Technische Universität Braunschweig

Algorithmen und Datenstrukturen
Große Übung 0

Ramin Kosfeld und Chek-Manh Loi

02.11.2023

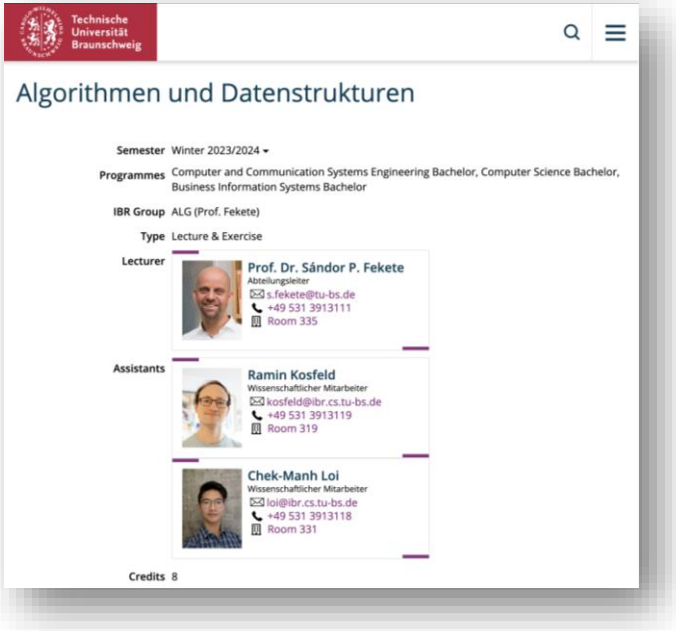
Organisation

Homepage und Anmeldung

Stud.IP?

