



Technische  
Universität  
Braunschweig



# Algorithmen und Datenstrukturen

## Große Übung 0

Matthias Konitzny, Arne Schmidt

04.11.2021

# Organisation

# Homepage und Anmeldung

- Veranstaltungsübersicht:

<https://www.ibr.cs.tu-bs.de/courses/ws2122/aud/index.html>

The screenshot shows the course page for 'Algorithmen und Datenstrukturen' on the website of the Institute for Business Systems and Computing (IBR) at TU Braunschweig. The page includes a navigation menu, a search bar, and a sidebar with a table of contents. The main content area displays the semester (Wintersemester 2021/2022), study programs, IBR group (ALG), and the course type (Vorlesung/Übung). It lists the lecturer, Prof. Dr. Sándor P. Fekete, and two assistants, Matthias Konitzny and Dr. Arne Schmidt, with their contact information. The page also shows the course level (LP 8), SWS (4-2-2), and the location and time of the lectures and exercises.

Technische Universität Braunschweig

Studium & Lehre    Forschung    International    Die TU Braunschweig    Struktur    Q

Technische Universität Braunschweig > Struktur > Fakultäten > Carl-Neuberg-Fakultät > Institute  
> Institut für Betriebssysteme und Rechnerverbund

Institut für Betriebssysteme und Rechnerverbund

IBR Login

Algorithmen und Datenstrukturen

Semester: Wintersemester 2021/2022 -  
Studiengänge: Bachelor Wirtschaftsinformatik, Bachelor Informations-Systemtechnik, Bachelor Informatik  
IBR Gruppe: ALG (Prof. Fekete)  
Art: Vorlesung/Übung

Dozent

Prof. Dr. Sándor P. Fekete  
Abteilungsleiter  
s.p.fekete@tu-bs.de  
+49 531 3913111  
Raum 335

Assistenten

Matthias Konitzny  
Wissenschaftlicher Mitarbeiter  
m.konitzny@ibr.cs.tu-bs.de  
+49 531 3913113  
Raum 333

Dr. Arne Schmidt  
Wissenschaftlicher Mitarbeiter  
a.schmidt@ibr.cs.tu-bs.de  
+49 531 3913115  
Raum 319

LP 8  
SWS 4-2-2

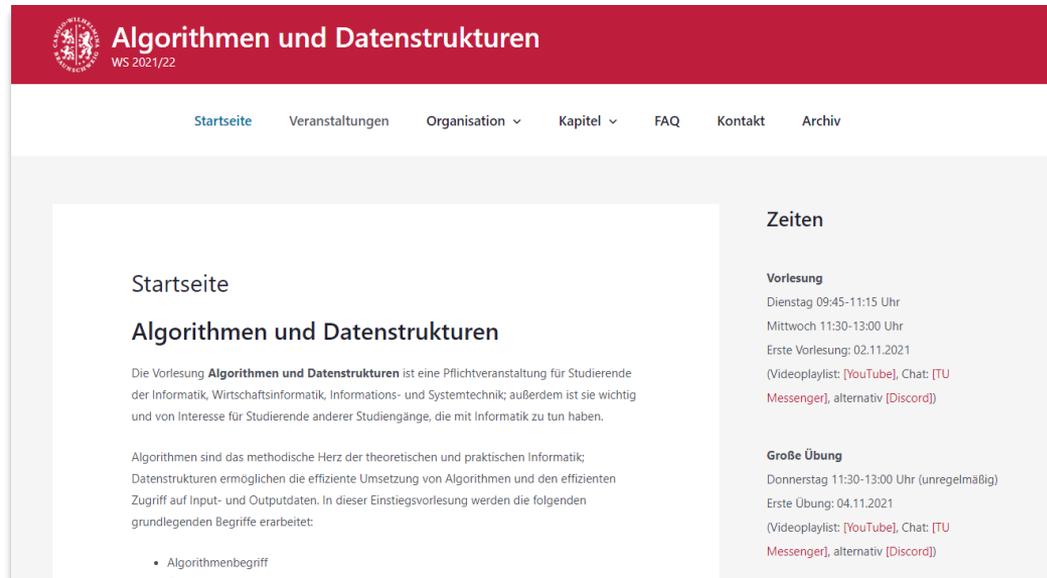
Ort & Zeit: Vorlesung (online): Dienstag, 09:45 - 11:15 und Mittwoch, 11:30 - 13:00  
Große Übung (online): Donnerstag, 11:30 - 13:00 (unregelmäßig)

