01 - Einführung

Alexander Krause

IRB

Veranstaltungswebseite

Die Informatik

- Was ist Informatik?
- Was tut eine Programmierer:in?
- Wie sieht ihr Arbeitsplatz aus?

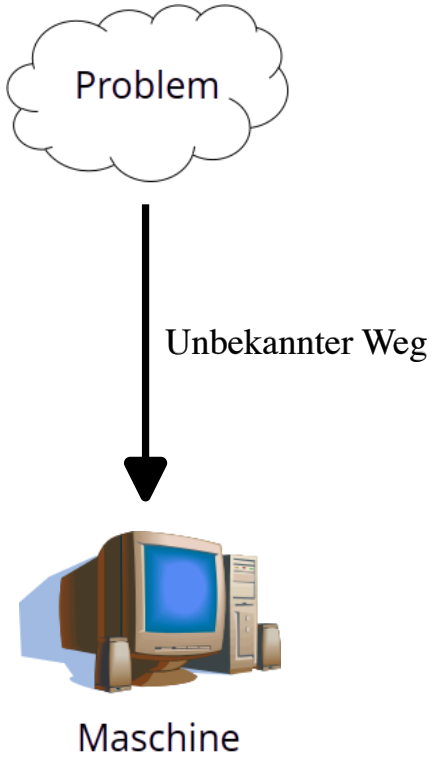
© Google Gemini AI

Die Informatik

- Was ist Informatik?
- Was tut eine Programmierer:in?
- Wie sieht ihr Arbeitsplatz aus?

© Google Gemini AI

Begriffsdefinition Informatik



- Die Informatik handelt vom **maschinellen Lösen von Problemen**.
- Die Informatik ist die Wissenschaft von der **methodischen Beherrschung algorithmisch lösbarer Probleme**.

Bereiche der Informatik

Angewandte Informatik
Anwendung auf fachfremde Bereiche

Praktische Informatik
Algorithmen(entwurf), Datenstrukturen, Problemlösung Softwaretechnik

Technische Informatik
Hardware, systemnahe Software, Kommunikation, Betriebssysteme

Theoretische Informatik
Komplexität, Entscheidbarkeit, Automaten, Form. Sprachen, Logik

Bereiche der Informatik

Angewandte Informatik
Anwendung auf fachfremde Bereiche

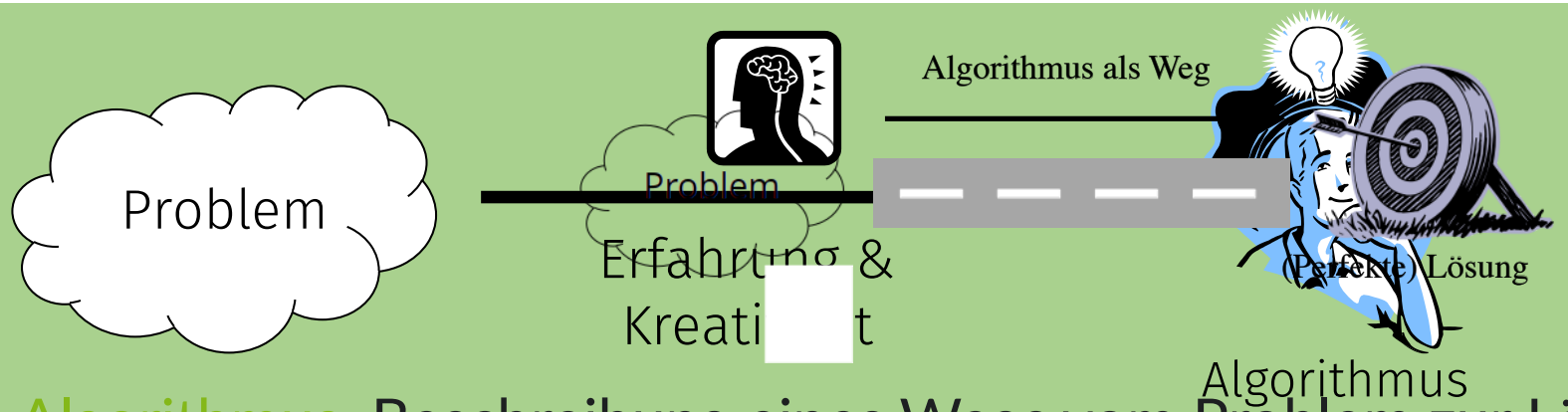
Praktische Informatik
Algorithmen(entwurf), Datenstrukturen, Problemlösung Softwaretechnik

Technische Informatik
Hardware, systemnahe Software, Kommunikation, Betriebssysteme

Theoretische Informatik
Komplexität, Entscheidbarkeit, Automaten, Form. Sprachen, Logik

Was ist Informatik? Revisited

Was ist ein Algorithmus?



