



Technische  
Universität  
Braunschweig



# Algorithmen und Datenstrukturen

## Große Übung 0

Ramin Kosfeld und Chek-Manh Loi

02.11.2023

# Organisation

# Homepage und Anmeldung

Allgemeine Informationen	
Veranstaltungsname	Vorlesung/Übung: Algorithmen und Datenstrukturen
Veranstaltungsnummer	4227001
Semester	WiSe 2023/24
Aktuelle Anzahl der Teilnehmenden	0
erwartete Teilnehmendenanzahl	430
Heimat-Einrichtung	Abteilung Algorithmik (ALG)
Veranstaltungstyp	Vorlesung/Übung in der Kategorie Lehre

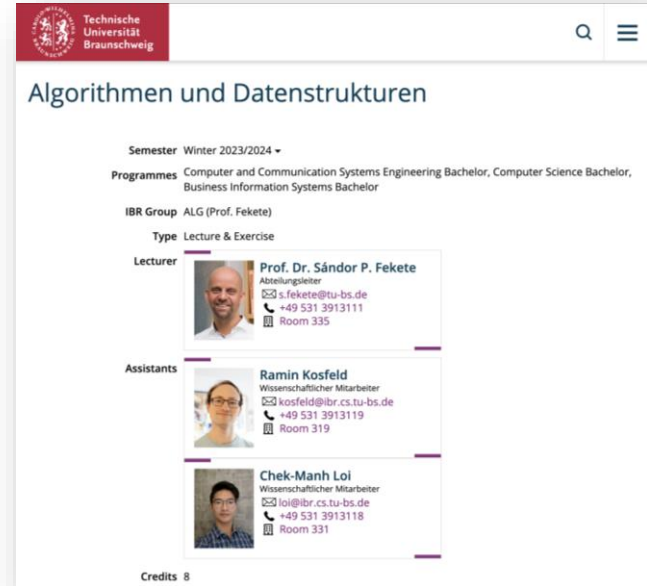
Stud.IP?



# Homepage und Anmeldung

Veranstaltungsübersicht:

[ibr.cs.tu-bs.de/courses/ws2324/aud/index.html](http://ibr.cs.tu-bs.de/courses/ws2324/aud/index.html)

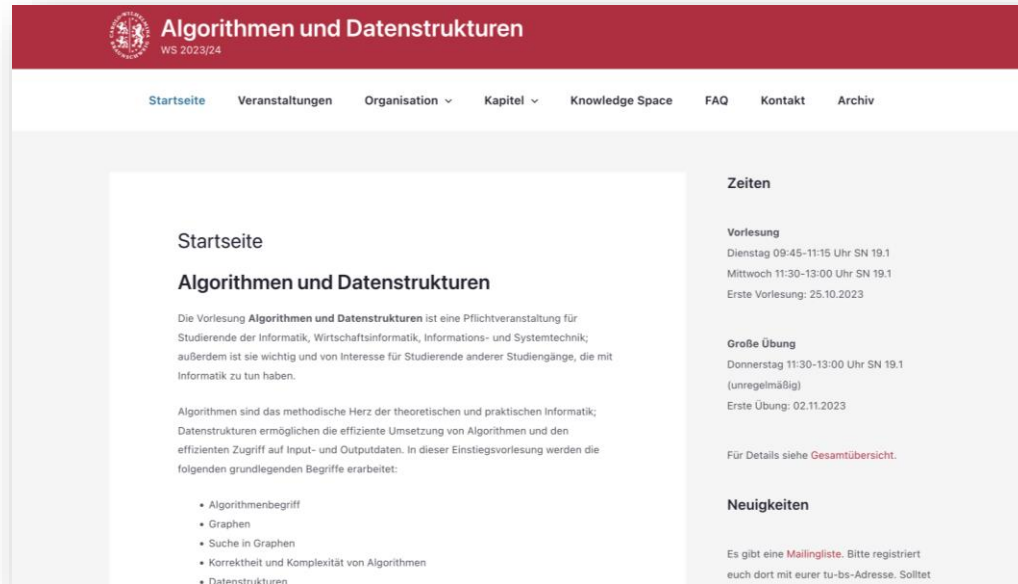


The screenshot shows the course page for 'Algorithmen und Datenstrukturen' on the website of Technische Universität Braunschweig. The page includes the university logo, a search icon, and a menu icon. The course title is prominently displayed. Below the title, the semester is listed as 'Winter 2023/2024'. The program is identified as 'Computer and Communication Systems Engineering Bachelor, Computer Science Bachelor, Business Information Systems Bachelor'. The IBR Group is 'ALG (Prof. Fekete)'. The course type is 'Lecture & Exercise'. The lecturer is 'Prof. Dr. Sándor P. Fekete', an 'Abteilungsleiter' with contact information: email 's.fekete@tu-bs.de', phone '+49 531 3913111', and room 'Room 335'. There are three assistants listed: 'Ramin Kosfeld' (Wissenschaftlicher Mitarbeiter, email 'kosfeld@ibr.cs.tu-bs.de', phone '+49 531 3913119', room 'Room 319') and 'Chek-Manh Loi' (Wissenschaftlicher Mitarbeiter, email 'loi@ibr.cs.tu-bs.de', phone '+49 531 3913118', room 'Room 331'). The course is worth 'Credits 8'.

