Notizen AIK ProgTech 3: Objekte mit dynamischen Daten

Klaus Kusche

Enthalten Objekte *als Membervariablen Pointer*, die auf mit **new** angelegte Arrays (oder Objekte) zeigen, sind einige Hinweise zu beachten:

- Solche Klassen brauchen normalerweise einen **Destruktor**, der für jedes solches Pointer-Member ein **delete** aufruft.
- Weiters brauchen solche Klassen normalerweise einen expliziten **operator=** und einen selbstgeschriebenen **Copy-Konstruktor**.

  Der automatisch erzeugte **operator=** und der automatisch erzeugte Copy-Konstruktor würden nämlich eine *bitweise* Kopie bzw. Zuweisung machen, auch von den Pointer-Membern, und *die dynamischen Daten nicht kopieren*.

  Beide Objekte würden sich dann *dieselben dynamischen Daten teilen*: Ändert einer die Daten, sieht auch der andere die geänderten Daten, und gibt einer (z.B. im Destruktor) die Daten frei, zeigen die Pointer-Member des anderen ins Leere.
- Der Copy-Konstruktor legt zuerst die dynamischen Arrays oder Objekte mit **new** an und kopiert dann die Werte aus den dynamischen Daten des Originals hinein.
- Der **operator=** ist komplexer:
  - Als erstes sollte er ein **if** enthalten, das bei *Selbst-Zuweisung* sofort ein **return** macht.
  - Wenn die vorhandenen dynamischen Daten schon die richtige Größe haben, können die Werte gleich kopiert werden, ohne neue Daten anzulegen. Sonst (z.B. bei neuer Array-Größe) müssen zuerst die *alten Daten freigegeben* und neue in der richtigen Größe *angelegt* werden, dann erst wird *kopiert*.
- In allen Konstruktoren darf man das **new** *direkt in die Init-Liste* schreiben:

  ```
  myArrClass(int size) : mSize(size), mArray(new int[size]) { …
  ```

Wiederholung Copy-Konstruktor:

- Copy-Konstruktoren müssen eine **const**-*Referenz* auf ein Objekt der eigenen Klasse als einzigen Parameter haben.

  Würde man den Parameter “by Value” übergeben, wäre eine Endlos-Rekursion die Folge, denn zur by-Value-Übergabe wird der Copy-Konstruktor aufgerufen…
- Bei einem Copy-Konstruktor in einer abgeleiteten Klasse muss in der Init-Liste als Erstes der Copy-Konstruktor der Basisklasse aufgerufen werden (selbst dann, wenn die Basisklasse nur den automatisch erzeugten Copy-Konstruktor hat und keinen selbstgeschriebenen)!!!

  Macht man das nicht, ruft C++ nämlich den *Standard-Konstruktor* der Basisklasse auf, d.h. die Member der Basisklasse werden nicht kopiert, sondern bleiben uninitialisiert!