Tipps zu C++ Compilern

- Kompilieren im Terminal unter macOS:

```
clang++ code.cpp
```

- Kompilieren im Terminal unter Linux:

```
g++ code.cpp
```

- GCC unter Windows nachinstallieren (ohne IDE): Cygwin
- Dann kann man auch unter Windows im Cygwin-Terminal mit

```
g++ code.cpp
```

kompilieren

Gruppierung von Befehlen

- So lassen sich mehrere Befehle zu einem zusammenfassen

```
{
        Befehl 1;
        Befehl 2;
        ...
}
```

- Dadurch wird aber auch die Sichtbarkeit/Gültigkeit (scope) von Variablen eingeschränkt $\rightarrow$ `scope.cpp`

Division: int vs double

- Was passiert bei der Division von zwei `int` Variablen? → `div-int.cpp`
- Wie erzwingt man Gleitkommadivision?
  - Mindestens ein Operand muss eine Gleitkommazahl sein → `div-double-1.cpp`
  - oder durch explizite Typumwandlung (type casting) → `div-double-2.cpp`

Schleifen (loops)

- Sinn und Zweck: Eine Anweisung soll mehrfach ausgeführt werden
- Beispiele
  - Summationen: $\left(\sum_{i=1}^{n} i\right)$, $\left(\sum_{i=1}^{n} \frac{1}{i^2}\right)$ usw.
  - Setzen und Auslesen von Feldwerten
  - Wiederhole die Anweisung X so lange, bis die Bedingung Y wahr wird
  - Plotten: Berechne den Wert von $f(x)$ an den Stützstellen x_i
  - Implementierung von Algorithmen
- Verschachtelte Schleifen (z.B. für Matrizen und Tensoren) sind ebenfalls möglich
- Mit Schleifen lassen sich sowohl sehr einfache als auch höchst nichttriviale Probleme lösen

- Wichtige Konzepte in der Programmierung von Schleifen
  - Anfangsanweisung (AA): Wird nur einmalig ganz am Anfang ausgeführt.
  - Fortsetzungsbedingung (FB): Ist diese Bedingung wahr, so kann die nächste Iteration stattfinden, sonst wird die Schleife abgebrochen.
  - Iterationsanweisung (IA): Ausführung bei jeder Iteration
  - Code: Die eigentliche Aufgabe der Schleife
- Drei Grundtypen der Schleifen in C++
  - `for`-Schleifen

    ```
    for(AA; FB; IA) { Code };
    ```

  - `while`-Schleifen

    ```
    while (FB) { Code };
    ```

  - `do`-Schleifen

    ```
    do { Code } while(FB);
    ```

- Diese drei Formen sind äquivalent, jedoch gibt es Codestellen, für die sich eine Form viel besser als die beiden anderen eignet

- `for`: Die Anzahl der Wiederholungen ist bereits bekannt
- `while`: Die Anzahl der Wiederholungen ergibt sich im Laufe des Programms
- `do...while`: Wie `while`, aber der Code wird mindestens einmal ausgeführt
- Implementierungsbeispiele → `quad-werte.cpp`
- Weitere explizite Beispiele
  - Ausgabe der Werte des Felds `myPrimes` → `primes.cpp`

```
int myPrimes[6] = {2, 3, 5, 7, 11, 13};
```

  - Berechnung eines Näherungswerts der Riemannschen Zeta-Funktion an der Stelle 2: $\zeta_2 = \sum_{i=1}^{\infty} \frac{1}{i^2} = \pi^2/6 \approx 1,6449341$ → `zeta2.cpp`
  - Länge eines n-Dimensionalen euklidischen Vektors $\vec{p}$:

$$|\vec{p}| = \sqrt{\sum_{i=1}^{n} (p^i)^2}$$

→ `vec-len.cpp`

if-Anweisungen

- `if`-Anweisungen gehören zu den wichtigsten Kontrollstrukturen eines Programms
- Ausschließlichkeit erzwingen durch `if-else` → `frage-1.cpp`
- Verschachtelte `if`-Anweisungen: mächtig aber manchmal schwer zu überblicken → `frage-2.cpp`
- Validierung der Benutzereingaben möglich → `sqrt.cpp`

- Bei `if`-Anweisungen immer an den Programmablauf denken: was passiert, wenn die Bedingung wahr/falsch ist?
- Häufiger Denkfehler: Diese Bedingung ist doch eigentlich immer *wahr*, sie kann nie falsch sein
- Daher: Immer Checks einbauen, selbst für unmögliche bzw. extrem unwahrscheinliche Ereignisse

Nachbesprechung Übungsaufgaben

- Gleitkommazahlen können mit `float` und `double` nur näherungsweise dargestellt werden
- Beliebig große (oder kleine) Zahlen sind auf diese Weise nicht möglich
  → `a3-exponential.cc`
- Immer im Hinterkopf behalten, sonst drohen fehlerhafte Ergebnisse!

Vorbesprechung Übungsaufgaben

- Aufgabe 4: verschachtelte Schleifen benutzen $\rightarrow$ `matrix.cc`
- Aufgabe 5
  - Darstellung einer 3-stelligen Zahl im Dezimalsystem

$$z = c_2 \cdot 10^2 + c_1 \cdot 10^1 + c_0 \cdot 10^0, \quad 0 \leq c_i \leq 9$$

z.B.

$$147 = 1 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0 \Rightarrow 147 = 147_{\text{Basis } 10}$$

  - Gleiches Vorgehen im Binärsystem

$$z = c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0, \quad 0 \leq c_i \leq 1$$

$$6 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \Rightarrow 6 = 110_{\text{Basis } 2}$$

  - Analog im Dreiersystem