# Homepage und Anmeldung

Die eigentliche Kursseite: [aud.ibr.cs.tu-bs.de](http://aud.ibr.cs.tu-bs.de)

... Folien, Hausaufgaben, Vorlesungsvideos, Altklausuren, **Übungsanmeldungen:**



The screenshot shows the homepage for the course 'Algorithmen und Datenstrukturen' (Algorithms and Data Structures) for the winter semester 2023/24. The page has a dark red header with the course title and the university's logo. Below the header is a navigation menu with links for 'Startseite', 'Veranstaltungen', 'Organisation', 'Kapitel', 'Knowledge Space', 'FAQ', 'Kontakt', and 'Archiv'. The main content area is divided into two columns. The left column contains the 'Startseite' section, which includes the course title, a brief description of the course as a mandatory event for students in Informatics, and a list of topics to be covered: Algorithmenbegriff, Graphen, Suche in Graphen, Korrektheit und Komplexität von Algorithmen, and Datenstrukturen. The right column contains the 'Zeiten' (Times) section, which lists the lecture schedule: Tuesday 09:45-11:15 Uhr SN 19.1, Wednesday 11:30-13:00 Uhr SN 19.1, and the first lecture on 25.10.2023. It also lists the 'Große Übung' (Large Exercise) on Thursday 11:30-13:00 Uhr SN 19.1 (irregularly) on 02.11.2023. A link to 'Gesamtübersicht' (Overall Overview) is provided for details. The 'Neuigkeiten' (News) section at the bottom right mentions a mailing list and asks students to register with their TU-BS address.

# Homepage und Anmeldung

- Anmeldung für die kleinen Übungen ebenfalls [auf der Kursseite](#)
- Zuweisung erfolgt (voraussichtlich) am 10.11.23

## Anmeldung Kleine Übungen

Allgemein / 26. Oktober 2022

Die Anmeldung für die kleinen Übungen ist ab sofort bis einschließlich 08.11.22 geöffnet.

Bitte beachtet folgende Punkte:

- Bitte wählt so viele Termine wie möglich, damit wir am Ende eine faire Zuteilung garantieren können.
- Wir werten immer die neueste Anmeldung (einfach das Formular erneut abschicken).
- Bei erfolgreicher Anmeldung erhält man eine Mail mit einer Übersicht der eingegebenen Daten. Das ist noch keine Zuweisung der Gruppen!
- Überprüft **vor** dem Absenden des Formulars, ob eure Daten (vor allem Matrikelnummer und eMail) korrekt sind.
- Die **Einteilung** in die Gruppen findet voraussichtlich am 09.11.22 statt. Ihr erhaltet dazu eine separate Mail.

Nachname

Vorname

Matrikelnummer

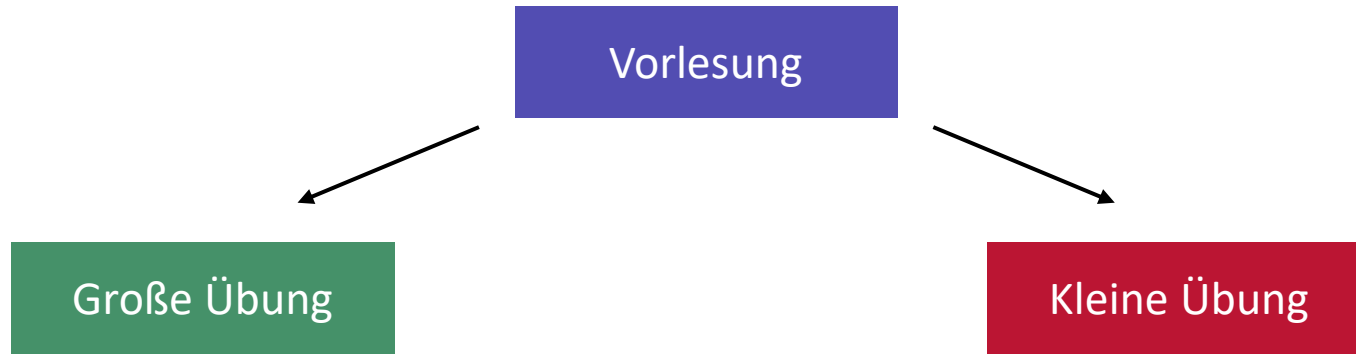
# Semesterplan

## Semesterplan Algorithmen und Datenstrukturen WS23/24

Woche (KW)	Woche (Datum)	Vorlesung (Di.,Mi)	Gr. Übung (Do.)	Kl. Übung (Mo.-Fr.)	Ausgabe (tba)	Abgabe (Mo. 14 Uhr)	Besprechung (in kl. Übung)	
43	23.10.	/,0						
44	30.10.	/,1	0					
45	06.11.	2,3			P0+HA1			
46	13.11.	4,5	1	1			P0	
47	20.11.	6,7			P1+HA2	HA1		
48	27.11.	8,9	2	2			P1+HA1	
49	04.12.	10,11			P2+HA3	HA2		
50	11.12.	12,13	3	3			P2+HA2	
51	18.12.	14,15	4		P3+HA4	HA3		
52	25.12.	Weihnachtsferien						
1	01.01.	Weihnachtsferien						
2	08.01.	16,17	5	4			P3+HA3	
3	15.01.	18,19	6		P4+HA5	HA4		
4	22.01.	20,21		5			P4+HA4	
5	29.01.	22,23	7		P5	HA5		
6	05.02.	24,25	8	6			P5+HA5	