News  
Wir über uns  
Connected and Mobile Systems  
Verteilte Systeme  
Algorithmik  
Mikroprozessoriabor  
Studium  
Wintersemester 2021/2022  
Sommersemester 2021  
Abschlussarbeiten  
Service  
Spin-Offs  
Forschungsverbünde

# Homepage und Anmeldung

- Folien, Hausaufgaben, Vorlesungsvideos, Altklausuren, Übungsanmeldungen:

<https://www.aud.ibr.cs.tu-bs.de>

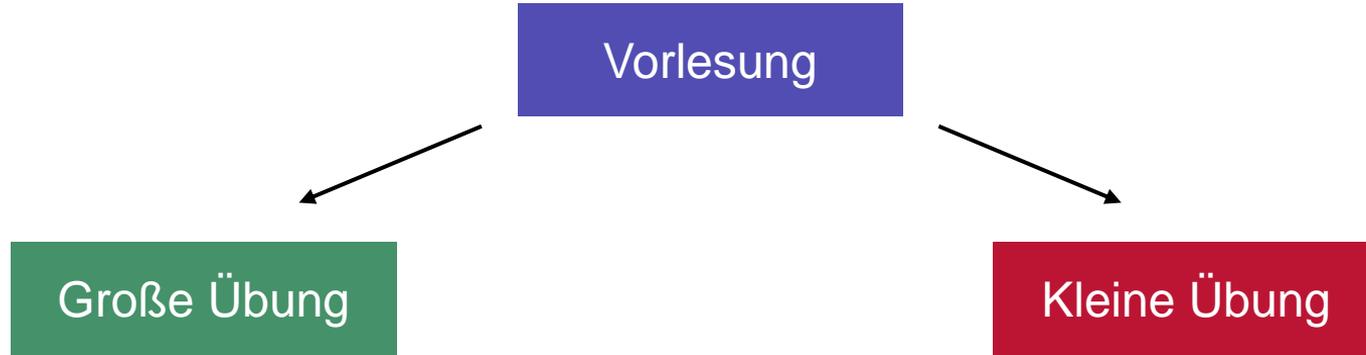


The screenshot shows the homepage of the 'Algorithmen und Datenstrukturen' course. The header is dark red with the course title and 'WS 2021/22'. A navigation menu includes 'Startseite', 'Veranstaltungen', 'Organisation', 'Kapitel', 'FAQ', 'Kontakt', and 'Archiv'. The main content area has a 'Startseite' heading, followed by the course title and a paragraph describing the course as a mandatory event for students of Informatics, Business Informatics, Information Systems, and System Technology. It also mentions that it is for students of other programs who are interested in Informatics. Below this, it states that algorithms are the methodical heart of theoretical and practical Informatics, and data structures enable the efficient implementation of algorithms and efficient access to input and output data. A list of topics is partially visible, starting with 'Algorithmenbegriff'. On the right, a sidebar titled 'Zeiten' lists lecture and exercise times: 'Vorlesung' on Tuesday (09:45-11:15) and Wednesday (11:30-13:00), with the first lecture on 02.11.2021; and 'Große Übung' on Thursday (11:30-13:00, irregular), with the first exercise on 04.11.2021. Both sections provide links for video playlists, chat, and messaging.