- Algorithmus: Beschreibung eines Wegs vom Problem zur Lösung
- Bedingungen:
  - Der Weg muss formal so präzise definiert sein, dass er im Prinzip von einer Maschine (rein mechanisch) gegangen werden kann.
  - Problem und Lösung müssen vorher (formal) spezifiziert werden.
- Bedeutet:
  - Im Grunde Kochrezepte für Informatiker:innen
  - Komplexität von Algorithmen reicht vom „Rührei bis zur vierstöckigen Hochzeitstorte"

Maschine

Compiler

```
#include<stdio.h>
int main() {
    printf("Hello World");
    return 0;
}
```

Programm

Design, Analyse, Korrektheits- und Laufzeitbeweis, Pseudocode

Entwurf/Planung
Konkrete Umsetzung

Programmierung (EIDP)

Testen, Fehler-beseitigung Wartung/Pflege

Beispielalgorithmus

1. Weise zwei Zahlen den Variablen x und y zu

   → x = 1, y = 2

2. Addiere x und y und weise das Ergebnis der Variable z zu

   → z = x + y

3. Multipliziere z mit dem Faktor 4711. Weise das neue Ergebnis z zu

   → z = 4711 × z

4. Gib z als das Endergebnis aus

   → print(z)

Abstrakte Beschreibung ist häufig in Pseudocode

Was ist eine Programmiersprache?

„Eine Sprache, um Anweisungen an den Computer zu stellen”

```c
int x = 1;
x = x + 1;
printf("X hat den Wert: %d\n", x);
```

- Wie in der natürlichen Sprache gibt es …
  - eine Syntax,
  - eine Grammatik,
  - eine Rechtschreibung,
  - unterschiedliche Dialekte und
  - verschiedene Sprachen

Was ist eine Programmiersprache?

- *Problem:* Ein Computer versteht die (Programmier-)Sprachen nicht.

  Er kennt nur `0` und `1`.

- *Lösung:* Ein Übersetzer (*Compiler*) übersetzt von einer in die andere Sprache
- Ein Übersetzer für eine Programmiersprache …
  - überprüft Syntax und Grammatik
  - übersetzt von der Programmiersprache in die Maschinensprache: `11001100`
- Ein Programm besteht nur noch aus `0` und `1`

Exkurs zu Übersetzungsarten

- Kompilierung:
  - Geschieht **bevor** der Ausführung des Programms
  - Übersetzer überführt Programmcode in Maschinensprache
  - Ist spezifisch für einen „Computer”
  - Beispiel: C/C++
- Interpretation:
  - Ein **anderes** Programm liest den Programmcode und führt ihn aus
  - „Ein anderes Programm” heißt **Interpreter**
  - Beispiel: Shell (Kommandozeile), Python

Was ist ein Betriebssystem?

- Betriebssystem hat zwei Hauptaufgaben [1]:

1. Bereitstellung von Abstraktionen der Betriebsmittel anstelle der unschönen Hardware, mit denen Anwendungsprogrammierer arbeiten können
2. Verwaltung der Hardwareressourcen, die der Computer besitzt

Was ist ein Betriebssystem?

- Daraus folgt:

  Im Grunde ist alles eine Anwendung, die auf dem Betriebssystem aufsetzt

- Dies ist alles nicht Teil eines Betriebssystems
  - Graphische Oberfläche
  - Anwendungen
  - Kommandozeile
- Beispiel GNU/Linux
  - Linux ist der Betriebssystemkern (*Kernel*)
  - GNU ist Sammlung von essenziellen Programmen (*GNU Core Utils*) aufsetzen

Was wird behandelt?