Wenn kurzfristig Änderungen im Zeitplan notwendig sein, werden diese über die Mailingliste bekannt gegeben.

The screenshot shows a website interface with a navigation bar at the top containing 'Kapitel', 'Knowledge Space', 'FAQ', 'Kontakt', and 'Archiv'. Below the navigation bar, there are sections for 'Zeiten' (times) and 'Neuigkeiten' (news). The 'Zeiten' section lists lecture and exercise times. The 'Neuigkeiten' section contains a message about a mailing list. A red circle highlights the text 'Für Details siehe Gesamtübersicht.' in the 'Neuigkeiten' section.



## Tafelübungen im Hörsaal (mit allen)

- Aufarbeitung der Inhalte
- Beantwortung von Fragen
- Interaktion!

## Kleingruppen (Seminarräume)

- Vertiefung der Inhalte
- Selbständiges Arbeiten
- Besprechung von Hausaufgaben
- Noch individueller auf Fragen eingehen



# Hausaufgaben und Übungsblätter

5 verpflichtende  
Hausaufgabenblätter

6 freiwillige  
Übungsblätter

- Studienleistung
- 20 Punkte pro Blatt
- Einzelabgabe

- Zusätzliche Vertiefung
- Prüfungsvorbereitung

- Studienleistung: 50% der Gesamtpunkte
  - Studienleistung ist **keine** Voraussetzung, um an der Prüfung teilzunehmen.
  - Studienleistung ist **eine** Voraussetzung, um das Modul abzuschließen.
  - Studienleistung ist nicht benotet und fließt nicht in die Prüfung ein.

Algorithmen und Datenstrukturen  
Prof. Dr. Sándor P. Fekete  
Ramin Kosfeld  
Chek-Manh Loi

Winter 2023/24  
Abgabe: 21.11.2023  
Rückgabe: ab 28.11.2023

## Hausaufgabenblatt 1

Abgabe der Lösungen bis zum 21.11.2023 um 14:00 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet.

Bitte die Blätter zusammenheften und vorne mit Namen und Matrikelnummer versehen!



# Hausaufgaben

## Wozu Hausaufgaben?

Die Hausaufgaben dienen *Euch* (nicht uns) zur Vorbereitung auf die Klausur.

- Ideale Nachbereitung der Vorlesungsinhalte
- Zeitersparnis bei der Prüfungsvorbereitung
- Direktes Feedback über euren aktuellen Lernstand

- Zu späte Abgaben: 0 Punkte
- Zusammen überlegen ist okay, **ABER:** Einzelne aufschreiben und abgeben, sonst: 0 Punkte

# Hausaufgaben



Hausaufgabenschrank

# Klausur

- Voraussichtlich 13.02.24, zwischen 08:30 Uhr und 10:30 Uhr.
- Raumaufteilung und Beginn der Klausur folgen auf Webseite.
- Inhalt: Prinzipiell alles aus VL und Übung
- Dauer: 2 Stunden



Technische Universität Braunschweig  
Institut für Betriebssysteme und Rechnerverbund  
Abteilung Algorithmik

Wintersemester 2019/2020

Prof. Dr. Sándor P. Fekete  
Arne Schmidt

Klausur  
*Algorithmen und Datenstrukturen*  
28.02.2020

Name: .....

Vorname: .....

Matr.-Nr.: .....

Studiengang: .....

