



Programmiertechnik II

Algorithmen

Ralf Herbrich



Egal was ist, du kannst uns anrufen!

Wir führen unsere Gespräche wertschätzend und unvoreingenommen.
Du bleibst dabei **anonym**.

dienstags, mittwochs, donnerstags und sonntags
von 21 bis 24 Uhr
unter 0331 977 1834 oder im Chat

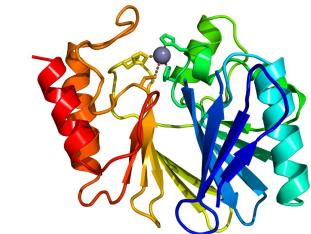
Weitere Infos:



<https://nightline-potsdam.de>

Warum Algorithmen studieren?

Wissenschaft	Probleme die Algorithmen benötigen
Netzwerk	Websuche, <i>packet routing</i> , verteilte Dateisysteme
Rechnerarchitektur	Schaltkreisentwurf
Betriebssysteme	Dateisysteme, Compiler
<i>Cybersecurity</i>	Verschlüsselungen
Computergrafik	<i>Virtual Reality</i>
Kommunikation	Empfehlungen, Nachrichtenfeeds, Werbung
Biologie	<i>protein folding</i>



Programmiertechnik II

Unit 1 - Algorithmen

Warum Algorithmen studieren?

■ Beschäftigung des Intellekts

- *"For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing." — Francis Sullivan*
- *"An algorithm must be seen to be believed." — Donald Knuth*



Francis Sullivan



Donald Knuth
(1938 –)

■ Grundfähigkeit von Informatikern

- *"I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships." — Linus Torvalds*
- *"Algorithms + Data Structures = Programs." — Niklaus Wirth*



Linus Torvalds
(1969 –)



Niklaus Wirth
(1934 – 2024)

Programmiertechnik II

Unit 1 - Algorithmen

Überblick

1. Einordnung der Lehrveranstaltung
2. Algorithmenbegriff
3. Eigenschaften von Algorithmen
4. Historie von Algorithmen

Überblick

- 1. Einordnung der Lehrveranstaltung**
2. Algorithmenbegriff
3. Eigenschaften von Algorithmen
4. Historie von Algorithmen

Informatik am HPI

- *Bachelor of Science* (6 Semester)
- *Master of Science* (4 Semester)
- Doktorandenprogramm (2-3 Jahre)

	1. Semester	2. Semester	3. Semester	4. Semester	5. Semester	6. Semester
6 LP	HPI-PT1 Programmiertechnik I	HPI-PT2 Programmiertechnik II	HPI-BS Betriebssysteme	HPI-SWT Softwaretechnik	HPI-PEM Projektentwicklung und -management	
6 LP	HPI-MO Modellierung	HPI-SB Software-basisssysteme	HPI-TI1 Theoretische Informatik I	HPI-TI2 Theoretische Informatik II	HPI-VT1-V 1. Vertiefungsgebiet	HPI-BA Bachelorarbeit
6 LP	HPI-DS Digital Systeme	HPI-SB Software-basisssysteme	HPI-SWA Software-Architektur	HPI-VT1-G 1. Vertiefungsgebiet	HPI-VT2-V 2. Vertiefungsgebiet	
6 LP	HPI-MA1 Mathematik I	HPI-MA2 Mathematik II	HPI-MA3 Mathematik III	HPI-VT2-G 2. Vertiefungsgebiet		Softwareprojekttätigkeit
6 LP	HPI-RG Rechtliche Grundlagen I	HPI-RG Rechtliche Grundlagen II	HPI-SB Software-basisssysteme	HPI-PSK / DTH Professional-Skills / Design Thinking		
	HPI-WG Wirtschaftliche Grundlagen I	HPI-WG Wirtschaftliche Grundlagen II				

Programmiertechnik II

Unit 1 - Algorithmen

Ziel & Themen

- **Mein Ziel:** Begeisterung und Verständnis für Algorithmen und Datenstrukturen zu entwickeln

- **Themenübersicht**

1. C++ Einführung
2. Analyse von Algorithmen
3. Grundlegende Datentypen
4. Sortieren
5. Suchen
6. *Hashing*
7. Bäume
8. Graphen
9. Algorithmenparadigmen

