Programmiermethoden in der Mathematik
Klassen in C++

Implementierung einer Klasse am Beispiel der Klasse `Bruch`:

- **Deklaration** erfolgt in einer Header-Datei `bruch.h` (oder `.hh`): vgl. VL. Bemerkungen:
  - Attribute in Klassen sind standardmäßig `private`, also privat und vor dem Zugriff von außen geschützt. Es ist dennoch sinnvoll, dieses Schlüsselwort explizit aufzuführen.
  - Konstruktoren haben keinen Rückgabewert, auch nicht `void`!!! Sie sind also ein Sonderfall der bisher bekannten Funktionen.
  - Ist kein Konstruktor deklariert, so existiert immer ein Default-Konstruktor ohne Parameter. Sobald irgendein Konstruktor in der Klassendeklaration enthalten ist, ist dies nicht der Fall.
  - Eine Klasse in C++ kann als eine Erweiterung einer Struktur um Zugriffsbeschränkungen und Methoden verstanden werden.
- **Instanzierung und Benutzung der Methoden** im Programm:
  - Instanzierung eines Objektes der Klasse `Bruch`:
    - Default-Konstruktor: `Bruch b;`
    - Konstruktor mit einem Parameter: `Bruch b = 1;` oder `Bruch b(1);`
    - Konstruktor mit zwei Parametern: `Bruch b(1,2);`
  - Ausgabemethode aufrufen:

    ```
    b.print();                  // Methode print fuer Objekt b aufrufen
    ```

    - Die Schreibweise `b.print` entspricht der für Elemente einer Struktur.
    - Die Methode `print` bekommt keinen Parameter. Im OO-Sinne *sendet* die Anweisung `b.print()` *die Nachricht* "`print`" an das Objekt `b` der Klasse `Bruch`.

  Bemerkungen:
  - Ein direkter Zugriff auf die Attribute `zaehler` und `nenner` in der Form `nenner=1` etc. ist *außerhalb der Methoden der Klasse* nicht möglich, da diese privat (`private`) und damit geschützt sind.
  - Objekte können als Ganzes zugewiesen werden:

    ```
    Bruch b1(1,2),b2;
    b2=b1;
    ```

  - Es können Felder von Objekten einer Klasse angelegt werden:

    ```
    Bruch b[10];
    ```

  - Man kann einen Zeiger auf ein Objekt definieren:

    ```
    Bruch b,*bp;
    bp = &b;                    // bp zeigt auf b
    bp->print();                // entspricht (*bp).print(), also b.print();
    ```

- **Implementierung** der Klasse und ihrer Methoden: in einer separaten Datei `bruch.cc`, vgl. VL. Bemerkungen:
  - Vor jedem Methodennamen steht der **Bereichsoperator** `::` – er ordnet dem nachfolgenden Namen als Gültigkeitsbereich die davorstehende Klasse zu. `Bruch::print()` bedeutet also: Die Methode `print` hat in dieser Form nur Gültigkeit, wenn sie auf ein Objekt der Klasse `Bruch` angewandt wird.
  - Die Konstruktoren können noch vor Beginn der eigentlichen Anweisungen (in { }) eine Initialisierung der Attribute (hier `zaehler`, `nenner`) vornehmen: In der sog. **Initialisierungsliste** nach dem Doppelpunkt. Daher ist z.B. beim ersten Konstruktor der eigentliche Anweisungsteil leer.
  - Nur die Konstruktoren und Methoden der Klasse können *direkt* auf die *privaten* Attribute (hier `zaehler`, `nenner`) zugreifen.
  - Ein Destruktor kann, muss aber nicht implementiert werden:
    - *Lokale Objekte* werden automatisch zerstört und ihr Speicherplatz freigegeben, wenn ihr Gültigkeitsbereich (also die Funktion, in der sie instanziert wurden) verlassen wird.
    - *Globale Objekte* werden beim Beenden der Funktion `main` zerstört. Da darüber hinaus Konstruktoren andere Funktionen aufrufen können, kann also bei der Verwendung von globalen Objekten vor (bzw. nach) den Anweisungen in der Funktion `main` schon (bzw. noch) einiges passieren.