1. Einführung in das Programmieren mit C/C++
   - Grundlagen: Identisch in den meisten (populären) Programmiersprachen
     - Variablen
     - Funktionen
     - Kontrollfluss mithilfe von konditionalen Bedingungen
   - Darstellung von Informationen
   - Speicher/Speichermanipulation (hardwarenahe Programmiersprachen)
2. „Die Werkbank”
   - Rechnerarchitektur: Was passiert in einem Prozessor?
   - Einführung in Betriebssysteme (für ein Gesamtverständnis)
   - Betrachtung des Übersetzungsprozesses und der Integration in das Betriebssystem

Was wird behandelt?

3. Weiterführende Programmierkonzepte in C++
   - Objektorientierte und generische Programmierung
   - Vorstellung von anderen Herangehensweisen an die Programmierung: Gruppierung von Informationen und Funktionen in logischen Einheiten
   - Verwendung von Datenstrukturen, die C++ bereits mitliefert (zum Beispiel Stapel, Warteschlange)
   - Behandlung von Ausnahmen
4. Best Practices
   - Grundsätzliches Vorgehen für saubere Programmierung
   - Vermeidung von häufigen Fallstricken durch vorsichtige Herangehensweise (defensives Programmieren)

Was wird *nicht* behandelt?

- Tiefergehende Algorithmen und Algorithmenanalyse

  → Vorlesung „Datenstrukturen und Algorithmen 2 (DAP2)"

- Nebenläufige Programmierung
  - Parallele Ausführung von (gleichen) Aufgaben auf mehreren Prozessoren
  - Hochinteressantes und aktuelles Thema
  - Volle Ausnutzung der vorhandenen Hardware möglich
  - Essenziell wichtig für rechenintensive Anwendungen (z.B. Simulationen)
  - Guter Ausgangspunkt → Vorlesung „Betriebssysteme"
- Programmierung einer grafischen Oberfläche (*UI = User Interface*)

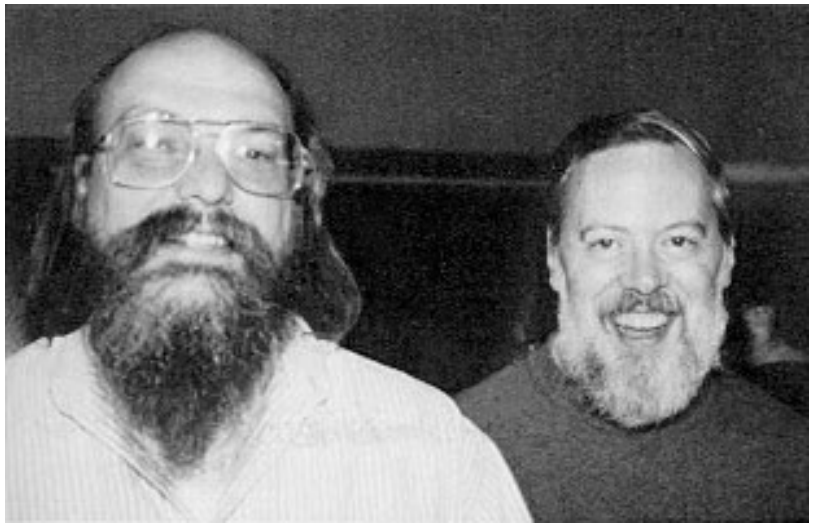
  → heutzutage häufig webbasiert in Browser-Container, wie bei Visual Studio Code

Was ist C?

- Anfang der 1970er Jahre von Dennis Ritchie erfunden
- C ist eine sogenannte Hochsprache
- Hardwarenahe Programmiersprache zur Systemprogrammierung
- Wurde für das Betriebssystem *UNIX* entwickelt
- Warum der Name C?

