

NumProg WS 20/21 : Tutorübung 01

Willkommen zu NumProg! 😊

Ganz kurze Erklärungen bevor wir anfangen können:

- besondere Situation dieses Semester
- Organisatorisches
- zusätzliche Lernhilfen
- Mein Tutoriums-Stil

NumProg WS 20/21 : Tutorübung 01

Willkommen zu NumProg! 😊

Ganz kurze Erklärungen bevor wir anfangen können:

- besondere Situation dieses Semester
- Organisatorisches
- zusätzliche Lernhilfen
- Mein Tutoriums-Stil



Besondere Situation

- Wegen Corona alle Übungen dieses WS auch online
- Klausur vermutlich auch online (open book)
- 2. Semester über BBB
 - wir haben noch keine perfekten Lösungen für alles gefunden
 - gerne Verbesserungsvorschläge 😊
- Generell: Alle Infos unter moodle.tum.de

Organisatorisches

- Tutorstunde jede Woche **2 Stunden lang!** (13:00 – 15:00)
- Klausurdatum tbd (ich gebe euch bei neuen Infos Bescheid)
Wichtig: Es gibt nur eine Klausur pro Semester!
- Es gibt über das Semester 4 Programmieraufgaben
 - 70% der Punkte für 0.3 Bonus (280 von 400 Punkten)
 - für Aufgaben ist Gruppenanmeldung in Moodle nötig
 - erste Abgabe ist am 30.November, 03:00 nachts

Organisatorisches

- Tutorstunde jede Woche **2 Stunden lang!** (13:00 – 15:00)
- Klausurdatum tbd (ich gebe euch bei neuen Infos Bescheid)
Wichtig: Es gibt nur eine Klausur pro Semester!
- Es gibt über das Semester 4 Programmieraufgaben
 - 70% der Punkte für 0.3 Bonus (280 von 400 Punkten)
 - für Aufgaben ist Gruppenanmeldung in Moodle nötig
 - erste Abgabe ist am 30.November, 03:00 nachts
- **Wir Tutoren haben nichts mit den Programmieraufgaben zu tun ;)**

zusätzliche Lernhilfen

- NumProg Legende Hendrik
 - <https://hendrik.fam-moe.de/de/tutorien/>
 - Hendrik ist 7-maliger Tutor und hat über 4 Jahre sein Skript erstellt
 - neben Skript auch Probeklausuren, Lerntrainer, Hausaufgaben, etc.
- meine Folien (mehr dazu gleich)
 - werden auch jeden Dienstag um 15:30 hochgeladen (Link in Notizen)
- per Mail bei Fragen: cora.moser@tum.de
- moodle Forum

Mein Tutoriums-Stil

- Jede Woche tutoriumsbegleitende Folien
 - Kurze Erklärungen zu jeder (Teil-)Aufgabe
 - Graphische Ergänzungen
 - Zusammengefasste Lösungen zum Weiterrechnen

Mein Tutoriums-Stil

- Jede Woche tutoriumsbegleitende Folien
 - Kurze Erklärungen zu jeder (Teil-)Aufgabe
 - Graphische Ergänzungen
 - Zusammengefasste Lösungen zum Weiterrechnen
- Wir gehen jede Aufgabe in kleinen Schritten durch
 - Schema: Erklärung, Stillarbeit Aufgabenblatt, Lösung besprechen

Mein Tutoriums-Stil

- Jede Woche tutoriumsbegleitende Folien
 - Kurze Erklärungen zu jeder (Teil-)Aufgabe
 - Graphische Ergänzungen
 - Zusammengefasste Lösungen zum Weiterrechnen
- Wir gehen jede Aufgabe in kleinen Schritten durch
 - Schema: Erklärung, Stillarbeit Aufgabenblatt, Lösung besprechen
- Manche Leute sind schneller als andere, ich richte mich hauptsächlich nach den langsameren!

