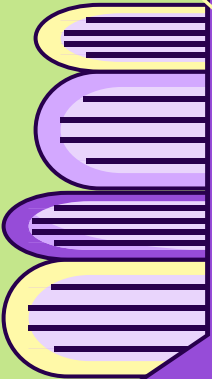
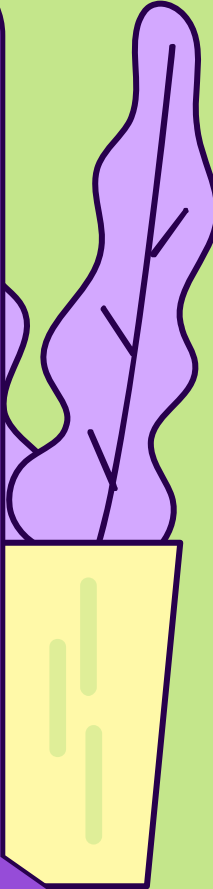




Herzlich willkommen
zum ersten Tutorium
des Semesters!





Erste Schritte

Mach mit bei Zulip! Es ist unglaublich nützlich



SCAN ME

Montag 10.00



SCAN ME

Montag 14.00



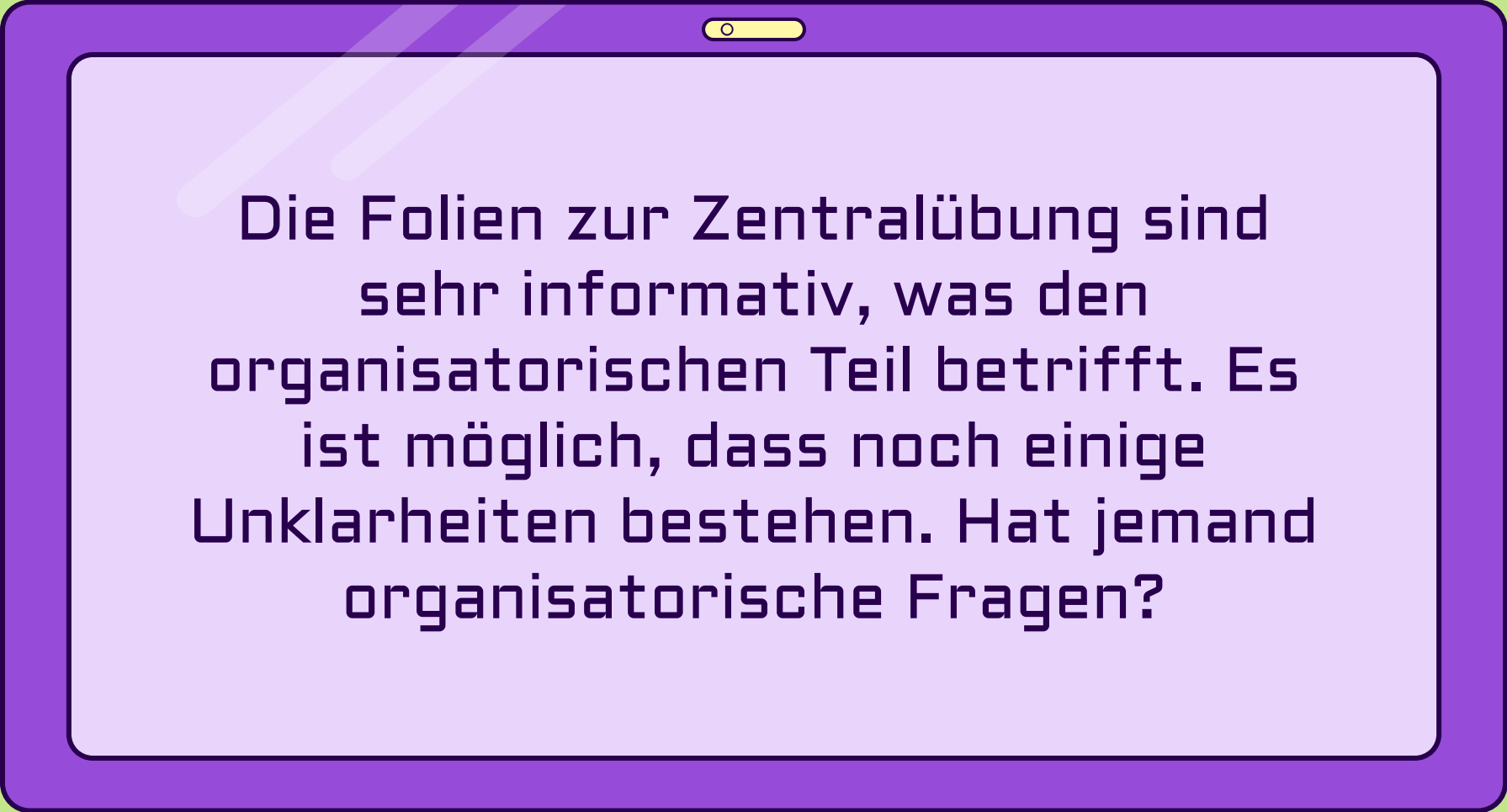
SCAN ME

Kanalgruppe

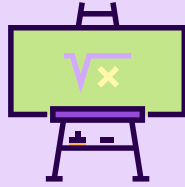


Erste Schritte

Ich habe auch eine Website, auf der ich meine Folien, meine Lösungen für die Übungsblätter und zusätzliche Informatikressourcen für andere Vorlesungen hochlade.



Die Folien zur Zentralübung sind sehr informativ, was den organisatorischen Teil betrifft. Es ist möglich, dass noch einige Unklarheiten bestehen. Hat jemand organisatorische Fragen?



Kurze Zfs


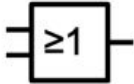
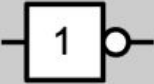

Dies ist kein Ersatz für die VL! Nur eine Erinnerung an die wichtigsten Themen der Woche.

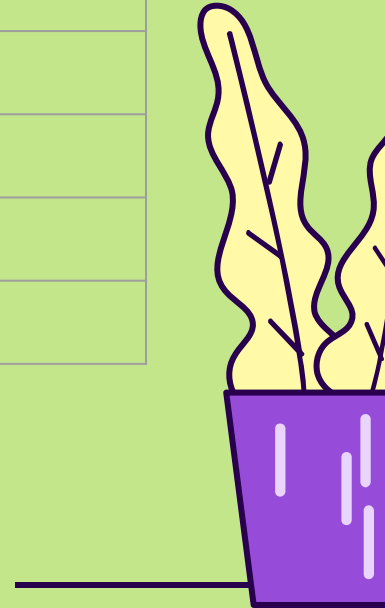


Lösen des Übungsblattes

Es ist nicht nötig, das Blatt vorher zu lösen. Du bekommst für jede Übung etwas Zeit, bevor wir sie gemeinsam lösen.

Logische Operatoren

		AND 	OR 	NOT 		XOR 