  → Nachfolger der Programmiersprache B

  (ebenfalls von Dennis Ritchie und Ken Thompson)



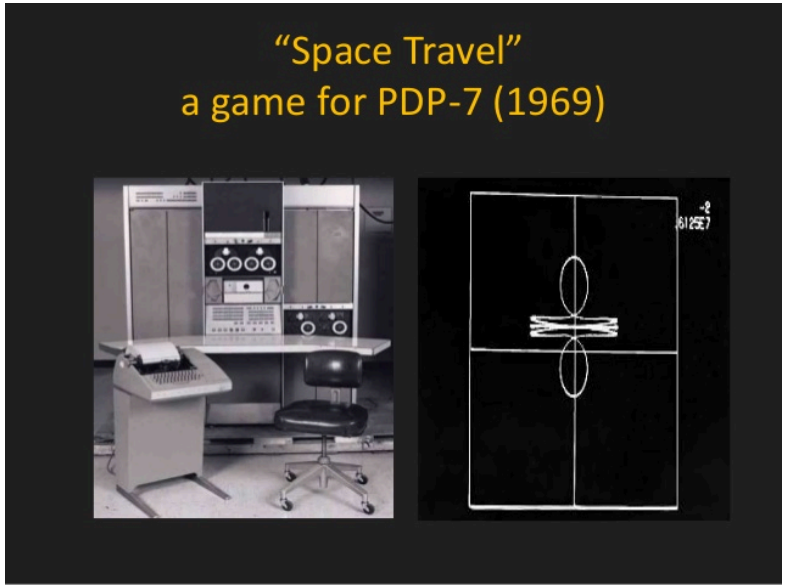
Exkurs: Entstehung von Unix

- Betriebssystem für Großrechner (*Mainframes*),
  z. B. *PDP-11*
- Entstand 1969-70 bei Bell Labs (später AT&T)
- Initiiert durch Ken Thompson
  - Hatte Spiel namens „Space Travel" für anderes Mainframe entwickelt
  - Wollte es unbedingt weiterspielen

Exkurs: Entstehung von Unix

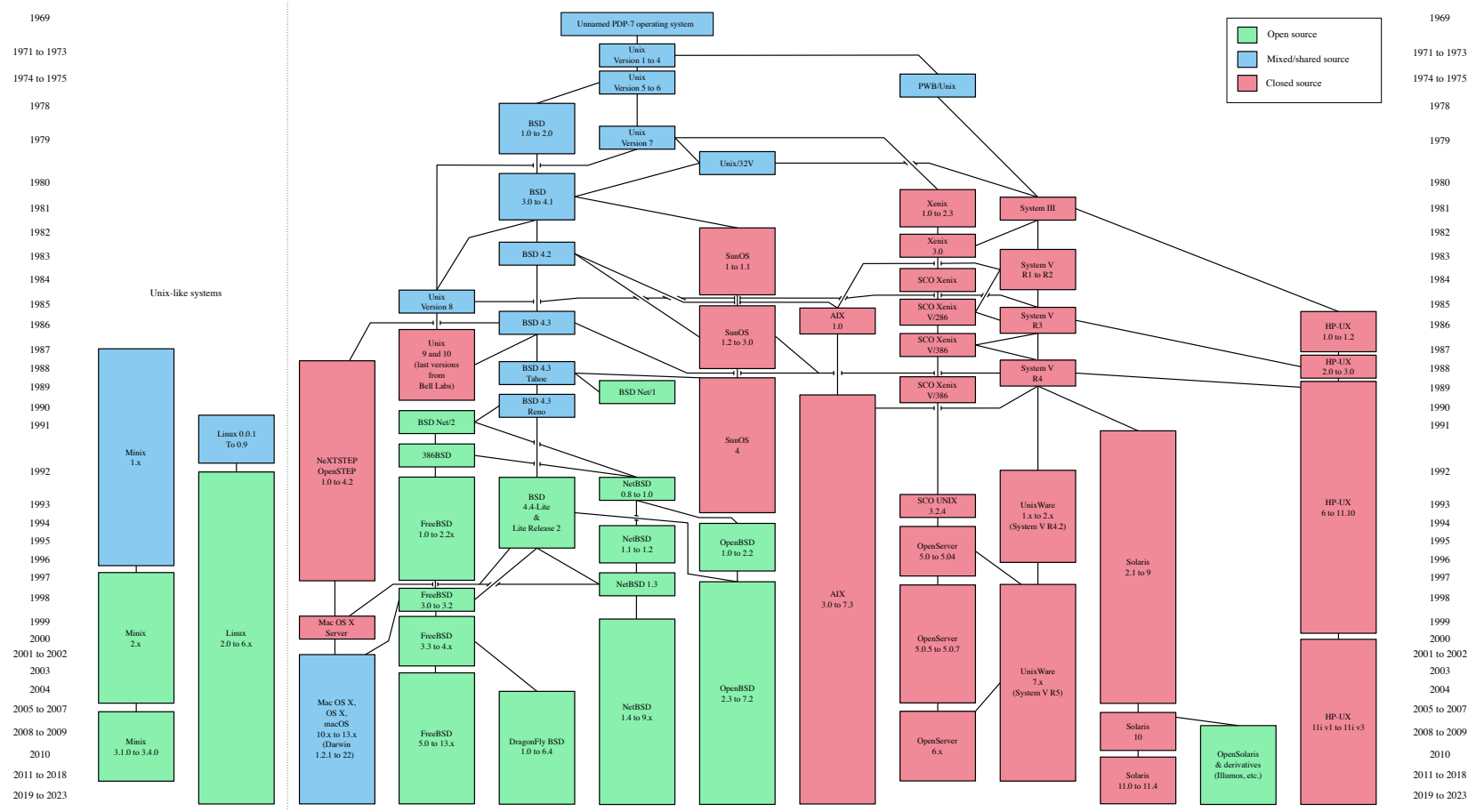
- Nutzte *DEC PDP-7*-Rechner wegen besserer graphischer Konsole
- Portierung des Spiel brachte neues Betriebssystem hervor
  - *Unics* für PDP-7 und später PDP-11
  - Erst in Maschinensprache (*Assembly*), später in *C*
- C wurde parallel zu Unix entworfen