# Semesterkalender

<https://aud.ibr.cs.tu-bs.de/events/>

The image displays four overlapping calendar views from the semester calendar application. Each calendar shows a grid of days with scheduled events. The events are color-coded: blue for lectures (Vorlesung) and green for exercises (Große Übung or Kleine Übung). Red horizontal bars indicate multi-day events like 'Kleine Übungen'. The interface includes a search bar at the top of each calendar, navigation arrows, and a 'Heute' (Today) indicator. The first calendar is for November 2021, the second for December 2021, the third for January 2022, and the fourth for February 2022. The fourth calendar also features buttons for 'Finde', 'Liste', 'Monat', and 'Tag'.



- Aufarbeitung der Inhalte
- Exkurse
- Beantwortung von Fragen
- Interaktion!

- Vertiefung der Inhalte
- Selbständiges Arbeiten
- Besprechung von Hausaufgaben

# Hausaufgaben und Übungsblätter

## 5 verpflichtende Hausaufgabenblätter

- Studienleistung
- 20 Punkte pro Blatt

## 6 freiwillige Übungsblätter

- 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  
Matthias Konitzny  
Dr. Arne Schmidt

Winter 2020/21  
Abgabe: 16.11.2020  
Rückgabe: 23.–27.11.2020

### Übungsblatt 1

Abgabe der Lösungen muss bis zum 16.11.2020 um 14:00 Uhr erfolgen. Lösungen müssen per Mail mit einer pdf-Datei (Name der Datei „blatt.[nr]\_[matrikel].pdf“) an den jeweiligen Tutor geschickt werden. Email-Adressen sind unter <https://www.ibr.cs.tu-bs.de/alg/index.html> zu finden.

*Beachte:* Bei der Bearbeitung der Hausaufgaben gelten folgenden Richtlinien:  
<https://www.ibr.cs.tu-bs.de/courses/ws2021/aud/HA-Hinweise.pdf>

Hausaufgabe 1 (Kantengraph): (3+5+5 Punkte)  
Der Kantengraph  $L(G) = (V_L, E_L)$  eines Graphen  $G = (V, E)$  besitzt für jede Kante  $e \in E$

# 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
- Anhang vergessen / Falsche Mailadresse: 0 Punkte
- Zusammen überlegen, **ABER:** einzeln aufschreiben und abgeben, sonst: 0 Punkte.

# Hausaufgaben

L<sup>A</sup>T<sub>E</sub>X



- Erstellen der elektronischen Lösung
  - Einscannen/Abfotografieren einer auf Papier geschriebenen Lösung
  - TeX, Word, LibreOffice, etc. (ggf. umständlich für Formeln)
- Abgabe per Mail
  - Abgaben nur im PDF-Format.
  - Dateiname `blatt_[nr]_[name]_[matrikel].pdf` (Beispiel: `blatt_1_Matthias Konitzny_7654321.pdf`)
  - **Keine** Filehosting-Dienste (Dropbox, Google-Drive, etc)
  - Nutzt Dateikompression (die Abgaben sollten  $\leq 5MB$  groß sein)

# Klausur

Der Termin für die Klausur wird später auf der Website bekannt gegeben.

Dasselbe gilt für weitere Informationen wie

- Raumaufteilung
- Beginn der Klausur



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 über Homepage
- Für Informationen wie
  - Raumänderungen,
  - Ausfälle,
  - Etc.
- Möglichkeit für Fragen

## Subscribing to Aud

Subscribe to Aud by filling out the following form. You will be sent email requesting confirmation, to prevent others from gratuitously subscribing you. Once confirmation is received, your request will be held for approval by the list moderator. You will be notified of the moderator's decision by email. This is also a hidden list, which means that the list of members is available only to the list administrator.

Your email address:

Your name (optional):

You may enter a privacy password below. This provides only mild security, but should prevent others from messing with your subscription. **Do not use a valuable password** as it will occasionally be emailed back to you in cleartext.

If you choose not to enter a password, one will be automatically generated for you, and it will be sent to you once you've confirmed your subscription. You can always request a mail-back of your password when you edit your personal options.