Mein Tutoriums-Stil und BBB

- Umfragen sind toll!
 - Zwischenfrage bei Erklärungen, ob alles soweit klar ist
 - Abfrage, wer schon mit einer Aufgabe fertig ist
 - yay, random Trollantworten
- Auf BBB schreiben ist gewöhnungsbedürftig
- Ihr könnt gerne mit mir reden 😊

Tutorübung 01 – heutige Themen

1. Wiederholung Zahlenbasen + binäre Brüche
2. Zweierkomplement
3. Assoziativgesetz bei Binärzahlen + Rundungsfehler
4. Gleitkommazahlen + 32-bit IEEE Standard

Zahlenbasen

Allgemeine Formel: $a_n \cdot b^n + \dots + a_1 \cdot b^1 + a_0 \cdot b^0$

	dezimal	binär	hexadezimal
Formel	$a_n \cdot 10^n + \dots + a_1 \cdot 10 + a_0$	$a_n \cdot 2^n + \dots + a_1 \cdot 2 + a_0$	$a_n \cdot 16^n + \dots + a_1 \cdot 16 + a_0$
Faktoren	0,1,2,3,4,5,6,7,8,9	0,1	0, ..., 9, A, B, C, D, E, F
Kennz.	N_{10}	N_2	N_{16} oder $0xN$
Bsp.	$19_{10} = 1 \cdot 10^1 + 9 \cdot 10^0$	$19_{10} = 10011_2$ $= 1 \cdot 2^4 + 0 \cdot 2^3 +$ $0 \cdot 2^2 + 1 \cdot 2^1 +$ $1 \cdot 2^0$	$19_{10} = 13_{16}$ $= 1 \cdot 16^1 + 3 \cdot 16^0$

Legende:

a := Faktor

b := Basis

N := natürliche Zahl

Zahlenbasen

Tipp:

Umrechnung

Binär \Leftrightarrow **Hexadezimal**

1011 0110

D 6

13₁₀ 6₁₀

13₁₀ 6₁₀

Lernen aller Zweierpotenzen bis 2^{12}

2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
4096	2048	1024	512	256	128	64	32	16	8	4	2	1

binäre rationale Zahlen

Allgemeine Formel: $a_n \cdot b^n + \dots + a_1 \cdot b^1 + a_0 \cdot b^0$

binäre rationale Zahlen

Allgemeine Formel: $a_n \cdot b^n + \dots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + \dots + a_{-m} \cdot b^{-m}$

binäre rationale Zahlen

Allgemeine Formel: $a_n \cdot b^n + \dots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + \dots + a_{-m} \cdot b^{-m}$

Beispiel an Binärzahl: $6.25_{10} = 110.01_2 =$
 $1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$
 $\qquad\qquad\qquad = 0,5 \qquad\qquad\qquad = 0,25$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10}$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10})$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$1 \div 101 =$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$\begin{array}{r} 1 \div 101 = 0 \\ \underline{-0} \end{array}$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$1 \div 101 = 0.$$

$$\begin{array}{r} -0 \\ \hline 10 \end{array}$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

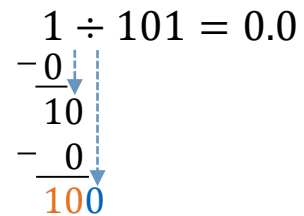
$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$\begin{array}{r} 1 \div 101 = 0.0 \\ -0 \\ \hline 10 \\ -0 \\ \hline 0 \end{array}$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

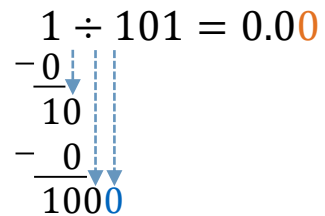
$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$1 \div 101 = 0.0$$


binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

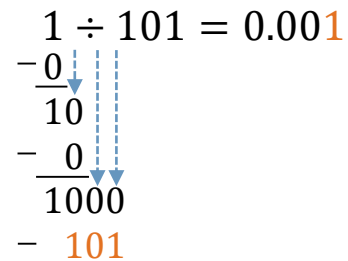
$$1 \div 101 = 0.0\textcolor{brown}{0}$$


$$\begin{array}{r} -0 \\ \hline 10 \\ -0 \\ \hline 1000 \end{array}$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