Programmiertechnik II

Unit 1 - Algorithmen

Format

- **Moodle:** <https://moodle.hpi.de/course/view.php?id=873>
 - Alle Termine und Vorlesungen + Zusatzmaterial verfügbar
 - Fragen und Antworten im Forum auf der Hauptseite
- **GitHub:** <https://github.com/HPI-Artificial-Intelligence-Teaching/25-pt2/>
 - Alle Quelltexte, Folien und ergänzendes Material wird hier auch hochgeladen
- **Vorlesung:**
 - Mo 13:30-15:00, HS 1
 - Do 9:15-10:45, HS 1
- **Übung:**
 - (Approximative) jede vierte Vorlesung ist eine Übung (= jeden zweiten Freitag)
 - Übungsgruppen bilden (2 Mitglieder) bis zum **13. April**
 - Abgabe der Übungen über Github Classrooms
 - Programmieraufgaben als Code
 - Schreibaufgaben als Textdatei, Markdown oder PDF

■ Klausurzulassung

- 50% der Punkte über alle Aufgabenserien müssen zur Prüfungszulassung erreicht werden; es darf kein Aufgabenblatt ignoriert werden.

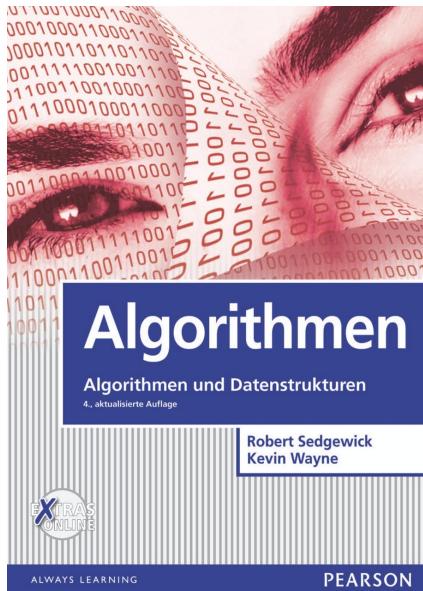
■ Klausurtermin

- Woche vom **29. Juli 2025, 10:00 Uhr**

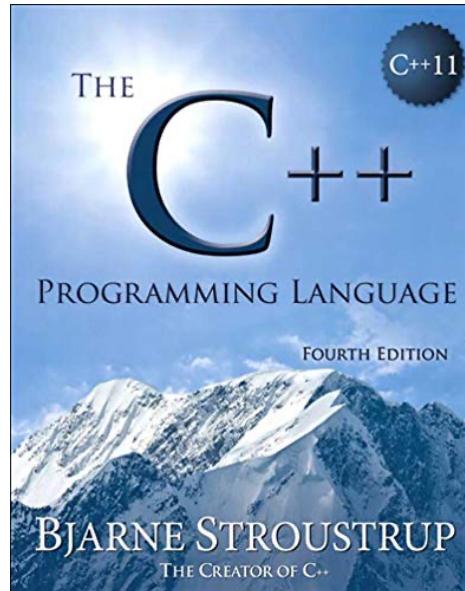
■ Klausurvorbereitung:

- Findet in der letzten Vorlesungswoche statt
- Multiple Choice + Tiefenfragen
- Kleinere Code-Fragmente können abgefragt werden (daher C++ Syntax lernen wichtig!)

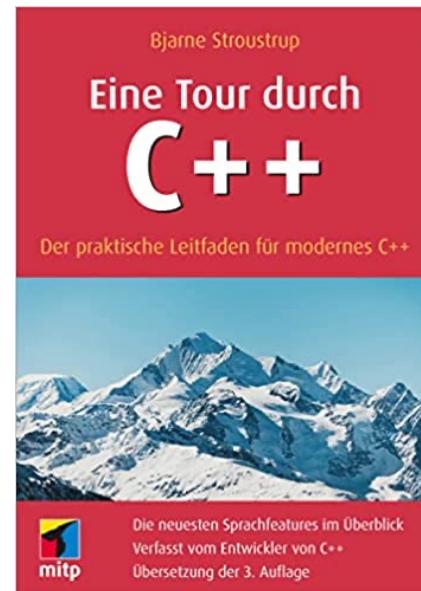
Literatur



Robert Sedgewick, Kevin Wayne. *Algorithmen*. Pearson (2014)



Bjarne Stroustrup.
The C++ Programming Language.
Addison-Wesley (2013)



Bjarne Stroustrup.
A Tour of C++ (3rd edition).
Addison-Wesley (2022)

Programmiertechnik II

Unit 1 - Algorithmen

Überblick

1. Einordnung der Lehrveranstaltung
- 2. Algorithmenbegriff**
3. Eigenschaften von Algorithmen
4. Historie von Algorithmen

Definition Algorithmen

- **Definition (Algorithmus).** Ein Algorithmus ist eine **präzise** (in einer festgelegten Sprache abgefasste) **endliche** Beschreibung eines **allgemeinen** Verfahrens unter Verwendung **ausführbarer** elementarer (Verarbeitungs-) Schritte.
- In der Informatik speziell
 - Berechnungsvorgänge statt Bearbeitungsvorgänge
 - Schwerpunkt auf Ausführbarkeit durch (abstrakte) Maschinen
- **Alternative Definition (Algorithmus).** Ein Prozessor führt einen Prozess (Arbeitsvorgang) auf Basis einer eindeutig interpretierbaren Beschreibung (dem Algorithmus) aus.