A	B	A & B	A B	!A	!B	A XOR B
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	1	0	0	0



Variante von Binärzahldarstellungen

Variante: Das Sign-Bit

- ▶ Verwende höchstwertiges Bit, um Vorzeichen darzustellen.
- ▶ Beispiel: Darstellung einer Zahl mit 8 Bits
 - $+7_{10} = 0000_0111_2$
 - $-7_{10} = 1000_0111_2$
- ▶ Vorteile:
 - trivial zu realisieren.
 - Vorzeichen durch erstes Bit repräsentiert → direkt ablesbar
- ▶ Nachteile:
 - Spezialbehandlung für Addition und Subtraktion.
 - Reduktion der Anzahl an darstellbaren Werten (zwei Darstellungen für die Zahl 0).



Variante von BinärzahlDarstellungen

Variante: Das Zweierkomplement

- ▶ Schritt 1: Einerkomplement
 - Bitkomplement: Invertierung jedes einzelnen Bits
 - Beispiel: Negation von Zahl 7_{10} : $0000\ 0111_2 \rightarrow 1111\ 1000_2$
- ▶ Schritt 2: Addition von „1“
 - $-7_{10} = 1111\ 1001_2$
- ▶ Vorteile:
 - siehe folgende Folien
 - Vorzeichen durch erstes Bit repräsentiert \rightarrow negative Zahl direkt erkennbar, aber Zahlenwert nicht
- ▶ Nachteile:
 - Zahl nicht direkt „ablesbar“ (außerdem: Breite muss bekannt sein)
 - kein Gegenstück für kleinste negative Zahl im positiven Zahlenbereich