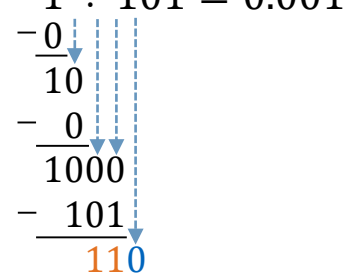
$$1 \div 101 = 0.00\textcolor{brown}{1}$$


$$\begin{array}{r} -0 \\ \hline 10 \\ -0 \\ \hline 1000 \\ -\textcolor{brown}{101} \end{array}$$

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

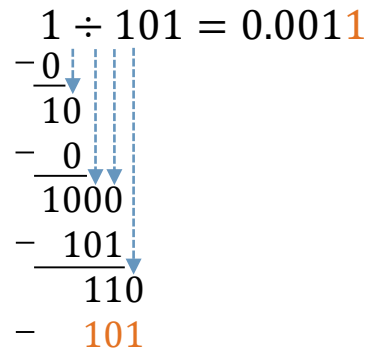
$$1 \div 101 = 0.001$$


Handwritten binary long division showing the calculation of $1 \div 101$. The dividend is 1, and the divisor is 101. The quotient is 0.001. The remainder is 110. Blue arrows point from the dividend to the quotient digits. The remainder 110 is written in orange and blue.

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$1 \div 101 = 0.001\textcolor{brown}{1}$$


```

  101 ) 1
        0
        10
        1000
        101
        110
        101
        ---
         101
  
```

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$1 \div 101 = 0.0011$$

Handwritten binary long division steps:

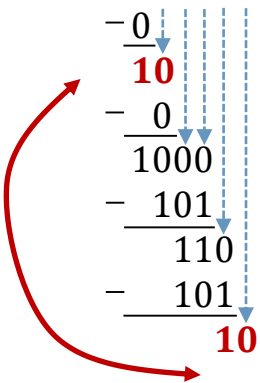
- 1 divided by 101 is 0, remainder 10.
- 10 divided by 101 is 0, remainder 1000.
- 1000 divided by 101 is 1, remainder 110.
- 110 divided by 101 is 1, remainder 10.

The final remainder 10 is shown in orange and blue.

binäre rationale Zahlen

binäre Brüche berechnen durch Polynomdivision:

$$\left(\frac{1}{5}\right)_{10} \equiv (1_{10} \div 5_{10}) \rightarrow \left(\frac{1}{101}\right)_2 \equiv (1_2 \div 101_2)$$

$$1 \div 101 = 0.\overline{0011}$$


Handwritten binary long division steps:

- 101 goes into 10 (101) 0 times, remainder 10.
- 101 goes into 1000 (1000) 1 time, remainder 110.
- 101 goes into 110 (101) 1 time, remainder 10.

Binärzahlen negativ darstellen

Überlegung: vorderstes Bit = Vorzeichenbit

Beispiel: -19 **1 10011**
 VZ eigentliche Zahl

Problem?

Binärzahlen negativ darstellen

Überlegung: vorderstes Bit = Vorzeichenbit

Beispiel: -19 **1 10011**
 VZ eigentliche Zahl

Problem: keine eindeutige Darstellung für 0,
 jeweils 1 Bit im Speicher für ± 0

Lösung: Zweierkomplementdarstellung

Stelle	1	2	3	...	$n - 1$	n
Wert	-2^{n-1}	2^{n-2}	2^{n-3}	...	2^1	2^0

Zweierkomplementdarstellung

Beispiel:

$$\begin{aligned}
 -19_{10} &= 1 \cdot (-2^5) + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\
 &= 101101_2
 \end{aligned}$$

bei einer Kodierung von 6 Bit (Bereich -32 bis $+31$)

Umrechnen von normaler zu Zweierkomplementdarstellung:

$$\begin{array}{rcl}
 19_{10} & = & 010011_2 \\
 \downarrow & & \text{Bits invertieren} \\
 101100_2 & & \\
 \downarrow & & \text{1 addieren} \\
 -19_{10} & = & 101101_2
 \end{array}$$

binäre Zahlen runden

Für eine binäre Zahl der Form

$$x, y \mid z$$

wobei wir bis \mid runden möchten, gelten folgende Rundungsregeln:

↓	$y \mid 0 z'$	abrunden, z' beliebig
↑	$y \mid 1 z'$	aufkunden, falls $z' \neq 0$
↓	$y' \text{ 0 } \mid 1 z'$	abrunden, falls $z' = 0$
↑	$y' \text{ 1 } \mid 1 z'$	aufkunden, falls $z' = 0$

verschiedene Rundungsregeln

5 bekannteste Rundungsregel-Definitionen:

Beispielzahlen	+3.5	−3.5	+4.5	−4.5
Runden zur nächsten geraden Zahl	+4	−4	+4	−4
Von 0 weg runden	+4	−4	+5	−5
Zu 0 hin runden	+3	−3	+4	−4
Zu $+\infty$ runden	+4	−3	+5	−4
Zu $-\infty$ runden	+3	−4	+4	−5

← default

nicht klausurrelevant

verschiedene Rundungsregeln

Warum ist „Runden zur nächsten geraden Zahl“ default?

Gegenbeispiel: gewohntes Runden (von 0 weg) während Rechnen mit 2 Zahlen

$$x = 1.00$$

$$y = 0.555$$

$$\rightarrow z = x + y = 1.56 \quad \text{aufrunden}$$

$$x' = z - y = 1.01 \quad \text{aufrunden}$$

$$z' = x' + y = 1.57 \quad \text{aufrunden}$$

...

→ tritt nicht auf, wenn immer auf nächste gerade Zahl gerundet wird

nicht klausurrelevant

Rundungsfehler

Absorption:

Kleine Zahl wird zu wesentlich größerer Zahl addiert/ von ihr subtrahiert

→ kleinere Zahl wird von größerer beim Runden „absorbiert“

Beispiel: $19_{10} + 0,25_{10} = 1011_2 + 0,01_2 = 1011,01_2 \xrightarrow{\text{runden auf 4 sig Stellen}} 1011_2 = 19_{10}$

→ $0,25_{10}$ wurde von der 19_{10} absorbiert

Auslöschung:

Zwei ähnlich große Zahlen werden voneinander subtrahiert

→ Zahlen auf gleichen Wert gerundet → Ergebnis 0 (schlecht zum Weiterrechnen)

Beispiel: $0,625_{10} - 0,5625_{10} = 0,1010_2 - 0,1001_2 \xrightarrow{\text{runden auf 2 sig Stellen}} 0,10_2 - 0,10_2 = 0$

→ beide Zahlen haben sich gegenseitig „ausgelöscht“, weiterrechnen schwer

Gleitkommazahlen

standardisierte Gestalt: $\pm 1, \dots \cdot 2^n$
 VZ normierte Zahl Exponent

Beispiel: $-19,25_{10}$ in standardisierte Form bringen

Vorzeichen: $-$

Zahl: $10011,01_2$

normierte Zahl: $1,001101_2$

Exponent dazu: 2^4 (Bitshift Komma um 4 Stellen)

$\rightarrow -19,25_{10} = -1,001101 \cdot 2^4$

32-bit IEEE Standard



Vorzeichen	1 Bit	0 := positiv, 1 := negativ
Exponent	8 Bit	gespeicherte 8 bit Binärzahl := Exponent + 127
Mantisse	23 Bit	normierte Zahl, 1 vor Komma nicht mitgespeichert

- reichen die 23 (24) Bit nicht für die Mantisse wird gerundet
- Zahlen 00000000 und 11111111 für Exponent sind reserviert

Rundungsfehler

absoluter Rundungsfehler:

$$f_{\text{abs}} = |x - \text{rd}(x)|$$

relativer Rundungsfehler:

$$f_{\text{rel}} = \left| \frac{f_{\text{abs}}}{x} \right|$$

Maschinengenauigkeit

Maschinengenauigkeit := größte positive Zahl ε , so dass $1 \oplus \varepsilon = 1$ *oder*
größte positive Zahl ε , so dass $\text{rd}(1 + \varepsilon) = 1$

Berechnung: $\varepsilon = \frac{1}{2} \cdot \beta^{1-t}$ $\beta := \text{Zahlenbasis}$
 $t := \text{Mantissenlänge}$