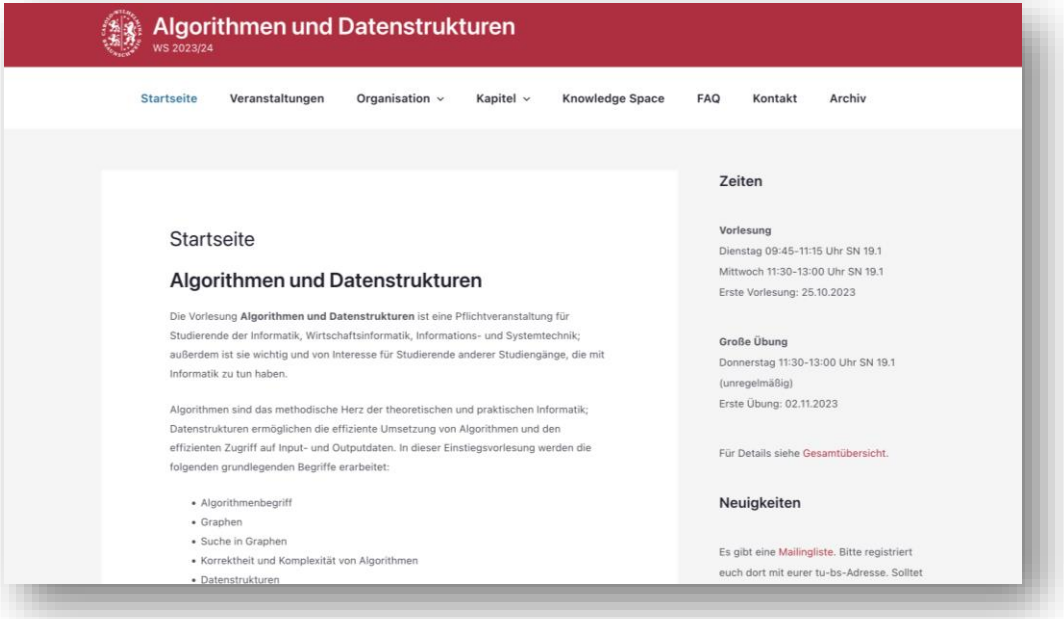
Homepage und Anmeldung

Veranstaltungsübersicht:
ibr.cs.tu-bs.de/courses/ws2324/aud/index.html



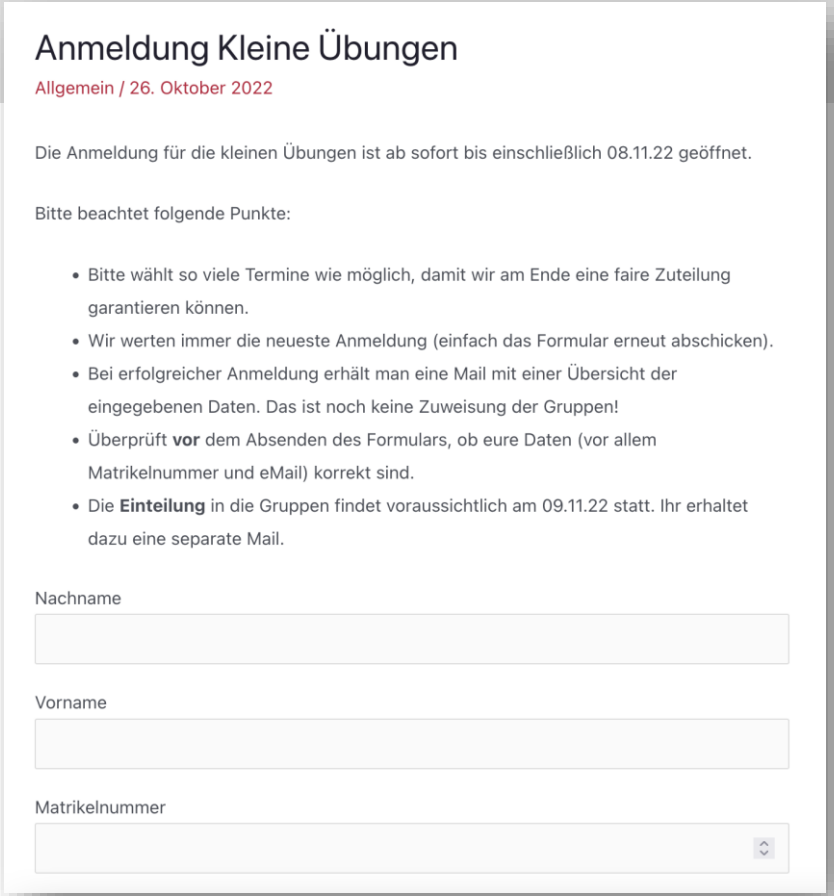
Homepage und Anmeldung

Die eigentliche Kursseite: *aud.ibr.cs.tu-bs.de*
… Folien, Hausaufgaben, Vorlesungsvideos, Altklausuren, **Übungsanmeldungen**:



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

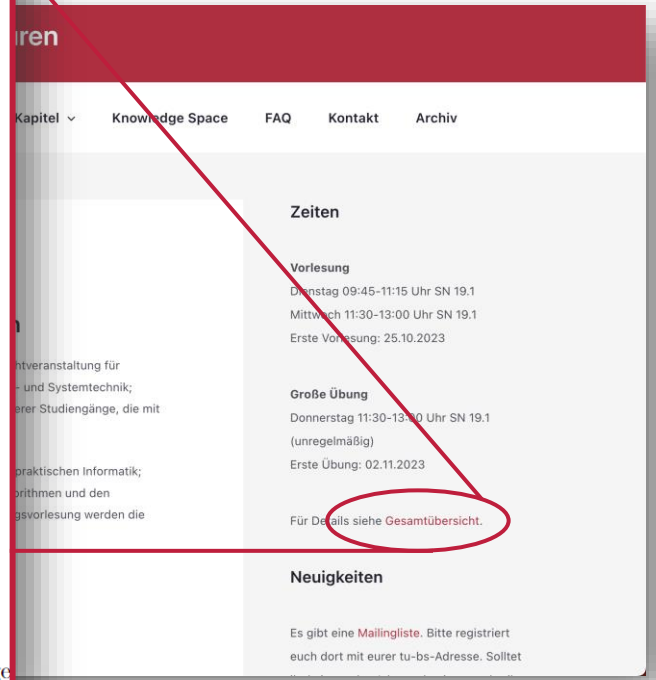
Matrikelnummer

Semesterplan

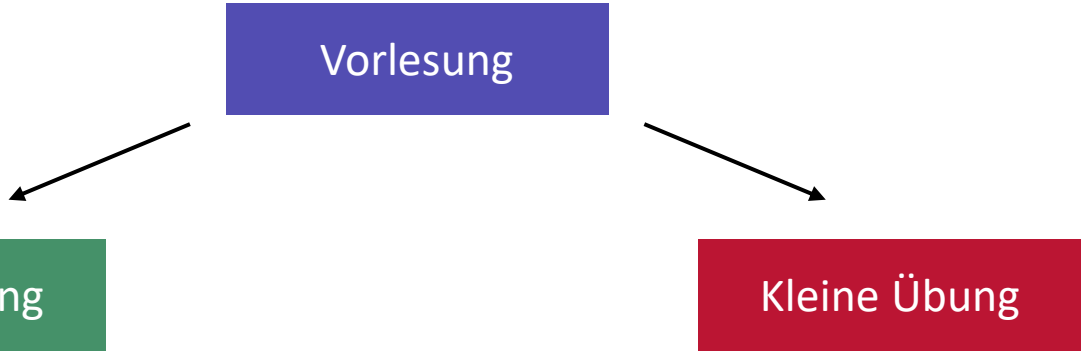
Semesterplan Algorithmen und Datenstrukturen WS23/24

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.						
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

…en kurzfristig Änderungen im Zeitplan notwendig sein, werden diese über die Mailingliste bekannt gege…



Formate



<u>Tafelübungen im Hörsaal (mit allen)</u>

- Aufarbeitung der Inhalte
- Beantwortung von Fragen
- Interaktion!

<u>Kleingruppen (Seminarräume)</u>

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

Hausaufgaben und Übungsblätter

5 verpflichtende Hausaufgabenblätter

- Studienleistung
- 20 Punkte pro Blatt
- Einzelabgabe

6 freiwillige Übungsblätter

- Zusätzliche Vertiefung
- Prüfungsvorbereitung

Algorithmen und Datenstrukturen **Winter 2023/24**
Prof. Dr. Sándor P. Fekete
Ramin Kosfeld
Chek-Manh Loi
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!

- 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.

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:** Einzeln 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 Wintersemester 2019/2020
Institut für Betriebssysteme und Rechnerverbund
Abteilung Algorithmik