Pick a password:

Reenter password to confirm:

Which language do you prefer to display your messages? English (USA)

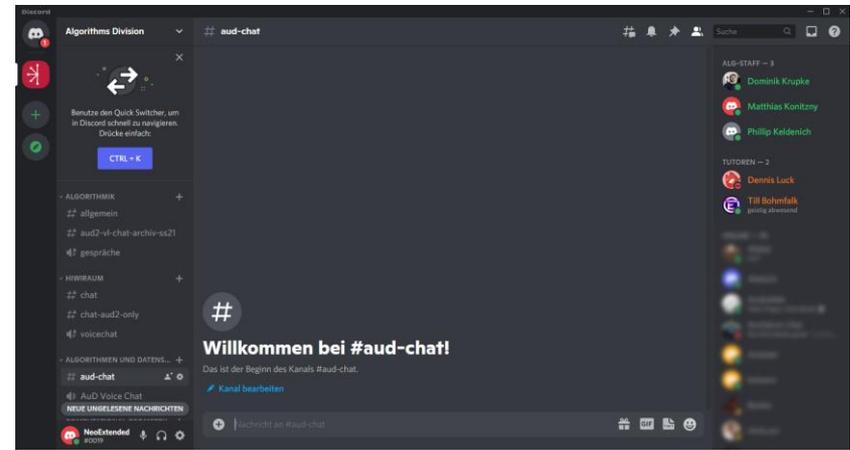
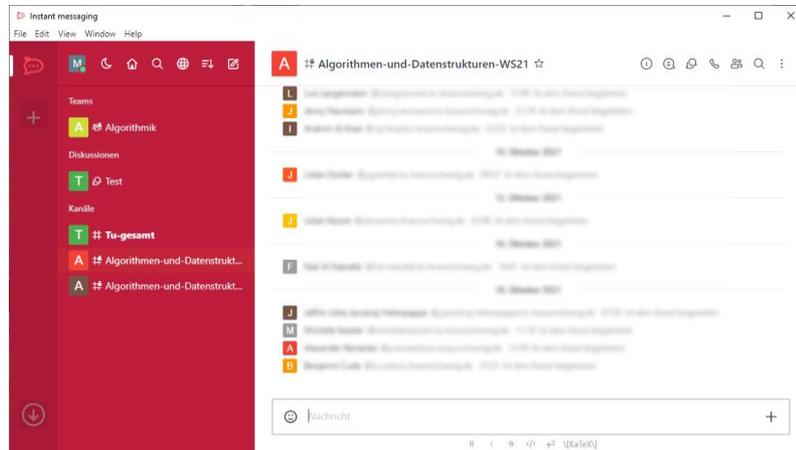
# Messenger



TU-Messenger (Rocket.Chat)



Discord



Kommunikation während der Vorlesung und Übung und Austausch mit Mitstudierenden.

# Fragen?

# Pseudocode

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

Me: \*starts programming without  
writing pseudocode first\*

My university:



# 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

vereinbartes Inventar sprachlicher Zeichen und Regeln zu ihrer Verknüpfung

# Warum Pseudocode?

```
public class HalloWelt {  
    public static void main(String[] args) {  
        System.out.println("Hallo Welt!");  
    }  
}
```

Ist dieser Code effizient zu lesen?

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

# Warum Pseudocode?

Java

```
public class HalloWelt {  
    public static void main(String[] args) {  
        System.out.println("Hallo Welt!");  
    }  
}
```

Pseudocode

```
function main(args)  
    print Hallo Welt!  
end function
```

# Problem und Instanz

## Problem



## Instanz

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

- Konkrete Fragestellung
- Lösung: Angabe einer konkreten Ausgabe
- Formuliert durch eine konkrete Eingabe eines bestimmten Problems

# Problem und Instanz Beispiel

Problem



Instanz

**Größter gemeinsamer Teiler  
zweier Zahlen**

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

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

Lösung: Euklidischer Algorithmus  
(gleich mehr)