Klassische
Hausaufgabe



Definitionsbereiche von verschiedenen Typen

Typname in C	Bits	Vorzeichen	min	max
char	8	ja	-128	127
unsigned char		nein	0	255
short	16	ja	-32.768	32.767
unsigned short		nein	0	65.535
int	32	ja	-2.147.483.648	2.147.483.647
unsigned int		nein	0	4.294.967.295
long long	64	ja	-9.223.372.036.854.775.808	9.223.372.036.854.775.807
unsigned long long		nein	0	18.446.744.073.709.551.615



Definitionsbereiche von verschiedenen Typen

Typname in C	Bits	Vorzeichen	min	max
char	8	ja	-128	127
unsigned char		nein	0	255
short	16	ja	-32.768	32.767
unsigned short		nein	0	65.535
int	32	ja	-2.147.483.648	2.147.483.647
unsigned int		nein	0	4.294.967.295
long long	64	ja	-9.223.372.036.854.775.808	9.223.372.036.854.775.807
unsigned long long		nein	0	18.446.744.073.709.551.615

Min (unsigned) = 0

Min (signed) = $-1 * 2^{(\text{Bitzahl} - 1)}$

Max (unsigned) = $(2^{\text{Bitzahl}}) - 1$

Max (signed) = $(2^{(\text{Bitzahl} - 1)}) - 1$



Keep in mind!

Strings werden in Zahlen umgewandelt: Oft mit NULL terminiert (C-Strings)
aber auch explizite Größenangabe zu Beginn möglich (Pascal-Strings)

→ Wichtig für das Arbeiten mit ASCII-Zeichen

Klassische
Klausurfrage

(John) Von Neumann Ansatz

Abstraktion eines Rechensystemes

- Wohldefinierte Komponenten
- Wohldefinierte Datenflüsse

Zentrale Eigenschaften

- Struktur des Rechners unabhängig vom bearbeiteten Problems
- Programm und Daten im gleichen Speicher
- Speicher ist in Zellen gleicher Größe unterteilt
- Programm ist eine Reihe von Befehlen



In der Informatik werden Zahlen in der Regel im sog. Stellenwertsystem dargestellt. Der Wert einer Zahl hängt dabei von der Position der Ziffern ab:

$$W = \sum_{i=0}^{n-1} a_i \cdot B^i, \quad (1)$$

wobei B die Basis, n die Anzahl der Stellen und a_i die i -te Ziffer aus dem Ziffernbereich von 0 bis $B - 1$ ist. Im Alltag ist die Basis meist 10 (Dezimalsystem). Weitere häufig verwendete Stellenwertsysteme sind das Dualsystem (Binärsystem, $B = 2$), das Oktalsystem ($B = 8$) und das Hexadezimalsystem ($B = 16$).

- a) Überlegen Sie sich einen Algorithmus, der beliebig lange Zahlen vom Dezimalsystem in das Binärsystem umwandelt. Testen Sie den Algorithmus mit den Zahlen: $(42)_{10}$, $(100)_{10}$ und $(1.000)_{10}$.



a) Hierfür eignet sich beispielsweise die Divisionsmethode ¹:

$$\begin{array}{rclcl} 42 & / & 2 & = & 21 & 0 \\ 21 & / & 2 & = & 10 & 1 \\ 10 & / & 2 & = & 5 & 0 \\ 5 & / & 2 & = & 2 & 1 \\ 2 & / & 2 & = & 1 & 0 \\ 1 & / & 2 & = & 0 & 1 \end{array}$$

$$\Rightarrow (42)_{10} = (10.1010)_2$$

$$\begin{array}{rclcl} 100 & / & 2 & = & 50 & 0 \\ 50 & / & 2 & = & 25 & 0 \\ 25 & / & 2 & = & 12 & 1 \\ 12 & / & 2 & = & 6 & 0 \\ 6 & / & 2 & = & 3 & 0 \\ 3 & / & 2 & = & 1 & 1 \\ 1 & / & 2 & = & 0 & 1 \end{array}$$

$$\Rightarrow (100)_{10} = (110.0100)_2$$

$$\begin{array}{rclcl} 1000 & / & 2 & = & 500 & 0 \\ 500 & / & 2 & = & 250 & 0 \\ 250 & / & 2 & = & 125 & 0 \\ 125 & / & 2 & = & 62 & 1 \\ 62 & / & 2 & = & 31 & 0 \\ 31 & / & 2 & = & 15 & 1 \\ 15 & / & 2 & = & 7 & 1 \\ 7 & / & 2 & = & 3 & 1 \\ 3 & / & 2 & = & 1 & 1 \\ 1 & / & 2 & = & 0 & 1 \end{array}$$

