Methoden für Container

Eine vollständige Liste aller Containermethoden findet man im WWW!

http://www.cplusplus.com/reference

https://en.cppreference.com/

Die folgenden **Methoden** werden von allen Containertypen unterstützt. Sie werden als Methoden aufgerufen: *container_instanz*`.`*methode*`()`

`size()`	Anzahl der Elemente des Containers; Ergebnis `std::size_t` (meist **`unsigned`** `long`)
`front()`	Das erste Elemente des Containers; Ergebnis ist Referenz auf das Element (man kann es ändern)
`back()`	Das letzte Elemente des Containers analog Ergebnis ist Referenz auf das Element
`begin(), end()`	Iteratoren auf das 1., **hinter** das letzte Element Ergebnis: `iterator`
`cbegin(), cend()`	konstante Iteratoren auf das 1., hinter das letzte Element Ergebnis: `const_iterator`
`rbegin(), rend()`	reverse Iteratoren auf das letze, vor das 1. Element Ergebnis: `reverse_iterator`
`crbegin(), crend()`	konstante reverse Iter. auf das letze, vor das 1. Element Ergebnis: `const_reverse_iterator`
`empty()`	ob der Container Elemente enthält Ergebnis: `bool`

Container, die wachsen können, haben auch folgende Methoden:

`push_back(val)`	fügt `val` hinten an den Container an Ergebnis: `void`
`pop_back()`	löscht letztes Argument Ergebnis: `void`
`insert(it, val)`	fügt `val` an der Iterator-Position `it` ein Ergebnis: `void`

Container mit Indexzugriff sind u.a. `std::array und std::vector`

Indexzugriff mit eckigen Klammern	`[3]`
Index-Zugriff mit Überprüfung, ob vorhanden:	`at(3)`

Alle Container außer `std::array` haben auch diverse Konstruktoren. **Allerdings muss man hier aufpassen, dass man die richtigen Klammern (runde oder geschwungene) verwendet.** C++ bricht hier mit der eigenen Empfehlung, dass es solche Unterschiede nicht geben soll.

Alle Container können mit den geschwungenen Klammern mit Werten **initialisiert** werden. **Eine Initialisierung mit den runden Klammern bewirkt immer etwas anderes:**

```
std::vector<int> a{100, 2}, b(100, 2);
```

`a` enthält danach die Elemente `100` und `2` (Initialisierung mit Werten), `b` enthält `100` Elemente `2` (Konstruktor-Aufruf)

Funktionen für Container

Die folgenden **Funktionen** für Container stehen im Namespace **`std`** zur Verfügung. Diese Funktionen funktionieren auch bei C-Arrays und entsprechen sonst den gleichnamigen Methoden. Aufruf: **`std::`***`funktion`*`(`*`container_instanz`*`)`. Da sie umfassender eingesetzt werden können, sollten sie gegenüber den glaichnamigen Methoden bevorzugt verwendet werden. Im Standard gibt es:

`std::size()` Ergebnis `std::size_t` (ein **`unsigned`** Integer mit 64 Bit)
`std::ssize``()` Ergebnis `std::ptrdiff_t` (ein **`signed`** Integer mit 64 Bit)
`std::begin(), std::end()`
`std::cbegin(), std::cend()`
`std::rbegin(), std::rend()`
`std::crbegin(), std::crend()`

Mit der **`std::ssize()`** −Funktion hat C++ endlich eine vernünftige Funktion, um die Größe von Containern als `signed`-Zahl zu erhalten, denn Vergleiche (und Arithmetik) zwischen `signed` und `unsigned` Daten sind zu vermeiden!!!!!!!!!

```
for (int i = 0; i < std::size(c); ++i)        // (böse!!!!)
for (auto i = std::size(c); i >= 0; --i)      // (Endlosschleife!!!!)

for (auto i = static_cast<long long>(size(c)); i >= 0; --i)
            // ok, aber hässlich

for (int i = 0; i < std::ssize(c); ++i)       // endlich
for (auto i = std::ssize(c); i >= 0; --i)     // endlich
```