Al-Chwarizmi

Programmiertechnik II

Unit 1 - Algorithmen

Typische algorithmisierbare Prozesse

■ Kochrezepte

- Pizza backen



■ Bedienungsanleitungen

- **Kasse:** Herausgeben von Wechselgeld
- **Casino:** Sortieren von Spielkarten
- **Möbel:** Aufbau eines Regals

■ Berechnungsvorschriften

- Schriftliches Addieren
- Schriftliches Multiplizieren
- Berechnung des größten gemeinsamen Teilers

Programmiertechnik II

Unit 1 - Algorithmen

Überblick

1. Einordnung der Lehrveranstaltung
2. Algorithmenbegriff
- 3. Eigenschaften von Algorithmen**
4. Historie von Algorithmen

Eigenschaften von Algorithmen

■ Abstraktion, Parameter

- Algorithmus löst Klasse von Problemen

$ggT(p, q)$ wenn $p \geq q$

■ Endlich: Beschreibung hat endlich Länge

- Algorithmus benötigt in jedem Schritt nur endlich viel Platz

Berechne $x = p \div q$ und $r = p - x \cdot q$

■ Terminierung

- Resultat steht nach endlich vielen Schritten fest

Wenn $r = 0$

■ Determinismus

- Zu jedem Zeitpunkt besteht höchstens eine Möglichkeit der Fortsetzung

Ja

$ggT = q$

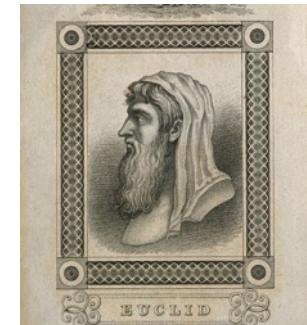
Nein

$p = q$
 $q = r$

$$1071 = 1 \cdot 1029 + 42 \quad (x = 1, r = 42)$$

$$1029 = 24 \cdot 42 + 21 \quad (x = 24, r = 21)$$

$$42 = 2 \cdot 21 + 0 \quad (x = 2, r = 0)$$



Euclid (300 BC)

Notation



HAL, bitte berechne
Fakultät von 20.

$$0! = 1$$
$$x! = x \cdot (x - 1)!$$

```
int x = 20; int y = 1;  
while (x > 1) {  
    y = y * x;  
    x = x - 1;  
}
```

Programmiertechnik II

Unit 1 - Algorithmen

Berechenbarkeit

- Kann man „alles“ programmieren (berechnen)?
 - **Antwort:** Nein!
- Es gibt **nicht-entscheidbare** Probleme:
 - **Halteproblem:** Entscheide, ob ein (beliebiger) gegebener Algorithmus terminiert.
 - Semantische Eigenschaften von Algorithmen, z.B.:
 - **Programmäquivalenzproblem:** Berechnen zwei Algorithmen dieselbe Funktion?
 - Ist ein gegebener Algorithmus korrekt, d.h. berechnet er die gegebene (gewünschte) Funktion?
- Mehr Details in der Vorlesung Theoretische Informatik

Programmiertechnik II

Unit 1 - Algorithmen

Korrektheit

- Algorithmen (Programme) sollen sich wie beabsichtigt verhalten.
- Beispiele für Programmfehler
 - **1962:** Verlust der Venus-Sonde Mariner 1 durch fehlenden Bindestrich in einem Fortran-Programm (80 Millionen Dollar)
 - **1996:** Zerstörung der Ariane 5-Rakete kurz nach dem Start durch übernommenen und nicht korrekt spezifizierten Programmcode
 - **1999:** Verlust der Mars-Sonde Climate Orbiter durch falsches Maßsystem (Yard statt Meter) bei Programmierung
- Programmverifikation



Programmiertechnik II

Unit 1 - Algorithmen

Determinismus

- Legt „Wahlfreiheit“ bei der Ausführung eines Algorithmus fest

- **Determiniertes Ergebnis**
 - Eindeutiges Ergebnis bei vorgegebenen Parameterwerten
 - Normalerweise erwünscht
 - Schwierig bei Verwendung von Zufallszahlen und Parallelität