$$\Rightarrow (1.000)_{10} = (11.1110.1000)_2$$

In der Informatik werden Zahlen in der Regel im sog. Stellenwertsystem dargestellt. Der Wert einer Zahl hängt dabei von der Position der Ziffern ab:

$$W = \sum_{i=0}^{n-1} a_i \cdot B^i, \quad (1)$$

wobei B die Basis, n die Anzahl der Stellen und a_i die i -te Ziffer aus dem Ziffernbereich von 0 bis $B - 1$ ist. Im Alltag ist die Basis meist 10 (Dezimalsystem). Weitere häufig verwendete Stellenwertsysteme sind das Dualsystem (Binärsystem, $B = 2$), das Oktalsystem ($B = 8$) und das Hexadezimalsystem ($B = 16$).

- b) Überlegen Sie sich einen Algorithmus, der beliebig lange Zahlen vom Binärsystem in das Dezimalsystem umwandelt. Testen Sie den Algorithmus mit den Zahlen: $(1.0101)_2$ und $(1110.0011)_2$.



b) Ein einfaches Verfahren für kleinere Zahlen ist die Multiplikation der Ziffer mit ihrem Stellenwert und anschließendem Aufaddieren ²:

$$(1.0101)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (21)_{10}$$

$$(1110.0011)_2 = 2^7 + 2^6 + 2^5 + 2^1 + 2^0 = (227)_{10}$$

c) Lösen Sie mit Hilfe von „Tricks“ folgende Umwandlungen:

- $(1111.1111)_2 = (?)_{10}$
- $(1.0000.0000)_2 = (?)_{10}$
- $(65)_{10} = (?)_2$
- $(1.0000.1001.0010)_2 = (4242)_{10}$
 $(10.0001.0010.0100)_2 = (?)_{10}$



- c)
- $(1111.1111)_2 = (1.0000.0000)_2 - (1)_2 = 2^8 - 1 = (255)_{10}$
 - $(1.0000.0000)_2 = 2^8 = (256)_{10}$
 - $(65)_{10} = (64)_{10} + (1)_{10} = 2^6 + (1)_{10} = (100.0000)_2 + (1)_2 = (100.0001)_2$
 - $(1.0000.1001.0010)_2 = (4242)_{10}$
 $(10.0001.0010.0100)_2 = (1.0000.1001.0010)_2 \cdot (2)_{10} = (4242)_{10} \cdot (2)_{10} = (8484)_{10}$

d) Wandeln Sie die folgenden Zahlen vom Binär- ins Hexadezimalsystem bzw. umgekehrt um:

- $(1111.1111)_2 = 0x?$
- $(1010.1100.0011)_2 = 0x?$
- $0x1234 = (?)_2$
- $0xC0FFEE = (?)_2$



d) i) $(1111.1111)_2 = 0xFF$

ii) $(1010.1100.0011)_2 = 0xAC3$

iii) $0x1234 = (1.0010.0011.0100)_2$

iv) $0xC0FFEE = (1100.0000.1111.1111.1110.1110)_2$

a) Die vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division verhalten sich im Binärsystem wie im Dezimalsystem: die einzelnen Ziffern werden stellenweise verarbeitet. Lösen Sie die folgenden Rechenaufgaben (alle Zahlen sind positiv):

- $(10.1010)_2 + (11.0011)_2 = (?)_2$
- $(11.0011)_2 - (10.1010)_2 = (?)_2$
- $(10.1010)_2 \cdot (11)_2 = (?)_2$
- $(01.1100)_2 : (0100)_2 = (?)_2$



a) • $(10.1010)_2 + (11.0011)_2 = (101.1101)_2$

• $(11.0011)_2 - (10.1010)_2 = (1001)_2$

• $(10.1010)_2 \cdot (11)_2 = (111.1110)_2$

$$011100_2 : 0100_2 = 111_2$$

$$\begin{array}{r} 0111 \\ - \quad 0100 \\ \hline 0011 \end{array}$$

• $(01.1100)_2 : (0100)_2 =$

$$\begin{array}{r} 0110 \\ - \quad 0100 \\ \hline 0010 \\ - \quad 0100 \\ \hline 0000 \end{array}$$

b) Bisher haben wir lediglich positive Zahlen betrachtet. Wie könnten negative Zahlen im Binärsystem dargestellt werden? Vergleichen Sie anhand der Zahl $-(42)_{10}$ die Vor- und Nachteile der Darstellungsarten. Betrachten Sie dazu 8 binäre Stellen.