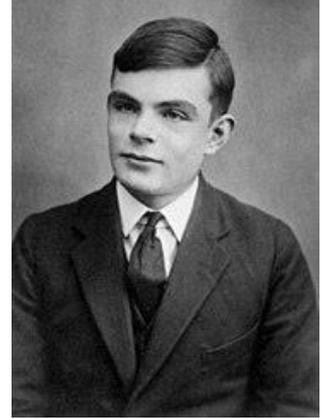
**Größter gemeinsamer Teiler  
zweier Zahlen**

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

# Algorithmus

Eindeutige Handlungsvorschrift zur Lösung eines Problems

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



Alan Turing

# Beispiel: Euklidischer Algorithmus

## 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.

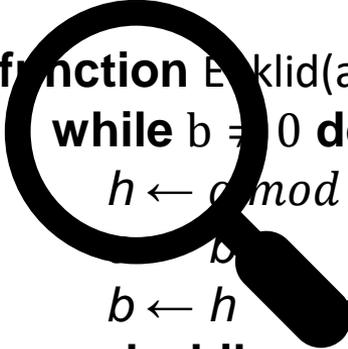
## Pseudocode

```
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 ≠ 0 do
    h ← a mod b
    b ← 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

```
 $a := b$  (Weist a den Wert b zu)  
 $a \leftarrow b$ 
```

```
 $a := b + c$  (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**  $c$

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 – 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** Mult( $a, b$ )
2.      $c \leftarrow 0$
3.     **for**  $d := b$  **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
-----------	---	---	---	---

# Pseudocode – In Aktion

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

1. **function** AbsoluteDiff( $a, b$ )
2.     **return** Absolute( $a$ ) – Absolute( $b$ )
3. **end function**

Aufruf: AbsoluteDiff(-10, 3)

Rückgabe: 7

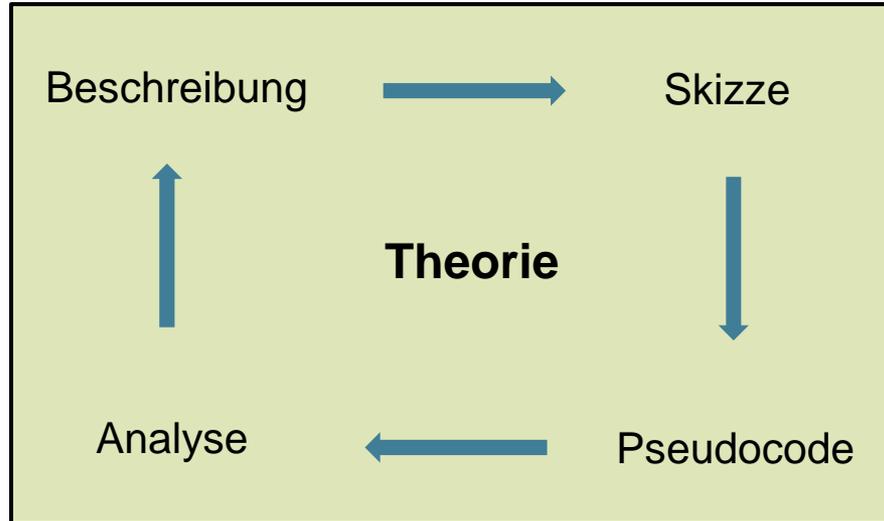
Zeile (AbsoluteDiff)	Zeile (Absolute)	a (AbsoluteDiff)	a (Absolute)	b
1	-	-10	-	3
2	1	-10	-10	3
2	2	-10	-10	3
2	3	-10	10	3
2	4	-10	10	3
2	5	-10	10	3
2	1	-10	3	3
2	2	-10	3	3
2	4	-10	3	3
2	5	-10	3	3