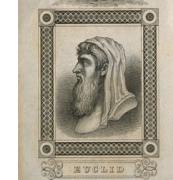
- **Deterministischer Ablauf**
 - Eindeutige Vorgabe der Folge der auszuführenden Schritte
 - Nicht dringend nötig, solange determiniertes Ergebnis
 - Nichtdeterministischer Ablauf häufiges Konzept beim Entwurf von Algorithmen

Überblick

1. Einordnung der Lehrveranstaltung
2. Algorithmenbegriff
3. Eigenschaften von Algorithmen
- 4. Historie von Algorithmen**

Historie von Algorithmen

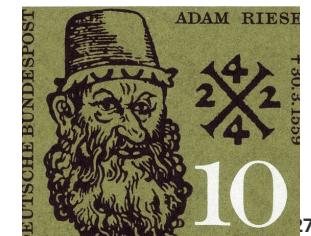
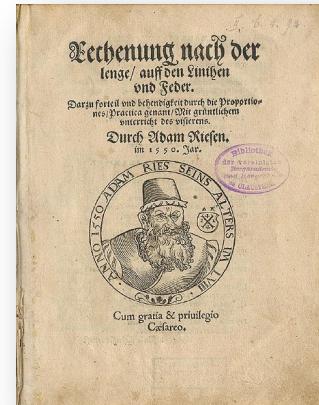
- **300 v. Chr.:** Euklids Algorithmus zur Bestimmung des größten gemeinsamen Teilers im 7. Buch der Elemente
- **800 n. Chr.:** Muhammed ibn Musa abu Djafar al Choresmi
 - Aufgabensammlung für Kaufleute und Testamentsvollstrecker (lat.: Liber Algorismi, Kunstwort aus dem Namen und griechisch „arithmos“ für Zahl)
- **1574:** Adam Rieses Rechenbuch
 - Mathematische Algorithmen auf Deutsch



Euklid (300 v.Chr.)



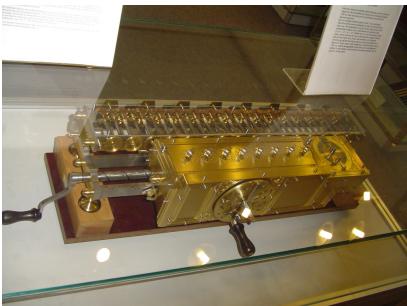
Muhammed ibn Musa (800 n. Chr.)



Adam Riese
(1492 – 1559)

Historie von Algorithmen

- **1614:** Logarithmentafeln werden algorithmisch berechnet
 - John Napier: Mirifici Logarithmorum Canonis Descriptio
 - 30 Jahre für Berechnung



John Napier's MIRIFICI LOGARITHMORUM CANONIS DESCRIPTIO 1
LIBER I (Translated and annotated by Ian Bruce.)

**The Description of the Wonderful Canon of Logarithms,
and the use of which not only in Trigonometry, but also in all
Mathematical Calculations, most fully and easily explained in the most
expeditious manner.**

*By the author and discoverer
John Napier.
Baron of Merchiston, etc. Scotland.*



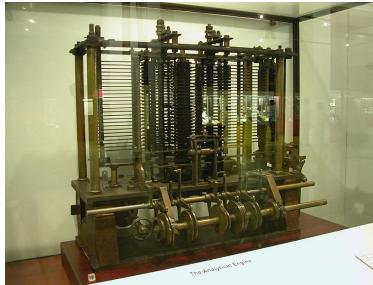
**John Napier
(1550 – 1617)**



Gottfried Leibniz
(1646 – 1716)

Historie von Algorithmen

- **1833:** Charles Babbage entwirft die *difference engine* und *analytical engine*
 - Mechanische Rechenmaschinen, zu Lebzeiten nie gebaut



An engraving of James K. Polk, the 11th President of the United States. He is shown from the chest up, wearing a dark suit, a white shirt, and a dark bow tie. His hair is receding and wavy. The style is characteristic of 19th-century political portraiture.

Charles Babbage (1791 – 1871)

- **1843:** Ada Lovelace entwirft Algorithmus zur Berechnung von Bernoulli Zahlen auf der *analytical engine*
 - „Erstes Computerprogramm“



Pro II
Un

Ada Lovelace (1815 – 1852)

Historie von Algorithmen

- **1931:** Kurt Gödels Unvollständigkeitssatz
 - Nicht jeder mathematische Satz kann aus den Axiomen eines mathematischen Teilgebietes (Arithmetik, Geometrie und Algebra) formal abgeleitet oder widerlegt werden.
 - Nicht alles kann mit algorithmisch konstruierten Beweisen belegt werden.