Art	Beschreibung	Vor-/Nachteil	Beispiel
Vorzeichenbit	Ein extra Bit nur für das Vorzeichen	\oplus Analog zu Dezimalsystem \ominus Doppelte Null \ominus neue Arithmetik nötig	1010.1010
b) Einerkomplement	Alle Bits invertieren	\oplus intuitiv \ominus Doppelte Null \ominus neue Arithmetik nötig	1101.0101
Zweierkomplement	Einerkomplement +1	\oplus volle Ausnützung des Zahlenraums \oplus Arithmetik wieder verwendbar \ominus Zahl nicht direkt ablesbar	1101.0110

c) Berechnen Sie den Wert des Terms $(0011.0011)_2 - (0010.1010)_2 = (?)_2$ (bekannt aus Aufgabe 2) indem Sie den Subtrahenden negieren und anschließend auf den Minuenden aufaddieren. Verwenden Sie das Zweierkomplement.



c) Als erstes wird das Zweierkomplement des Subtrahenden berechnet:

$$-(0010.1010)_2 = (1101.0110)_2$$

Und dann auf den Minuenden addiert:

$$(0011.0011)_2 + (1101.0110)_2 = (0000.1001)_2$$

d) Lösen Sie die folgenden Aufgaben, indem Sie die Zahlen zuerst ins Binärsystem mit jeweils 5 binären Stellen umwandeln und dann das Ergebnis im Binärsystem ausrechnen. Benutzen Sie das Zweierkomplement.

- $(-1) - 1$
- $(-2) \cdot (-3)$
- $(-8) : 2$



d) • $-1 - 1 = 11111_2 + 11111_2 = (1)11110_2 = -2_{10}$

• $-2 \cdot -3 = 11110_2 \cdot 11101_2$

	11110 · 11101	
	11110	
+	00000	
+	11110	
+	11110000	Übertrag
	10010110	Zwischenergebnis 1
+	11110	
+	111100000	Übertrag
	110000110	Zwischenergebnis 2
+	11110	
+	1100000000	Übertrag
	1101100110	Ergebnis

Ergebnis sind die unteren 5 Bits: $00110_2 = 6_{10}$

- $-8 : 2 = 11000_2 : 00010_2 = 11100_2$

$$\begin{array}{r}
 11000 : 00010 = 01100 \\
 \hline
 11 \\
 - 10 \\
 \hline
 010 \\
 - 10 \\
 \hline
 000
 \end{array}$$

Ergebnis ist positiv, sollte aber negativ sein. Wir müssen somit für die Division zuerst das Vorzeichen des Ergebnisses berechnen und dann die Division mit dem Zweierkomplement der negativen Zahlen durchführen.

Zweierkomplement der negativen Zahl bilden: $11000_2 \rightarrow 01000_2$

$$\begin{array}{r}
 01000 : 00010 = 00100 \\
 \hline
 010 \\
 - 10 \\
 \hline
 000
 \end{array}$$

Ergebnis negieren: $00100_2 \rightarrow 11100_2 = -4_{10}$

Welcher Zahlenbereich kann mit den folgenden Binärformaten dargestellt werden?

- 8-Bit vorzeichenlos (unsigned char)
- 8-Bit vorzeichenbehaftet im Zweierkomplement (char)
- 16-Bit vorzeichenlos (unsigned short)
- 32-Bit vorzeichenbehaftet im Zweierkomplement (int)



- von 0 bis $2^8 - 1 = 255$
- von $-2^7 = -128$ bis $2^7 - 1 = 127$
- von 0 bis $2^{16} - 1 = 65535$
- von -2^{31} bis $2^{31} - 1$



Viel Erfolg bei den Hausaufgaben

Ich bin für eure
Fragen da! Sendet
mir einfach eine PM
auf Zulip oder
schreib ins Channel!