Prof. Dr. Sándor P. Fekete
Arne Schmidt

Klausur
Algorithmen und Datenstrukturen
28.02.2020

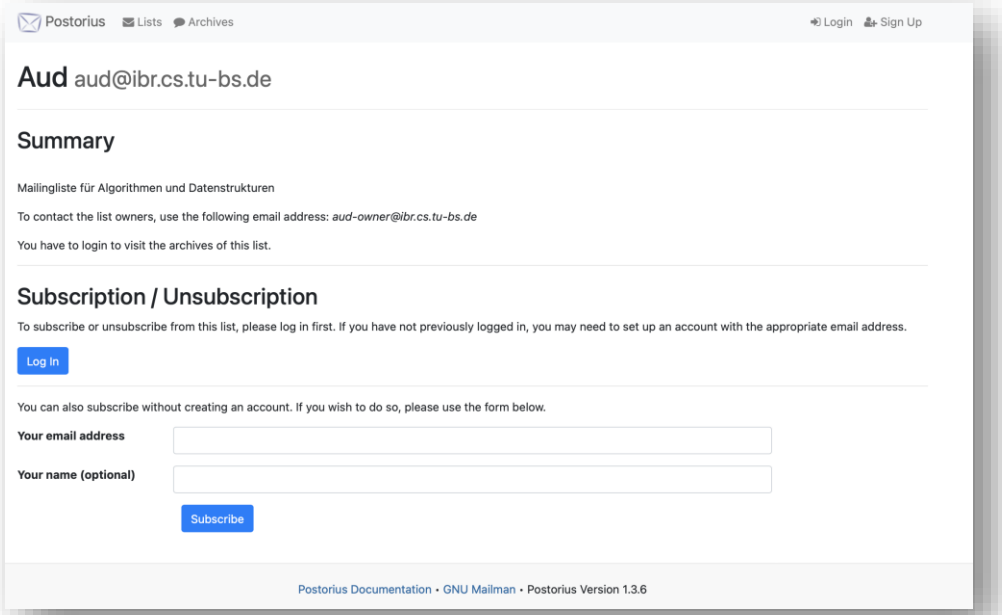
Name: .....
Vorname: .....
Matr.-Nr.: .....
Studiengang: .....
☐ Bachelor ☐ Master ☐ Andere

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

Muster

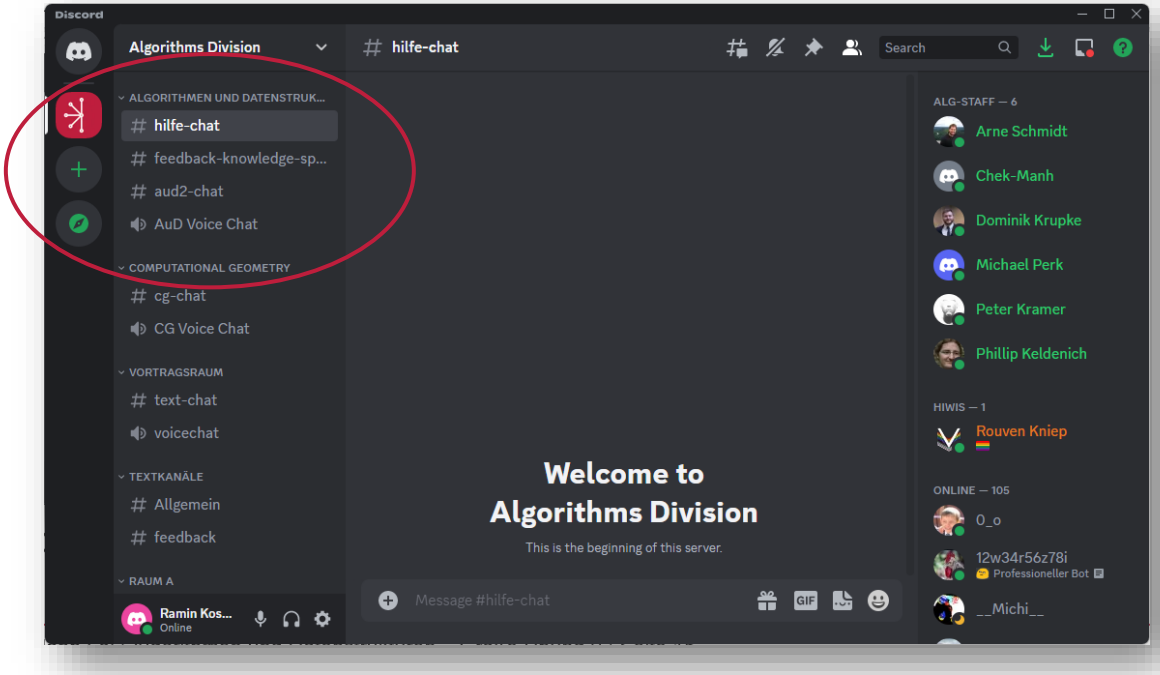
Mailingliste

- Anmeldung hier oder über Homepage
- Für Informationen wie
  - Raumänderungen,
  - Ausfälle,
  - etc.
- Möglichkeit für Fragen



Discord

Discord



Kommunikation abseits der Vorlesung und Austausch mit Mitstudierenden.