Bachelor     Master     Andere

**Klausurcode:**  
*Dieser wird benötigt, um das Ergebnis der Klausur abzurufen.*

# Mailingliste

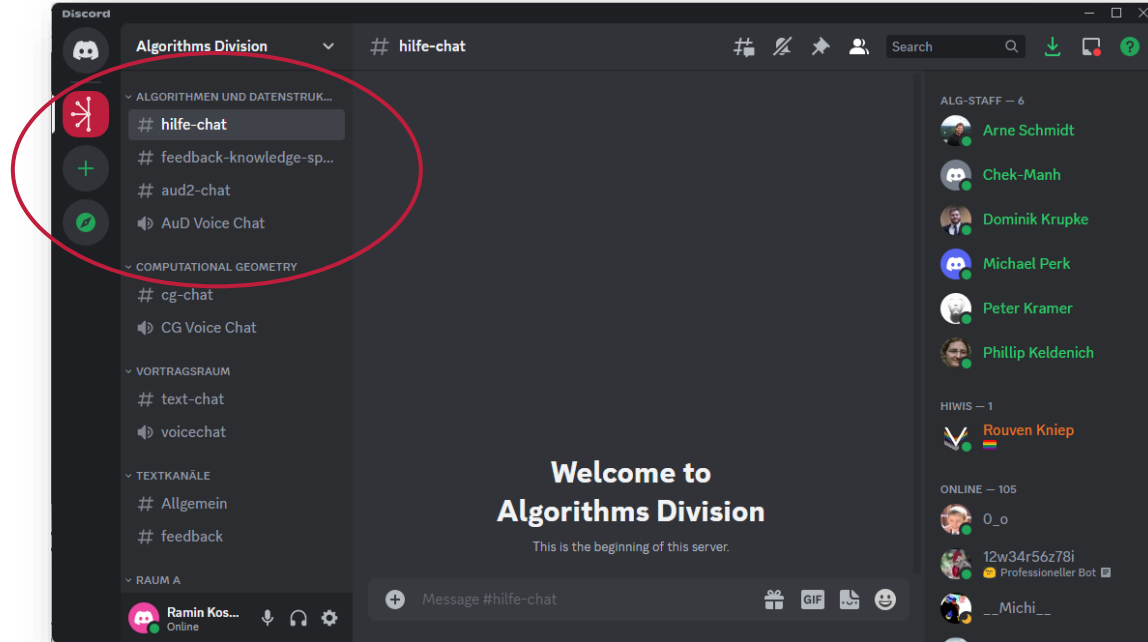
- Anmeldung [hier](#) oder über Homepage
- Für Informationen wie
  - Raumänderungen,
  - Ausfälle,
  - etc.
- Möglichkeit für Fragen

The screenshot shows the Postorius interface for the mailing list 'Aud' at 'aud@ibr.cs.tu-bs.de'. The page includes a navigation bar with 'Postorius', 'Lists', and 'Archives' links, and 'Login' and 'Sign Up' buttons. The main content area is divided into sections: 'Summary' (describing the list as 'Mailingliste für Algorithmen und Datenstrukturen'), 'Subscription / Unsubscription' (with a 'Log In' button), and a form for subscribing without an account (with fields for 'Your email address' and 'Your name (optional)', and a 'Subscribe' button). The footer contains 'Postorius Documentation · GNU Mailman · Postorius Version 1.3.6'.

# Discord



Discord



Kommunikation abseits der Vorlesung und Austausch mit Mitstudierenden.

# Fragen

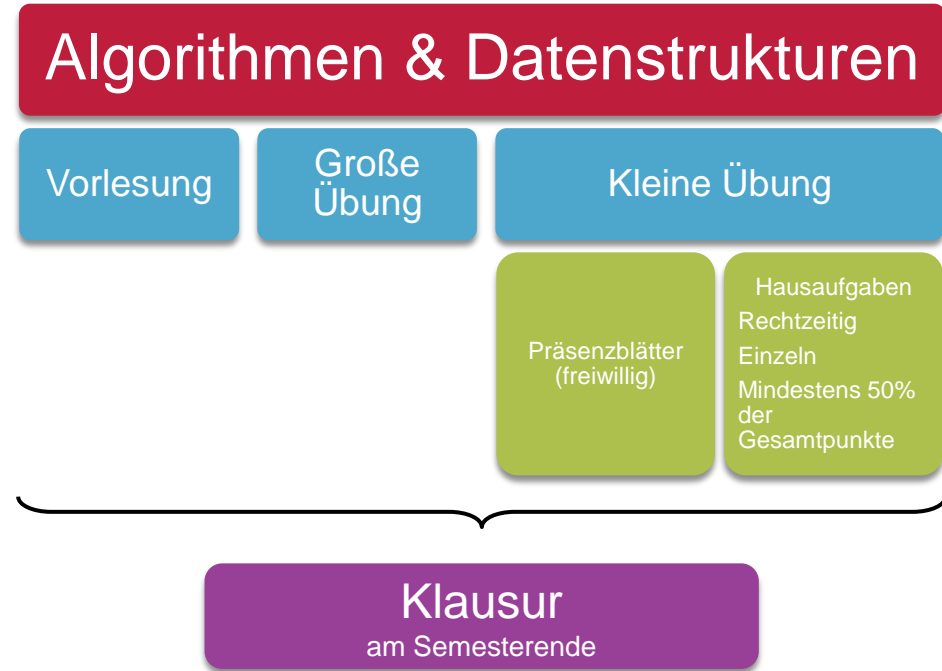
Stellt Eure Fragen ...