  → Ziel: Programme angenehmer verwirklichen zu können



© *Universal Videogames List*

Die *Unix*-Familie



Warum genau C?

- Optimal für betriebssystemnahe/hardwarenahe Programmierung: Sehr nahe an Assembly → häufig direktes Analog in Assembly für C-Instruktion
- C ist im Kern eine sehr kleine Sprache mit vielen zusätzlichen Erweiterungen (*Bibliotheken*)
- Kann sogar Programme für Computer ohne Betriebssystem erzeugen (*Bare-Metal*)
- Im Allgemeinen sehr schnelle Programme, wenn gut programmiert wurde
- In der Praxis häufig kleinster gemeinsamer Nenner, falls verschiedene Programmiersprachen miteinander interagieren müssen (fast immer kompatibel zu C)

Warum genau C?

- Für Elektrotechniker:innen

  Manche (obskure) Hardware-Plattformen haben nur einen Übersetzer für C
- Außerdem: Überall zu finden.
- Entstehung von riesigen Infrastrukturen auf Basis von *C*
  - Betriebssysteme: Linux, Teile von Windows und MacOS
  - Verschlüsselung (z.B. OpenSSL)

Was ist jetzt C++?

- Von Bjarne Stroustrup 1983 erfunden

  (Büronachbar von Ritchie und Thompson)

- Geplant war C mit Erweiterung um objektorientierte Programmierung
- Aber es wurde mehr …
- C++ ist Superset von C (= beinhaltet C vollständig)

  plus zahlreiche weitere Funktionalitäten
- C++ ist komplexer und hat viele Abstraktionen im Vergleich zu C

Was ist jetzt C++?

- **Faustregel:** Man *kann* alles mit C machen, aber man muss es manuell erstellen.

  → C++ bietet bereits eine Reihe von Lösungen.

