

# Programmieren 1 Übung: Objekte mit Pointern, Operatoren: Klasse für Zähler-Array

## *Klaus Kusche*

Gesucht ist eine Klasse **Counter**, deren Objekte intern ein Array von Zählern (**int**-Werten) enthalten. Die einzelnen Zähler werden mit ihrem Index angesprochen, d.h. sie haben die Nummer 0 bis (Zähleranzahl - 1). Man könnte damit z.B. zählen, wie oft jede einzelne Zahl beim Würfeln oder beim Lotto-Spielen auftritt, oder man könnte damit (da **char**'s ja auch durch **int**-Werte dargestellt werden) zählen, wie oft jeder Buchstabe in einem Text auftritt.

Das Array soll dynamisch angelegt (und wieder freigegeben!) werden, die Objekte enthalten einen Pointer darauf als Member-Variable. Die Größe der Arrays wird im Konstruktor als Parameter angegeben und auch in einer Member-Variablen gespeichert.

### Hinweise:

- Wenn du Array-Größe und Zähler-Index **unsigned int** machst, sparst du dir ein paar Sicherheitsabfragen auf negative Werte.
- Addition, Zuweisung und Vergleich müssen nur dann ein Ergebnis liefern, wenn die Arrays in beiden Operanden gleichviele Elemente haben bzw. wenn (bei der Addition einer Zahl) der zu erhöhende Zähler innerhalb des Arrays liegt. Wenn nicht, soll das Programm mit einer Fehlermeldung enden (siehe Zusatzaufgabe).
- Achte auf korrekte **const**, verwende in den Konstruktoren Initialisierungslisten!
- Du sollst nur jene Methoden inline implementieren, die keine Schleife enthalten.
- Member-Variablen dürfen wie üblich nicht von außen sichtbar sein, deine <<-Funktion soll aber schon direkt auf die Member zugreifen können.

Die Klasse soll folgende öffentliche Methoden bieten:

- Keinen Standard-Konstruktor.
- Den oben genannten Konstruktor (mit als Parameter angegebener Array-Größe). Er soll alle Zähler im frisch angelegten Array auf 0 setzen.
- Einen Copy-Konstruktor mit der üblichen Funktionalität (neues Array dynamisch anlegen und Inhalt kopieren).
- Einen Destruktor (was macht der?).
- Eine Get-Methode für die Array-Größe.
- Einen Index-Operator zum Auslesen des aktuellen Standes des **i**-ten Zählers: Er hat die Zähler-Nummer als Argument und liefert einen **int** als Ergebnis. Der Index-Operator dient nur zum Lesen (und nicht zum Ändern) eines Zählers. Er soll den Zählerwert daher "by Value" zurückliefern (anstatt wie sonst beim Index-Operator üblich eine Referenz auf das gewählte Element zu liefern).

Der Index-Operator soll Returnwert 0 (keine Fehlermeldung!) liefern, wenn der Zählerindex außerhalb der Arraygrenzen liegt.

- Einen Additionsoperator +, der zwei Counter-Objekte addiert. Jeder Zähler im Ergebnis-Objekt ist die Summe der beiden Zähler mit demselben Index in den Operanden.
- Einen zweiten Additionsoperator + mit einem **Counter-Objekt** als linken und einem **int** als rechten Operanden.

Der **int** ist die Nummer (der Index) eines Zählers.

Der betreffende Zähler wird im Ergebnis um 1 erhöht, alle anderen Zähler werden unverändert aus dem linken Operanden ins Ergebnis-Objekt übernommen (am einfachsten geht das, wenn du dein Ergebnis-Objekt als Kopie des linken Operanden und nicht als leeres Objekt anlegst!).

- Einen unären Operator ~ mit einem **int**-Ergebnis: Das Ergebnis soll die Summe aller Zählerstände im Array sein.
- Einen Vergleichsoperator <=, der genau dann **true** liefert, wenn alle Zähler des linken Objektes kleinergleich den paarweise entsprechenden Zählern des rechten Objektes sind, und sonst **false**.
- Einen Zuweisungsoperator mit der üblichen Funktion.
- Weiters ist der Ausgabe-Operator << als Funktion zu implementieren. Er soll der Reihe nach (mit Zwischenraum getrennt) für alle Zähler im Array, die nicht 0 sind (und nur für diese!), zählernummer:zählerstand ausgeben.