Fragen

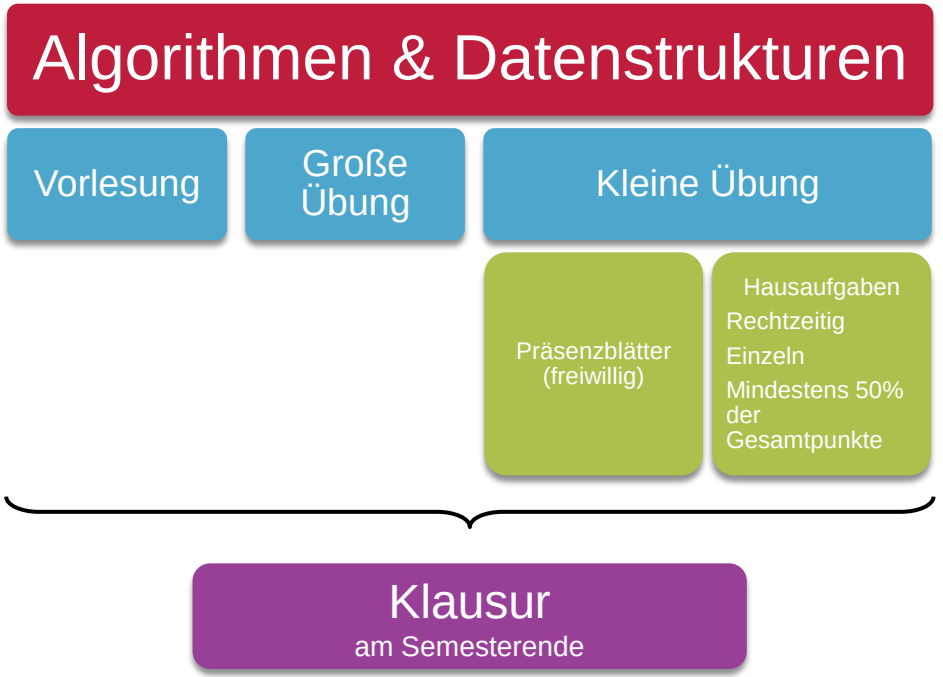
Stellt Eure Fragen …

- Während oder nach den Vorlesungen
- Euren Tutoren in den kleinen Übungen!
- Über die Mailingliste (ins Plenum)
- Per Mail (kosfeld@ibr.cs.tu-bs.de & 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)

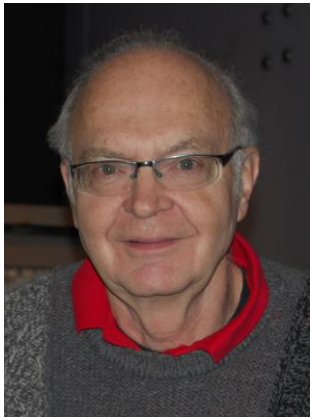


Fragen?