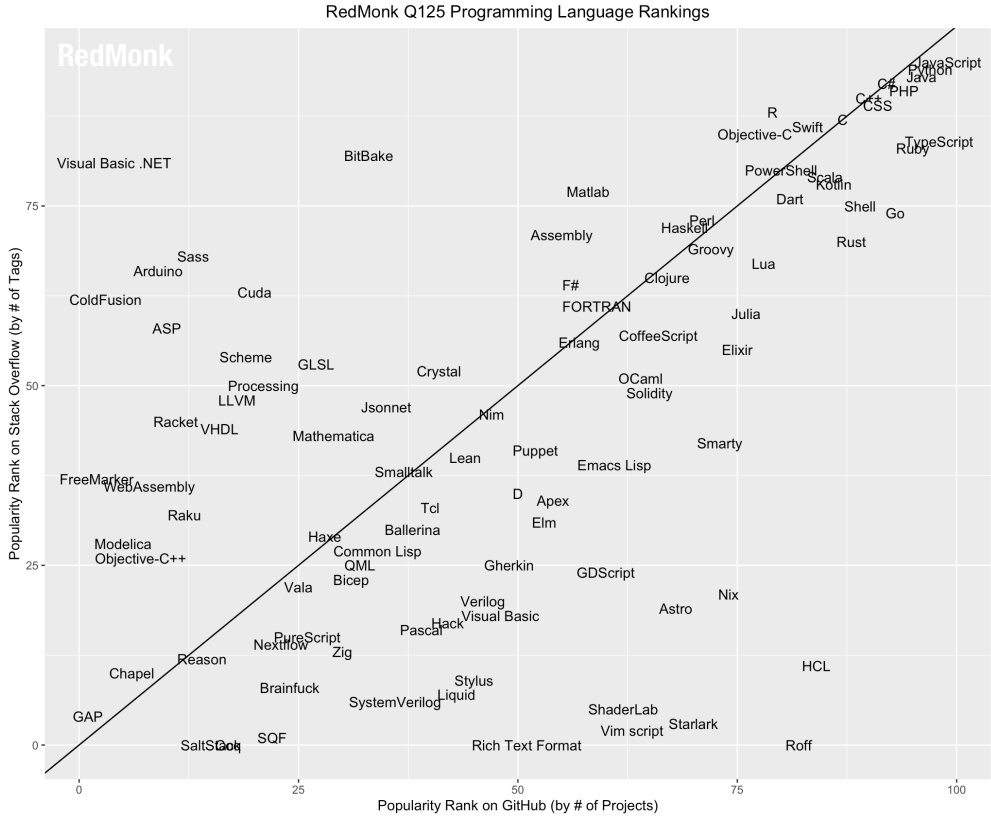
- C++ hat viele verschiedene Anwendungsbereiche
  - Software für Finanzbranche, Server/Cloud, Simulationen, …
  - Betriebssysteme und hardwarenahe Entwicklung
  - Computerspiele
- Jeder Bereich hat andere und teils gegensätzliche Anforderungen

Beliebtheit der Programmiersprachen

- C++ liegt auf Platz 7

  (mit CSS)

- Davor liegen nur interpretierte Sprachen
- C ist etwas weiter unten auf Platz 10



© *Ranking der Fa. Redmonk der verschiedenen Programmiersprachen*

C/C++-Standards

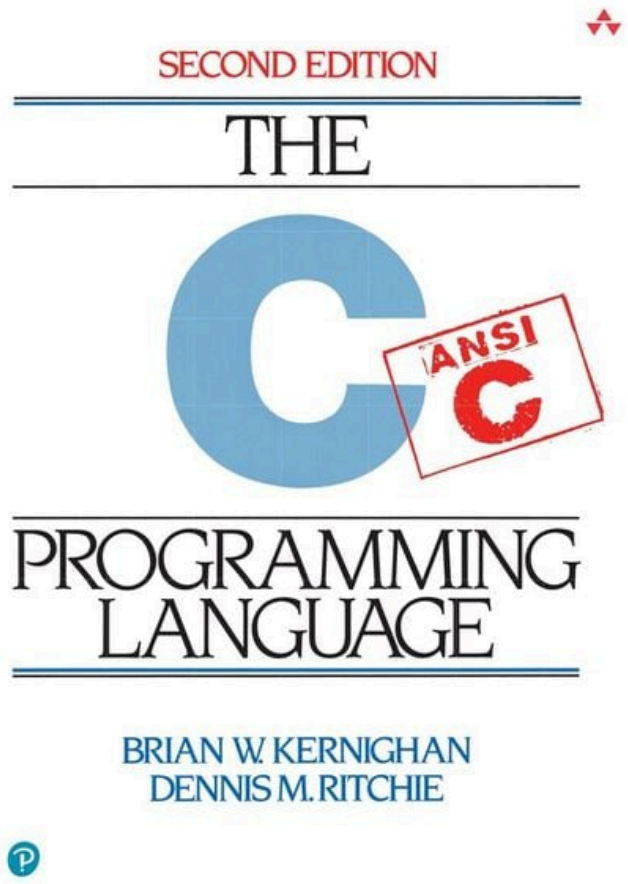
- ISO legt Sprachstandards fest
  - Festlegung des Funktionsumfangs
  - Beschreibung des Verhaltens, inkl. undefiniertem Verhalten (*Undefined Behavior*)
  - Z. B. ”Wie groß sind die darstellbaren Zahlen?”
- C++-Standards
  - C++98
  - C++03
  - C++11 (C++0x)