- Während oder nach den Vorlesungen
- Euren Tutoren in den kleinen Übungen!
- Über die Mailingliste (ins Plenum)
- Per Mail ([kosfeld@ibr.cs.tu-bs.de](mailto:kosfeld@ibr.cs.tu-bs.de) & [loi@ibr.cs.tu-bs.de](mailto:loi@ibr.cs.tu-bs.de))
  - Sprechstunde von Ramin und Chek-Manh (nach Vereinbarung)



# Zusammenfassung

- Kursseite
  - Semesterplan
  - Anmeldung kleine Übung
- Kommunikation
  - In Person
  - Mailingliste!
  - Mail an uns beide
  - Discord (informeller Austausch)





# Fragen?

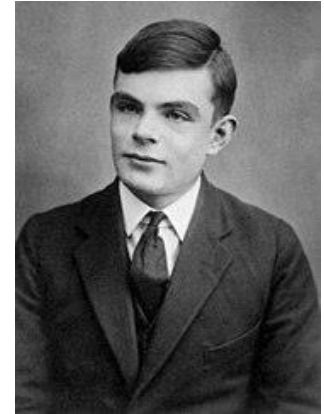
# Algorithmus

Eindeutige Handlungsvorschrift zur Lösung eines Problems

- Eigenschaften
  - Ausführbarkeit
  - Endlichkeit
  - Endliche Ausführzeit
  - Endlicher Speicherbedarf
- Beschrieben durch
  - Prosatext
  - **Pseudocode**



Donald Knuth



Alan Turing

# Pseudocode

## Wörterbuch



pseu·do

/pseúdo/

Adjektiv **UMGANGSSPRACHLICH**

nicht echt, nur nachgemacht, nachgeahmt

## Wörterbuch



Quell·code

/Quéllcode/

Substantiv, maskulin [der] **EDV**

in einer [höheren] Programmiersprache geschriebene Abfolge von Programmanweisungen, die vom Menschen gelesen, aber erst nach einer elektronischen Übersetzung vom Computer verarbeitet werden können

# Warum Pseudocode?

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Ist dieser Code effizient zu lesen?

Was können wir weglassen, wenn ein Mensch (kein Computer) diesen Code verstehen soll?

# Warum Pseudocode?

Java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Pseudocode

```
function main(args)  
    print Hello World!  
end function
```

# Beispiel: Euklidischer Algorithmus

*Algorithmus zur Berechnung des größten gemeinsamen Teilers zwei ganzer Zahlen*

## Als Prosatext

Solange die Zahl  $b$  nicht 0 ist, wählen wir  $h = a \bmod b$ , setzen  $a$  auf  $b$  und  $b$  auf  $h$ . Nachdem  $b = 0$ , geben wir  $a$  zurück.

## Als Pseudocode

```
function Euklid( $a, b$ )  
  while  $b \neq 0$  do  
     $h \leftarrow a \bmod b$   
     $a \leftarrow b$   
     $b \leftarrow h$   
  end while  
  return  $a$   
end function
```

(Für modulo gilt  $r = a \bmod b$ , sodass  $a = q \cdot b + r$  für ein  $q \in \mathbb{Z}$  gilt mit  $0 \leq r < b$ .)

# Beispiel: Euklidischer Algorithmus

## Pseudocode

```
function Euklid( $a$ ,  $b$ )  
  while  $b \neq 0$  do  
     $h \leftarrow a \bmod b$   
     $a \leftarrow b$   
     $b \leftarrow h$   
  end while  
  return  $a$   
end function
```



## Schlüsselwörter

- Anlehnung an höhere Programmiersprachen
- Vereinfachung zur Übersichtlichkeit (keine Klammern, Typen, etc.)
- ABER: **end** statements
- Können sowohl auf Deutsch als auch auf Englisch benutzt werden

# Pseudocode - Bausteine

## Methoden

```
function NAME (Param1, Param2, ...)  
    ...  
end function
```

## Zuweisungen

```
 $a := b$   
 $a \leftarrow b$  (Weist  $a$  den Wert  $b$  zu)  
  
 $a := b + c$  (Weist  $a$  die Summe aus  $b$  und  $c$  zu)  
 $A := B$  (Weist der Menge  $A$  die Menge  $B$  zu)
```

## Bedingungen

```
if Bedingung then  
    Aktion falls Bedingung wahr ist  
else  
    Aktion, falls Bedingung falsch ist  
end if
```

## Ausgaben / Rückgaben

```
print  $a$  (Gibt  $a$  aus)  
return  $a$  (Gibt  $a$  aus und beendet die Funktion)
```



# Pseudocode - Bausteine

## Schleifen - 1

**while** Bedingung **do**

Führe Aktion aus, *solange* eine Bedingung wahr ist.

**end while**

## Schleifen - 2

**repeat**

Führe Aktion aus, *bis* eine Bedingung wahr ist.

**until** Bedingung

## Schleifen - 3

**for**  $a$  in  $b, \dots, c$  **do**

Starte mit  $a := b$

$a$  wird nach jeder Iteration um 1 erhöht. ( $a := a + 1$ )

Wiederhole bis  $a$  den Wert  $c$  erreicht hat.

**end for**

## Schleifen - 4

**for**  $a := b$  **down to**  $c$  **do**

Starte mit  $a := b$

$a$  wird in jeder Iteration um 1 verringert. ( $a := a - 1$ )

Wiederhole bis  $a$  den Wert  $c$  erreicht hat.