# Pseudocode - Varianten

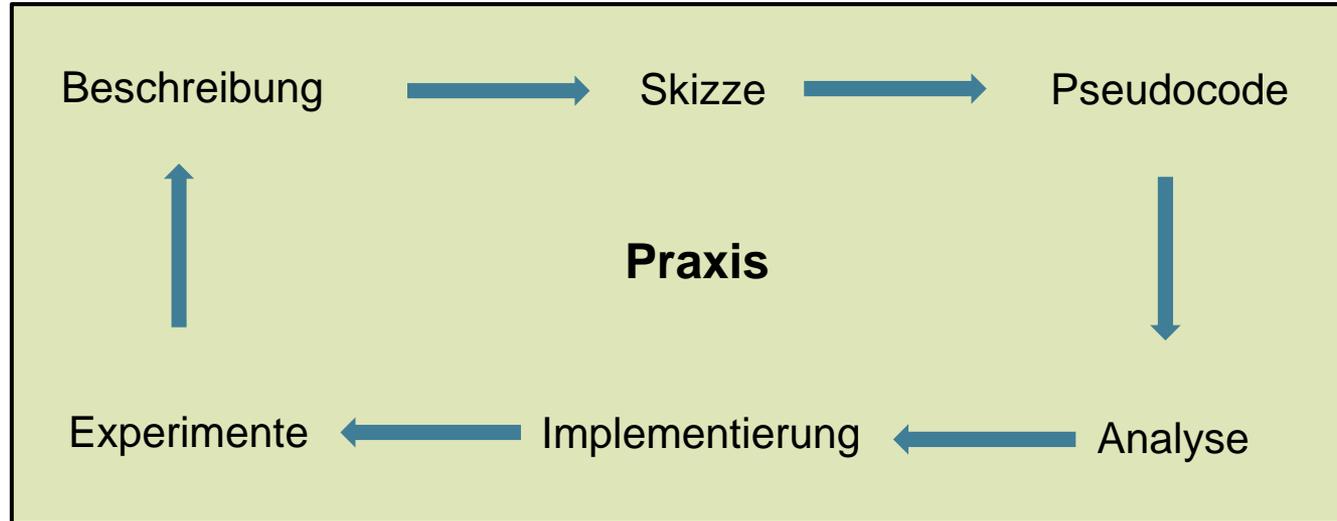
- Auf Deutsch
  - Wenn ... Dann ... Sonst ...
  - Solange ...
  - Für  $i \leftarrow 1, \dots, k$  ...
- Statt  $\leftarrow$  schreibt man gelegentlich  $:=$  oder sogar  $=$
- Mischung aus Prosa und Pseudocode:
  1. **function** Verhältnis(*liste*)
  2.     Sortiere die Zahlen in *liste* aufsteigend
  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!

# Algorithmenentwurf in der Theorie



# Algorithmenentwurf in der Theorie



# 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.          $\log \leftarrow \log + 1$
5.          $n \leftarrow n/b$
6.     **return**  $\log$

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

<b>log</b>	0	1	2	3	4	5	6
<b>n</b>	64	32	16	8	4	2	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.  $n \leftarrow n^{(10^k)}$       #  $10^k \log_b n = \log_b(n^{(10^k)})$
3.  $\text{log} \leftarrow \text{Log}(n, b)$
4. **return**  $\text{log} \cdot 10^{-k}$
5. **end function**

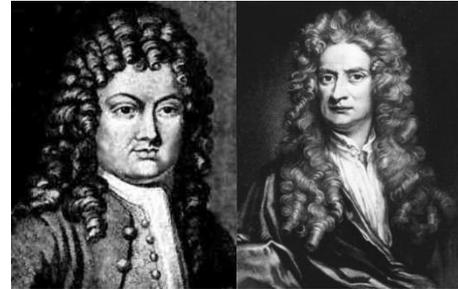
Problem 1: Schon für kleine  $k$  wird  $n$  sehr groß

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 Logarithmen nicht relevant.

Logarithmen werden aber dennoch benötigt, beispielsweise

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

# Fragen?