- **1936:** Church'sche These (Church-Turing-Theorie)
 - Die Klasse der Turing-berechenbaren Funktionen (also mit Turingmaschine) stimmt mit der Klasse der intuitiv berechenbaren / physikalisch realisierbaren Funktionen überein.
 - Vereinheitlicht die Welt der Sprachen zur Notation von Algorithmen: Alle haben gleiche Ausdrucksfähigkeit.

Nie bewiesen, nie widerlegt!



Kurt Gödel
(1906 – 1978)

2. Effective calculability. Abbreviation of treatment.

A function is said to be "effectively calculable" if its values can be found by some purely mechanical process. Although it is fairly easy to get an intuitive grasp of this idea, it is nevertheless desirable to have some more definite, mathematically expressible definition. Such a definition was first given by Gödel at Princeton in 1934 (Gödel [2], 36), following in part an unpublished suggestion of Herbrand, and has since been developed by Kleene [2]. These functions were described as "general recursive" by Gödel. We shall not be much concerned here with this particular definition. Another definition of effective calculability has been given by Church (Church [3], 356–358), who identifies it with λ-definability. The author has recently suggested a definition corresponding more closely to the intuitive idea (Turing [1], see also Post [1]). It was stated above that "a function is effectively calculable if its values can be found by some purely mechanical process". We may take this statement literally, understanding by a purely mechanical process one which could be carried out by a machine. It is possible to give a mathematical description, in a certain normal form, of the structures of these machines. The development of these ideas leads to the author's definition of a computable function, and to an identification of computability† with effective calculability. It is not difficult, though somewhat laborious, to prove that these three definitions are equivalent (Kleene [3], Turing [2]).

† We shall use the expression "computable function" to mean a function calculable by a machine, and we let "effectively calculable" refer to the intuitive idea without particular identification with any one of these definitions. We do not restrict the values taken by a computable function to be natural numbers; we may for instance have computable propositional functions.



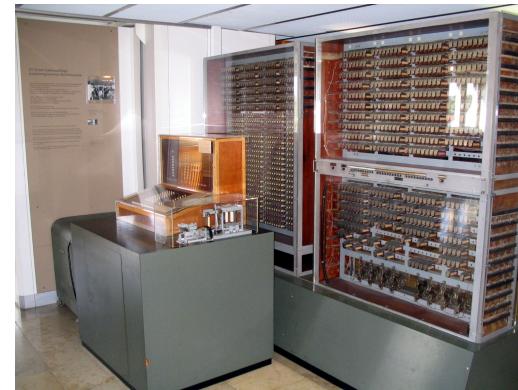
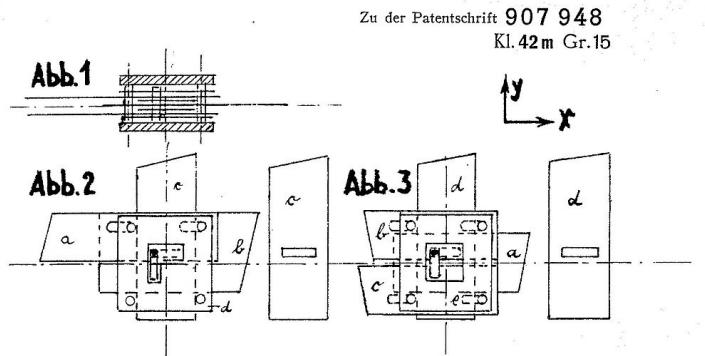
Alonzo Church
(1905 – 1995)



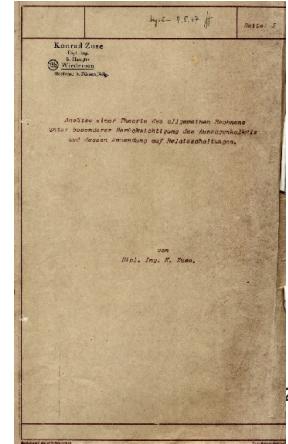
Alan Turing
(1912 – 1954)

Historie von Algorithmen

- **1937:** Konrad Zuse erfindet die Z1
 - rein mechanisch durch Steckbleche (binär)
 - Turing-komplett
 - nie nachgewiesen funktionabel
- **1941:** Konrad Zuse baut die Z3
 - Relaischaltungen (binär)
 - Erster funktionsfähiger Digitalrechner
 - Plankalkül als erste höhere Programmiersprache



Konrad Zuse
(1910 – 1995)



Viel Spaß bis zur nächsten Vorlesung!