**end for**

# Pseudocode - Varianten

- Auf Deutsch
  - Wenn ... Dann ... Sonst ...
  - Solange ...
  - Für  $i \leftarrow 1, \dots, k$  ...
- Statt  $\leftarrow$  schreibt man gelegentlich  $:=$  oder sogar  $=$
- Mischung aus Prosa und Pseudocode:
  1. **function** Verhältnis(*liste*)
  2.     Sortiere die Zahlen in *liste* aufsteigend
  3.     **return** *liste*[0] / *liste*[länge(*liste*)]
  4. **end function**
- end... statements dürfen weggelassen werden, aber Einrückung muss korrekt sein!

**Wichtig**  
Pseudocode muss immer  
verständlich und eindeutig sein!

... und konsistent!

# Problem und Instanz

Problem



Instanz

- Allgemeine Fragestellung
- Lösung: Angabe eines Algorithmus
- (Meist) Formuliert durch
  - Eingabe: Was ist gegeben?
  - Ausgabe: Was ist gesucht?

- Konkrete Fragestellung (Zahlen etc.)
- Lösung: Angabe einer konkreten Ausgabe
- Formuliert durch eine konkrete Eingabe eines bestimmten Problems

# Problem und Instanz Beispiel

Problem

**Größter gemeinsamer Teiler zweier Zahlen**

**Eingabe:** Zwei Zahlen  $a$  und  $b$ .

**Ausgabe:** Der größte gemeinsame Teiler  $q$  von  $a$  und  $b$ .

Lösung: Euklidischer Algorithmus (gerade gesehen)



Instanz

**Größter gemeinsamer Teiler zweier Zahlen**

- $ggT(5, 102)$ ; Lösung: 1
- $ggT(8, 64)$ ; Lösung: 8
- ...

# Pseudocode – In Aktion

1. **function** Mult( $a, b$ )
2.      $c \leftarrow 0$
3.     **for**  $d \leftarrow b$  **downto** 1
4.          $c \leftarrow c + a$
5.     **end for**
6.     **return**  $c$
7. **end function**

Variablen nach jeder Iteration der for-Schleife:

Iteration	$a$	$b$	$c$	$d$
-----------	-----	-----	-----	-----

Instanz: Berechne das Produkt von 9 und 3

Aufruf: Mult(9, 3)

# Pseudocode – In Aktion

1. **function** Mult( $a, b$ )
2.      $c \leftarrow 0$
3.     **for**  $d \leftarrow b$  **downto** 1
4.          $c \leftarrow c + a$
5.     **end for**
6.     **return**  $c$
7. **end function**

Instanz: Berechne das Produkt von 9 und 3

Aufruf: Mult(9, 3)

Rückgabe: 27



Variablen nach jeder Iteration der for-Schleife:

Iteration	$a$	$b$	$c$	$d$
1	9	3	9	3
2	9	3	18	2
3	9	3	27	1

\*Das ist hier nur eine Demo für Pseudocode, natürlich dürft ihr in eurem Pseudocode einfach direkt Multiplikationen hinschreiben.

# Pseudocode – In Aktion

1. **function** Euklid( $a, b$ )
2.     **while**  $b \neq 0$  **do**
3.          $h \leftarrow a \bmod b$
4.          $a \leftarrow b$
5.          $b \leftarrow h$
6.     **end while**
7.     **return**  $a$
8. **end function**

Instanz: Berechne ggT von  $a = 49, b = 21$

Aufruf: Euklid(49, 21)

Rückgabe: 7

Zeile	Iteration	$a$	$b$	$h$
1	-	49	21	-
2	-	49	21	-
3	1	49	21	7
4	1	21	21	7
5	1	21	7	7
6	1	21	7	7
2	1	21	7	7
3	2	21	7	0
4	2	7	7	0
5	2	7	0	0
6	2	7	0	0
2	2	7	0	0
7	2	7	0	0

# Pseudocode – In Aktion

1. **function** Absolute( $a$ )
2.     **if**  $a < 0$  **then**
3.          $a \leftarrow -a$
4.     **end if**
5.     **return**  $a$
6. **end function**

1. **function** AbsoluteSum( $a, b$ )
2.     **return** Absolute( $a$ ) + Absolute( $b$ )
3. **end function**

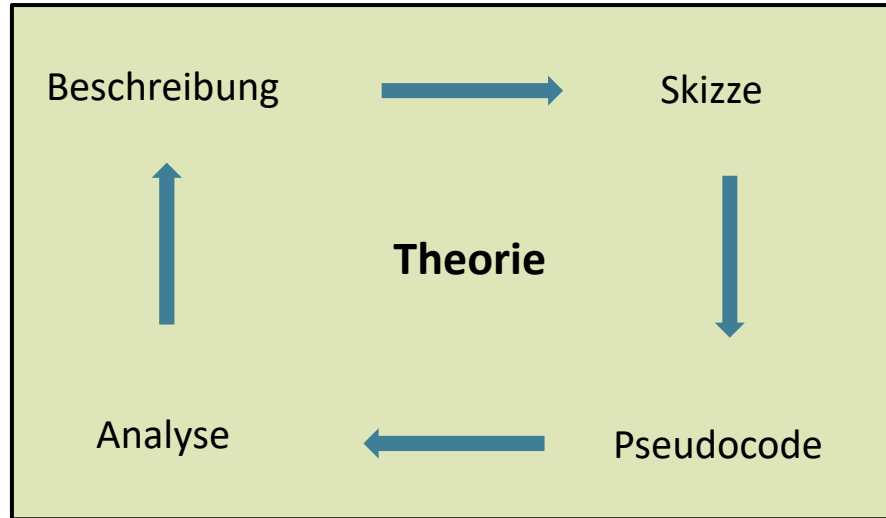
Aufruf: AbsoluteSum(-10, 3)

