis 3.0");
while (y<=3.0f) {
    editor.append("\ny: "+y);
    y+=0.3f;
}
```




```
double y=0.0;
editor.setText("Summe bis 3.0");
while (y<=3.0) {
    editor.append("\ny: "+y);
    y+=0.3;
}
```



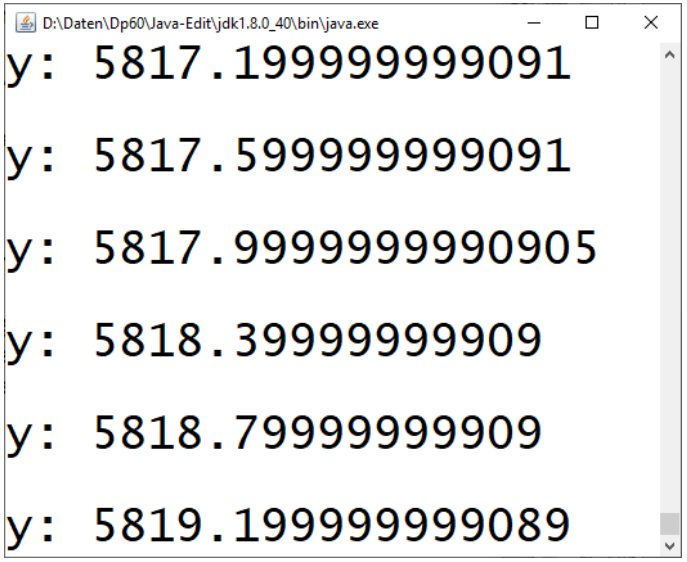

```
float y=0.0f;
editor.setText("Summe bis 4.0");
while (y<=4.0f) {
    editor.append("\ny: "+y);
    y+=0.4f;
}
```




```
double y=0.0;
editor.setText("Summe bis 12.0");
while (y<=12.0) {
    editor.append("\ny: "+y);
    y+=0.4;
}
```

7: Double: while (y!=12.0)

```
double y=0.0;
editor.append("\n\nSumme bis 12.0");
while (y!=12.0) {
    editor.append("\ny: "+y);
    System.out.println("\ny: "+y);
    y+=0.4;
}
```



8: Double: while (y<=12.0+0.4*0.001)

```
double y=0.0;
editor.append("\n\nSumme bis 12.0");
while (y<=12.0+0.4*0.001) {
    editor.append("\ny: "+y);
    System.out.println("\ny: "+y);
    y+=0.4;
}
```

8: Double: while (y<=12.0+0.4*0.001)