    → gilt als Beginn von modernem C++
  - Und seitdem alle 3 Jahre neue Version (C++14, 17, 20, 23, 26)
- C-Standards
  - C89 (ANSI C)
  - C99
  - C11
  - … und einige sehr kleine Aktualisierungen (C17, C23)

Vorlesungsbegleitende Materialien

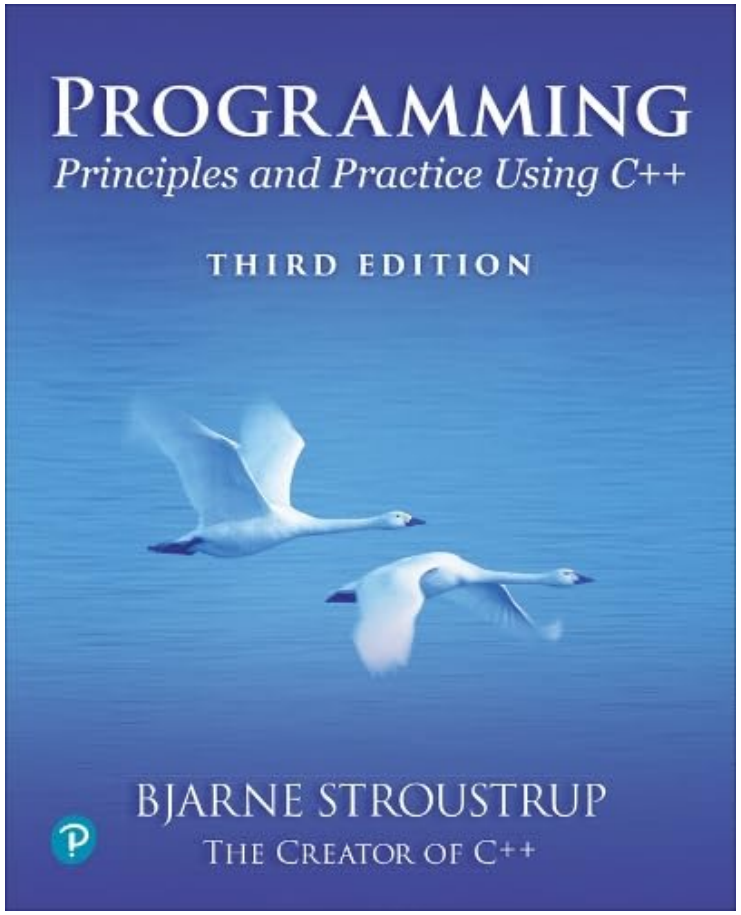
- Es gibt zahlreiche Bücher, Tutorials, Webseiten und Blog-Artikel
- Prinzipiell sind für das rudimentäre Wissen alle geeignet
- Wenig gute Literatur auf Deutsch, die Sprache der Wahl ist Englisch