Rückgabe: 13

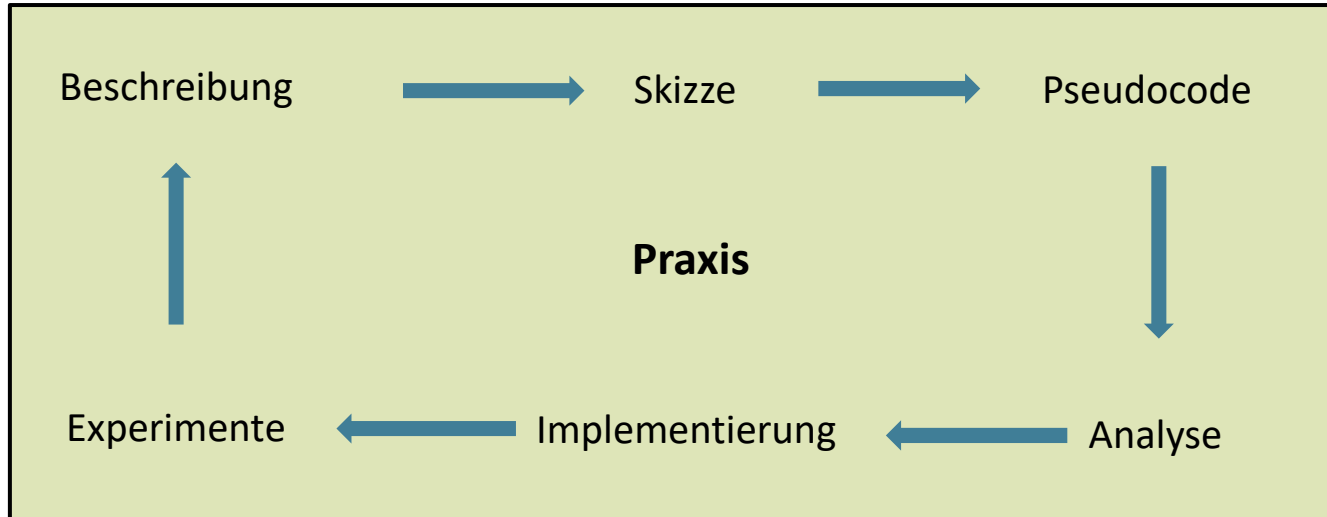
Zeile (AbsoluteSum)	Zeile (Absolute)	$a$ (AbsoluteSum)	$a$ (Absolute)	$b$
1	-	-10	-	3
2	1	-10	-10	3
2	2	-10	-10	3
2	3	-10	10	3
2	4	-10	10	3
2	5	-10	10	3
2	1	-10	3	3
2	2	-10	3	3
2	4	-10	3	3
2	5	-10	3	3



# Algorithmenentwurf in der Theorie



# Algorithmentwurf in der Praxis



# Ein einfacher Algorithmus für den Logarithmus

Betrachten wir den Logarithmus:

**Gegeben** Zwei (ganzzahlige) Zahlen  $n$  und  $b$

**Gesucht** Eine (nicht-ganzzahlige) Zahl  $x$ , sodass  $n = b^x$  (Oder:  $x = \log_b n$ )

*Einfach ausgedrückt: Wie oft muss man  $n$  durch  $b$  teilen, damit man auf 1 kommt?*

log	$n$
-----	-----

Betrachten wir zunächst den einfachen Fall:  $x$  ist ganzzahlig

1. **function** Log( $n, b$ )
2.      $\log \leftarrow 0$
3.     **while**  $n > 1$  **do**
4.          $n \leftarrow n/b$
5.          $\log \leftarrow \log + 1$
6.     **return**  $\log$

**Beispiel:** Log(64, 2)

# Ein einfacher Algorithmus für den Logarithmus

Betrachten wir den Logarithmus:

**Gegeben** Zwei (ganzzahlige) Zahlen  $n$  und  $b$

**Gesucht** Eine (nicht-ganzzahlige) Zahl  $x$ , sodass  $n = b^x$  (Oder:  $x = \log_b n$ )

*Einfach ausgedrückt: Wie oft muss man  $n$  durch  $b$  teilen, damit man auf 1 kommt?*

Betrachten wir zunächst den einfachen Fall:  $x$  ist ganzzahlig

1. **function** Log( $n, b$ )
2.      $\log \leftarrow 0$
3.     **while**  $n > 1$  **do**
4.          $n \leftarrow n/b$
5.          $\log \leftarrow \log + 1$
6.     **return**  $\log$

**Beispiel:** Log(64, 2)

log	$n$
0	64
1	32
2	16
3	8
4	4
5	2
6	1

# Ein Algorithmus für den Logarithmus

Was passiert, wenn  $x$  nicht ganzzahlig wird?