### Hauptprogramm:

Schreib zum Test deiner Klasse ein Hauptprogramm, das Lottoziehungen simuliert (zur Vereinfachung gehen unsere Lottozahlen von 0 bis 48 statt von 1 bis 49, wir brauchen also **Counter**-Objekte der Größe 49).

- Schreib dazu zuerst eine Funktion, die genau eine Ziehung durchführt, das heißt ein neues **Counter**-Objekt als Returnwert liefert, in dem genau 6 zufällige Zähler den Zählerstand 1 haben und alle anderen Zähler 0 sind. Die Funktion hat keinen Parameter.

Leg dazu ein neues **Counter**-Objekt (mit allen Zählern 0) für das Ergebnis an.

Dann wiederhole Folgendes, solange die Summe der Zählerstände im Ergebnis-Objekt (verwende dafür den ~-Operator!) kleiner 6 ist:

- Berechne eine Zufallszahl von 0 bis 48.
  - Prüfe, ob der Zählerstand dieser Zufallszahl noch 0 ist (denn eine Zahl darf in einer Lotto-Ziehung ja nicht doppelt gezogen werden).
  - Wenn ja: Zähle die neue Zahl, d.h. erhöhe den entsprechenden Zähler im Ergebnis-Objekt (mit dem +-Operator).
- Das Hauptprogramm selbst soll so lange immer wieder eine neue Ziehung mit der Funktion oben durchführen, bis alle Zahlen der neuen Ziehung in den vorangegangenen Ziehungen schon einmal vorgekommen sind.

Verwende dazu zwei Counter-Objekte:

Eines für die Zahlen der aktuellen Ziehung (Returnwert der Funktion) und eines für alle bisher gezogenen Zahlen.

Dass alle Zahlen der aktuellen Ziehung früher schon mindestens einmal gezogen worden sind, erkennst du daran, dass das **Counter**-Objekt der aktuellen Ziehung  $\leq$  dem **Counter**-Objekt mit der Summe aller bisherigen Ziehungen ist.

Wenn das so ist, soll das Programm die Anzahl der durchgeführten Ziehungen, die aktuelle Ziehung, und die Summe aller Ziehungen davor ausgeben und enden.

Wenn nicht, wird die aktuelle Ziehung zu den bisherigen Ziehungen dazugezählt (Operator +) und die nächste Runde gespielt.

Sorge dafür, dass das Programm bei jedem Programmlauf andere Zahlen zieht!

### Zusatzaufgabe:

Versuche, auch verschieden große Arrays richtig zu behandeln.

Grundidee ist, dass nicht vorhandene Zähler als 0 gelten.

- Bei beiden Additionen muss das Array im Ergebnis groß genug für beide Operanden gemacht werden (der Trick bei der zweiten Addition, das Ergebnis-Objekt einfach als Kopie des eigenen Objektes anzulegen, klappt daher nicht mehr).
- Beim Vergleich werden so viele Elemente verglichen, wie der längere Operand hat. Für die nicht vorhandenen Elemente im kürzeren wird 0 verwendet.
- Wenn die Längen bei der Zuweisung unterschiedlich sind, muss das alte Array des linken Operanden vor dem Kopieren des Inhalts freigegeben werden und stattdessen ein neues in der Größe des rechten Operanden angelegt werden.

### Hinweise:

- Für die häufige Berechnung der Länge des längeren Operanden kannst du dir eine Hilfsfunktion für das Maximum zweier Zahlen schreiben.
- Wenn du statt einem direkten Zugriff auf das jeweilige Element des Arrays den Index-Operator verwendest, kannst du auch auf nicht vorhandene Zähler zugreifen und bekommst automatisch 0 als Zählerstand.

Überlege: Wie kannst du den Index-Operator für das eigene Objekt aufrufen?