

# Programmieren C++: Dyn. Array, Operator Overloading

## Klaus Kusche

Gesucht ist eine Klasse, die ein einfaches Array samt Rechenoperationen implementiert.

Du kannst dazu von meiner Webseite einen Programm-Rahmen herunterladen, der **main** bereits enthält (**main** sollst du nicht ändern, nur ev. den Teil mit += entfernen!). Auch das erwartete Ergebnis kannst du dir zum Vergleich auf meiner Webseite ansehen.

Die Klasse **Array** soll zwei private Membervariablen haben:

- Die Größe des Arrays.
- Einen Pointer auf das eigentliche Array.

Es soll ein **double**-Array sein, das im Konstruktor mittels **new** in der gewünschten Größe dynamisch angelegt wird. (wo wird es wohl wieder freigegeben?)

Weiters soll die **Array**-Klasse folgende öffentliche Methoden haben:

- Keinen Standard-Konstruktor.
- Einen Konstruktor mit einem **int**-Argument: Der gewünschten Größe des Arrays. (du musst den Inhalt des Arrays nicht initialisieren)
- Einen Copy-Konstruktor. (Was muss der wohl alles tun???)
- **getSize()**: Returniert die Größe des Arrays.
- **getVal(index)**: Liefert den Wert des Elementes Nummer *index* (*index* beginnt bei 0).  
Ist *index* außerhalb des Arrays, soll 0 zurückkommen.
- **setVal(index, wert)**: Setzt den Wert des Elementes Nummer *index*.  
Liefert ein boolesches Ergebnis: Wahr bei Erfolg, falsch bei ungültigem *index*.

Versuche dann, folgende Operatoren in dieser Klasse **Array** zu implementieren:

- Einen Operator + zur elementweisen Addition zweier Arrays.
- Einen Operator + zur Addition einer Zahl zu allen Elementen des Arrays.
- Einen unären Operator – (bei allen Elementen das Vorzeichen umdrehen).
- Einen Operator == (Vergleich: Die Länge und alle Elemente müssen gleich sein).
- Die normale Zuweisung = (Array auf Array).
- Eine zweite Zuweisung = :  
Zahl auf Array: Die Zahl wird in allen Elementen des Arrays gespeichert.
- Die Ausgabe << .

### Zusatzaufgabe:

- Die kombinierte Addition und Zuweisung +=.

### Hinweise:

- Da alle Methoden bis auf den ersten Konstruktor und **getSize** etwas länger sind, sollten sie nicht inline sein!
- Die Ausgabe << muss eine normale Funktion (nicht Methode!) sein, alle anderen Operatoren sollen Methoden sein.  
 << soll direkt (ohne Get-Methode) auf die Member von **Array** zugreifen, obwohl diese nicht **public** sind. Was muss der Operator << daher sein?
- Bei + und +=: Sind die Arrays verschieden lang, soll das Ergebnis so viele Elemente wie das kürzere der beiden Arrays haben: Die überschüssigen Elemente des längeren Arrays werden ignoriert.  
 (Mathematisch sinnvoller wäre zwar, das Ergebnis so lang wie das längere Array zu machen und für die fehlenden Elemente des kürzeren Arrays 0 zu verwenden, aber der Code dafür ist deutlich länger, und die Mühe sparen wir uns.)
- Bei einer Zuweisung bekommt das Array auf der linken Seite die Größe des Arrays rechts, egal wie lang das Ziel-Array vorher war.  
 Wenn die Arrays ohnehin schon gleich lang sind, solltest du keinen neuen Speicher anlegen!  
 Die Zuweisung soll alle üblichen Regeln und Vorsichtsmaßnahmen beachten! (betreffend Selbstzuweisung usw.)

Tipps:

- Wenn du für alle **int**-Variablen, -Argumente und -Member “**unsigned int**” statt “**int**” als Typ verwendest, dann sparst du dir ein paar lästige Prüfungen, ob die Array-Größe negativ ist.
- Man darf mit **new** auch ein Array der Größe **0** anlegen (solange man auf das Ergebnis nicht zugreift). Auch das Freigeben eines solchen Arrays mit **delete** funktioniert.  
 Diesen Fall muss man also nicht abfangen und extra behandeln.