Der Klassiker: *C Programming Language Second Edition*



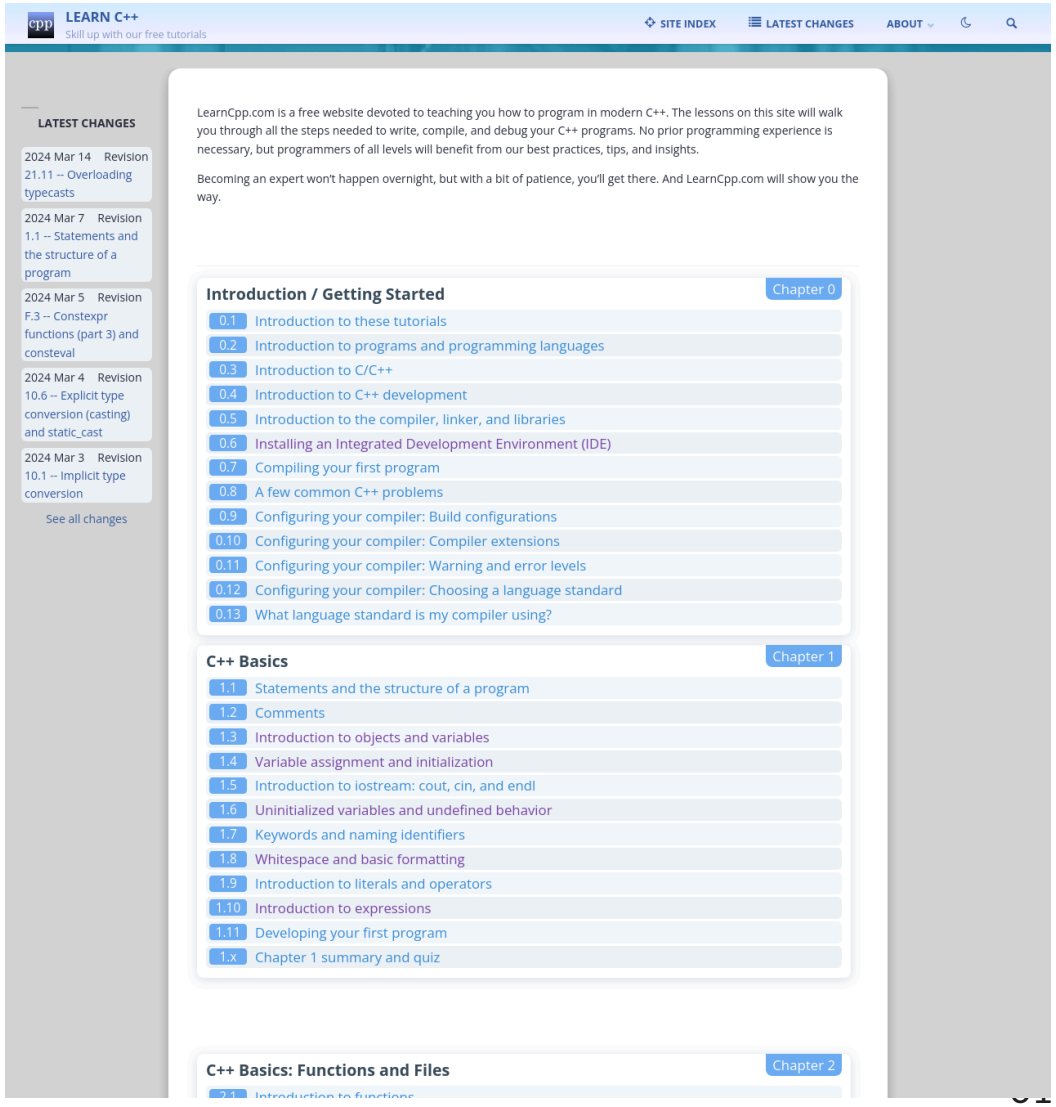
- **Das** Buch über *C* von Kernighan & Ritchie [2]
- 1989 erschienen
- Für den ersten Teil der Veranstaltung größtenteils benutzbar
- Auf Englisch in der Universitätsbibliothek verfügbar

Der Klassiker++: *Programming – Principles and Practice Using C++*



- Bjarne Stroustrup [3]
- Starker Fokus auf C++
- Ist stellenweise gewöhnungsbedürftig, aber definitiv geeignet
- Zweite Auflage ist auf Englisch in der Universitätsbibliothek verfügbar

learncpp.com



- Webseite mit Tutorials, die kapitelweise C++ erläutern
- Etwas näher an C++ orientiert, insgesamt sehr anschauliche Erläuterungen
- Viele gute Tipps zu den einzelnen Themen miteingestreut

cppreference.com

- **Sehr umfangreiche** Webseite zu C und C++
- Weniger zum Lernen geeignet
- Eher ein Nachschlagwerk

Zusammenfassung

- Plan für das Semester
- Streifzug durch
  - die UNIX-Geschichte
  - die Geschichte von C u

Das war’s mit der Einführung.
Jetzt wird programmiert! 🥳

Referenzen

[1] A. S. Tanenbaum und H. Bos, *Modern Operating Systems*, 4. Aufl. USA: Prentice Hall Press, 2014.

[2] B. W. Kernighan und D. M. Ritchie, *The C programming language*. USA: Prentice Hall Press, 1989.

[3] B. Stroustrup, *Programming: Principles and Practice Using C++*, 2. Aufl. Boston, MA: Addison-Wesley Educational, 2014.