**Problem:** Irrationale Ergebnisse

**Lösung:** Bestimme den Logarithmus nur auf  $k$  Stellen genau.

Idee: Nutze den alten Algorithmus, indem wir die Stellen vor das Dezimalkomma verschieben.

$10^k \log_b n$  gibt uns  $k$  Stellen mehr vor dem Komma.

1. **function**  $\text{Log}(n, b, k)$
  2.      $n \leftarrow n^{(10^k)}$
  3.      $\text{log} \leftarrow \text{Log}(n, b)$
  4.     **return**  $\text{log} \cdot 10^{-k}$
  5. **end function**
- $$10^k \log_b n = \log_b(n^{(10^k)})$$

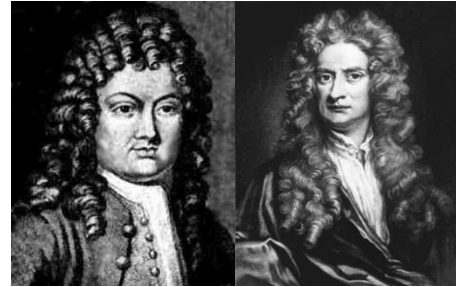
Problem 1: Schon für kleine  $k$  wird  $n$  sehr groß

Problem 2: Der Algorithmus benötigt exponentiell viele Schritte

# Mehr zu Logarithmen

Logarithmen lassen sich deutlich schneller und präziser berechnen.

- Taylor-Entwicklung (aus der Analysis)
- Newton-Verfahren (aus der Numerik)



Für diese Vorlesung ist die genaue Berechnung von (nicht ganzzahligen) Logarithmen nicht relevant.

Logarithmen werden aber dennoch benötigt, beispielsweise

- für Laufzeiten, z.B. beim Sortieren (Kapitel 5)
- zum Bestimmen der Höhe eines Suchbaumes (Kapitel 4)

**Wichtig:** Rechenregeln für Logarithmen!

# Zusammenfassung Pseudocode

<https://www.ibr.cs.tu-bs.de/alg/Merkzettel/pseudocode-booklet.pdf>



**Algorithmik**  
Technische Universität Braunschweig | Institut für Betriebssysteme und Rechnerverbund

## Merkzettel ▷ Pseudocode

Version – 12. April 2021

Pseudocode dient der übersichtlichen und eindeutigen Darstellung von Algorithmen. Er ist meist an moderne höhere Programmiersprachen angelehnt, kann aber auch mit natürlicher Sprache gemischt werden. Pseudocode verzichtet auf technische Besonderheiten realer Programmiersprachen und ist dadurch kompakter, sollte sich aber dennoch ohne viel Aufwand in eine reale Implementierung überführen lassen.

### Methoden

Funktionale Einheiten von Code werden zu benannten Blöcken (Funktionen) zusammengefasst.

- Funktions-Körper:  
`function NAME(Parameter1, Parameter2, ...)`  
...  
`end function`

# Fragen?



... nächstes Mal

# Beweistechniken

Logische Verknüpfungen

Negation („Nicht“, $\neg$ )	Konjunktion („Und“, $\wedge$ )	Disjunktion („Oder“, $\vee$ )	Implikation („wenn... dann“, $\Rightarrow$ )	Äquivalenz („genau dann wenn“, $\Leftrightarrow$ )		
$A$	$B$	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

1: Aussage ist wahr  
0: Aussage ist falsch



**Beweise – Direkter Beweis**

**Satz des Pythagoras:** Für ein rechtwinkliges Dreieck  $D$  mit Kathetenlängen  $x$  und  $y$  und Hypotenusenlänge  $z$  gilt  $x^2 + y^2 = z^2$ .

Beweis: Betrachte folgende Konstrukte:

Beides sind Quadrate mit Seitenlänge  $x + y$   
Also gilt  $x^2 + y^2 + 4 \cdot \text{Area}(D) = z^2 + 4 \cdot \text{Area}(D)$   
 $\Rightarrow x^2 + y^2 = z^2$

Technische Universität Braunschweig | Ramin Kosfeld und Chek-Manh Loi | 18.11.2023 | Übung 1 | Seite 27

**Beweise – Direkter Beweis**

Aussagen oft in der Form  $A \Rightarrow B$ . Unterscheide zwischen **Voraussetzung** ( $A$ ) und **Schlussfolgerung** ( $B$ )

Wird die Schlussfolgerung durch eine logische Folgerungskette aus den Voraussetzungen hergeleitet, so spricht man von einem **direkten Beweis**.

Beispiel:  
Wenn  $x, y \in \mathbb{R}$  und  $x, y \geq 0$ , dann gilt  $\sqrt{xy} \leq \frac{x+y}{2}$

Technische Universität Braunschweig | Ramin Kosfeld und Chek-Manh Loi | 18.11.2023 | Übung 1 | Seite 18

... in 2 Wochen, am 16.11.!