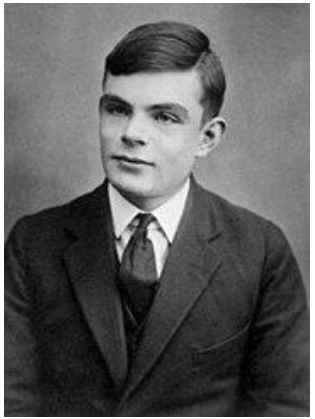
Algorithmus

Eindeutige Handlungsvorschrift zur Lösung eines Problems

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

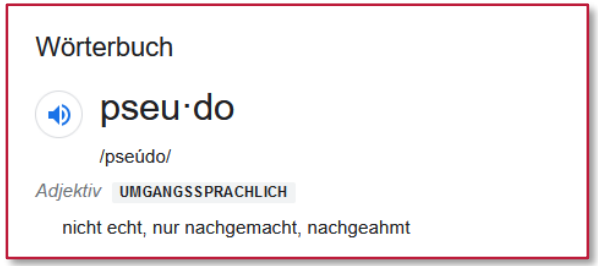


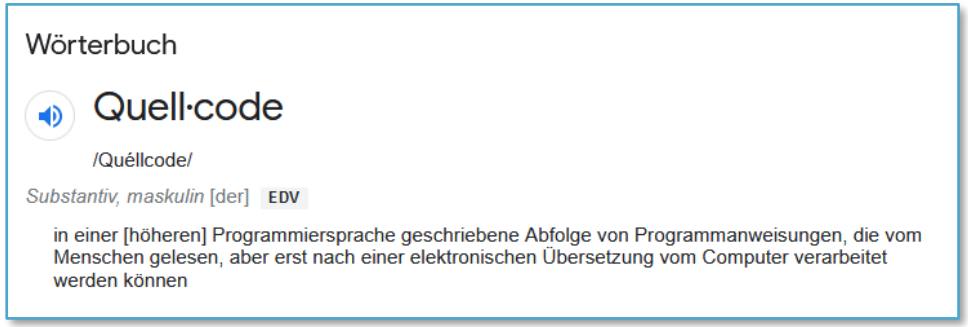
Donald Knuth



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?

```java
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

```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 $r = a \, mod \, b$, sodass $a = q \cdot b + r$ für ein $q \, \epsilon \, \mathbb{Z}$ 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

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 $i \leftarrow 1, \ldots, k$ …
- Statt $\leftarrow$ schreibt man gelegentlich $:=$ oder sogar $=$

Wichtig
Pseudocode muss immer verständlich und eindeutig sein!

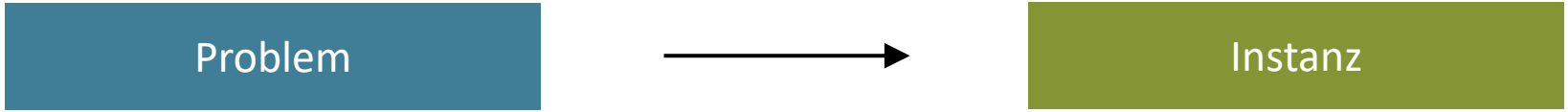
… und konsistent!

- 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!

Problem und 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 → 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 (gerade gesehen)

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**

Instanz: Berechne das Produkt von 9 und 3
Aufruf: Mult(9, 3)

Variablen nach jeder Iteration der for-Schleife:

Iteration	a	b	c	d

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. $h \leftarrow a \; mod \; b$
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**

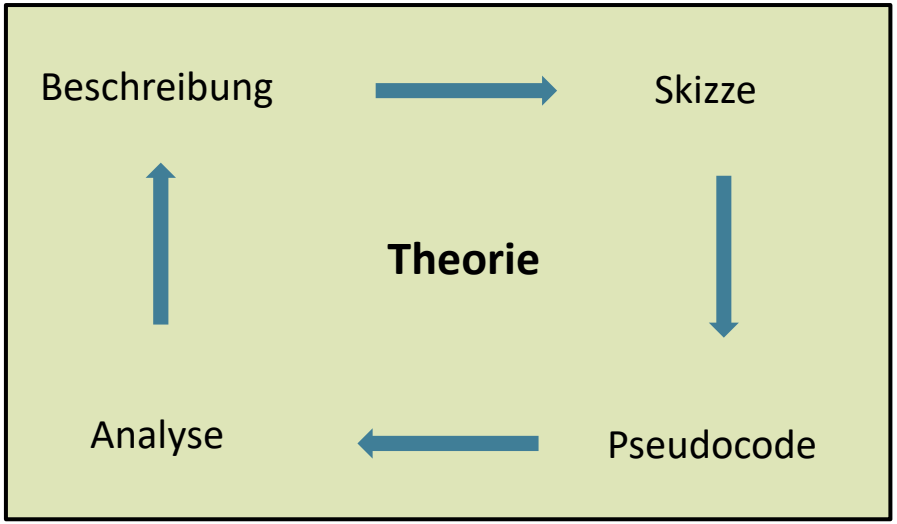
1. **function** AbsoluteSum(a, b)
2. **return** Absolute(a) + Absolute(b)
3. **end function**

