Programmieren C++: Operatoren, Array-Member, Templates, ...

Klaus Kusche

Summe mit allen Summanden

Gesucht ist eine Klasse, die im Wesentlichen eine *Zahl* enthält und
einen *Additionsoperator* bietet. Dabei soll die Klasse intern *alle Werte mitspeichern*,
die im Laufe der Zeit zur Zahl *dazugezählt* worden sind.

Die Klasse hat also drei (private!) *Member*:

- Die *Zahl* selbst (ein **`double`**).
- Einen Pointer auf ein *dynamisch angelegtes Array* mit *allen Werten*, die seit dem Anlegen des Objektes zur Zahl dazugezählt worden sind (incl. dem *Initialwert*).
- Die *aktuelle Größe des dynamischen Arrays* (das Array soll immer genau so groß sein, wie es gerade gebraucht wird, d.h. so viele Elemente haben, wie es Werte gibt).

Weiters hat die Klasse folgende *Methoden*:

- Einen *Standard-Konstruktor* (setzt die Zahl auf 0 und den Array-Pointer auf **`NULL`**).
- Einen *Konstruktor* mit einer Zahl als Parameter
  (speichert diese Zahl in der Zahl selbst und im einzigen Element des Arrays).
- Einen *Copy-Konstruktor* (kopiert Zahl und Array).
- Einen *Destruktor* (was muss der wohl tun?).
- Eine *Get-Funktion* zum Auslesen der Zahl.
- Einen **`+`**-*Operator*, dessen *rechter Operand eine Zahl* ist: Er hängt die Zahl hinten an das Array an (das Array des Ergebnisses ist daher um 1 länger als das Array des linken Operanden) und zählt die Zahl zum momentanen Wert dazu.
- Einen **`+`**-*Operator*, dessen *rechter Operand ein Objekt* unserer Zahl-Klasse ist: Das Ergebnis hat als Zahl die Summe der Zahlen der beiden Operanden, und im Array des Ergebnisses stehen nacheinander die Werte aus den Arrays beider Operanden.
- Einen *Zuweisungs-Operator* mit der üblichen Funktionalität
  (was musst du alles tun bzw. beachten?).
- Eine **`friend`**-Funktion **`<<`** zum *Ausgeben* der Zahl und aller Werte,
  aus denen sie entstanden ist (in **`[ ]`** hinter der Zahl).

Schreib dazu ein *kurzes Hauptprogramm*, das die Methoden testet.

Zusatzaufgabe:

- Schaffst du es auch, die Klasse als *Template für beliebige Zahlentypen*
  (**`int`**, **`long`**, **`double`**, ...) zu definieren?

  Hinweis: **`friend`**-Funktionen in Templates sind tückisch, *siehe Studenten-Skript*!

  Hinweis: Wenn du eine *Template-Spezialisierung* (siehe Studenten-Skript, frag mich) für den *Standard-Konstruktor* mit Init-Wert **`""`** statt **`0`** schreibst, funktioniert das Template sogar mit der Klasse **`string`** (die Strings werden zusammengehängt).