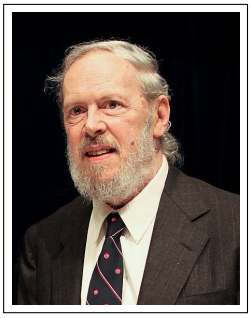
# Tutorium Programmierkurs

**Sommer 2022** **Übung 2**

---

Dennis MacAlistair Ritchie war ein amerikanischer Informatiker. Er entwickelte zusammen mit Ken Thompson und anderen die erste Version des Unix-Betriebssystems und schrieb die ersten sechs Ausgaben des Unix Programmer's Manual, ehe das Handbuch in „Unix Time-Sharing System“ umbenannt wurde.
Zusammen mit Thompson und Brian W. Kernighan entwickelte er die Programmiersprache C. Kernighan und Ritchie schrieben gemeinsam das Buch The C Programming Language, das oft mit dem Kürzel „K&R“zitiert wird. Bekannt und berühmt sind auch seine Initialen „dmr“, die Ritchie als Username für den System-Login verwendete. Für seine Arbeiten, insbesondere die Entwicklung des bis heute maßgeblichen Unix-Betriebssystems und der Sprache C, erhielt er einige der höchsten Auszeichnungen der Informatik wie etwa den Turing-Award.



*Dennis Ritchie*

---

## Aufgabe 2.1: Collatz-Vermutung

Schreiben Sie eine Funktion `void collatz(int number)`, die folgendes tut:

- Gib `number` auf dem Bildschirm aus.
- Falls `number` gerade ist, teile die Zahl durch 2.
- Andernfalls multipliziere die Zahl mit 3 und addiere 1.
- Wiederhole diese Schritte, bis einer der folgenden Zahlenwerte erreicht wird: $1, 0, -1, -5$ oder $-17$.
- Gib abschließend dieses Ergebnis aus.

**Hinweis**:
Um herauszufinden, ob eine Zahl $x$ gerade ist, testen Sie, ob $(x \mod 2) = 0$. Hierfür gibt es in C++ den Operator `x % y`, der den Rest der Ganzzahl-Division von $x$ durch $y$ berechnet:

```
1 int x = 23 / 5; // 4 (runden nach 0)
2 int y = 23 % 5; // 3 (vorzeichenbehafteter Divisionsrest)
```

Rufen Sie die obige Funktion aus der `main`-Funktion auf, wobei Sie den ersten Wert für `number` von der Tastatur einlesen. Auf diese Weise können Sie die entstehenden Zahlenfolgen für verschiedene Startwerte untersuchen. Warum kann der Wert 0 nur auf eine Weise erreicht werden? Und was haben alle Zahlen gemeinsam, die zum Wert 1 führen? Informieren Sie sich unter `https://en.wikipedia.org/wiki/Collatz_conjecture` über die mathematische Vermutung hinter diesen Zahlenfolgen.

## Aufgabe 2.2: Fibonacci-Folge

Jedes Element der Fibonacci-Folge wird durch Addition der beiden vorherigen Folgen-Elemente gebildet, wobei die ersten beiden Elemente durch 0 und 1 gegeben sind:

$$\begin{aligned} f_1 &= 0, \\ f_2 &= 1, \\ f_n &= f_{n-2} + f_{n-1}. \end{aligned}$$

Damit ergibt sich als Beginn der Folge:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$$

(a) Implementieren Sie die Berechnung der Folgen-Elemente in einem C++-Programm.

Schreiben sie eine Funktion `int fibonacci(int number)`, welche $f_N$ berechnet. Lesen Sie $N$ von der Kommandozeile ein und geben sie $f_N$ aus. Erweitern Sie Ihr Programm so, dass es nacheinander $f_n$ für alle Werte von 0 bis $N$ auf der Kommandozeile ausgibt. Probieren Sie Ihr Programm für verschiedene Werte von $N$ aus. Was passiert, wenn Sie $N$ groß werden lassen? Haben Sie eine Erklärung hierfür?

Hinweis: Sie können die Geschwindigkeit Ihres Programms verbessern, indem Sie beim Kompilieren die Option `-O3` angeben. Dadurch analysiert der Compiler Ihr Programm und erzeugt ein optimiertes Programm, das normalerweise deutlich schneller ist.

(b) Im folgenden geht es darum, das Programm aus dem ersten Teil zu verbessern.

*Anmerkung:* Es kann natürlich sein, dass Sie den vorherigen Aufgabenteil schon so implementiert haben, dass hier gar keine Probleme auftreten. In diesem Fall sind Sie bereits fertig.

Sorgen Sie dafür, dass Ihr Programm auch für grössere $N$ korrekt funktioniert. Schauen Sie sich hierzu die Folien zu Variablentypen an. Die Laufzeit des Programms steigt für ungefähr $N = 40$ sehr stark an. Woran liegt das? Schreiben Sie eine alternative Version des Programms, die dieses Problem umgeht und die Folgenglieder bis mindestens $N = 90$ ohne nennenswerte Verzögerung ausgeben kann.

## Aufgabe 2.3: Mitternachtsformel

Schreiben Sie ein Programm, das von der Kommandozeile die Koeffizienten einer quadratischen Gleichung der Form $ax^2 + bx + c = 0$ abfragt und mit der Mitternachtsformel die beiden Nullstellen ausrechnet und ausgibt. Falls es keine eindeutige Lösung gibt

($a = b = 0$) oder die Lösung komplex ist, soll das Programm das abfangen und eine entsprechende Meldung ausgeben.

Hinweise:

- Um eine Kommazahl von der Kommandozeile einzulesen, verwenden Sie folgenden Codeschnipsel:

```
1 double v;
2 std::cout << "a = " << std::flush;
3 std::cin >> v;
```

- Um die Wurzel aus einer Zahl zu ziehen, gibt es die Funktion `sqrt`:

```
1 double root = std::sqrt(2.0);
```

  Bevor Sie diese Funktion verwenden können, müssen Sie am Anfang des Programms die Mathematik-Bibliothek von C++ mit der Zeile `#include <cmath>` einbinden.

## Aufgabe 2.4: Einfache Primzahlsuche

Schreiben Sie ein Programm, das alle Primzahlen zwischen 1 und 100 ausgibt.

(a) Prüfen Sie für jede Zahl zwischen 1 und 100, ob sie noch andere Teiler als 1 und sich selbst hat. Sie müssen als Teiler nur Zahlen bis $\sqrt{n}$ prüfen. Die Verwendung von `for` oder `while` bietet sich hier an

(b) Ob eine Zahl Teiler ist, können Sie mit dem %-Operator prüfen (s. Aufgabe 2.1)

(c) Zählen Sie die Anzahl von Ganzzahldivisionen, die Sie benötigen, um alle Primzahlen zwischen 1 und 100 zu bestimmen. Im Verlauf der Vorlesung werden wir dies mit anderen Algorithmen vergleichen.