Aufruf: AbsoluteSum(-10, 3)

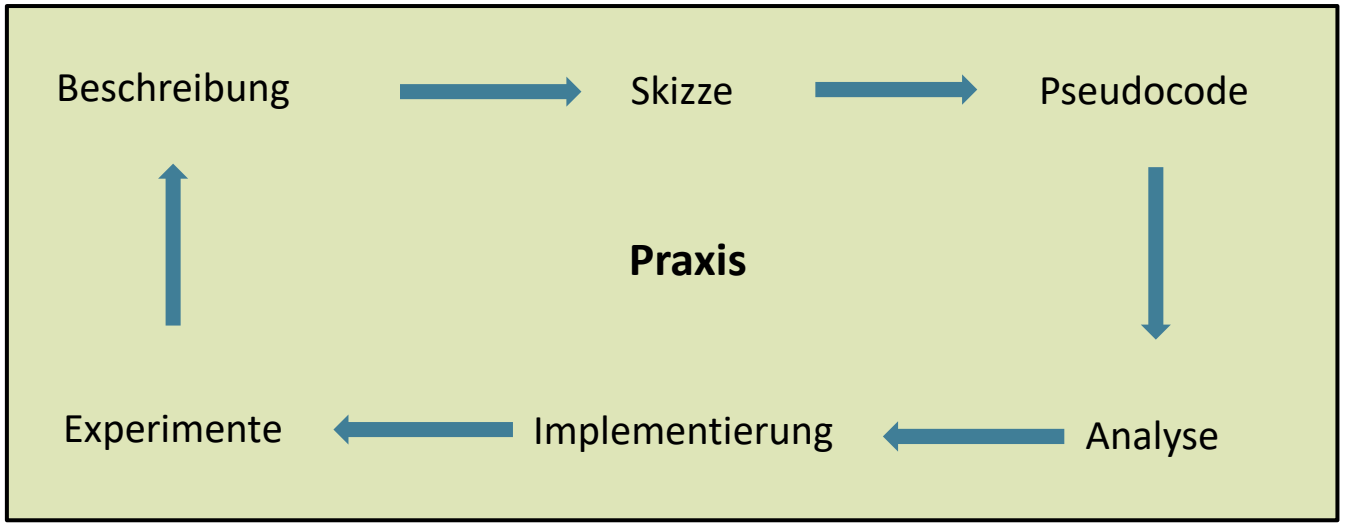
Rückgabe: 13

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



Algorithmenentwurf in der Praxis



Ein einfacher Algorithmus für den Logarithmus

log	n

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 ← 0
3.      while n > 1 do
4.          n ← n/b
5.          log ← 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 ← 0
3.      while n > 1 do
4.          n ← n/b
5.          log ← 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.

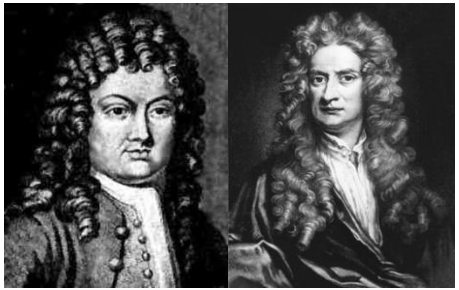
1. **function** Log(n, b, k)
2. $n \leftarrow n^{(10^k)}$
3. log $\leftarrow$ Log(n, b)
4. **return** log $\cdot\, 10^{-k}$
5. **end function**

$$10^k \log_b n = \log_b(n^{(10^k)})$$

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

… in 2 Wochen, am 16.11.!