7.4.0 Arrays

7.4.0.1 Native Container-Klassen

Der Gambas-Interpreter bietet Ihnen zwei Arten von nativen Container-Klassen.

7.4.0.1.1 Collection

Eine Collection ist eine Sammlung von Elementen, die über einen Schlüssel vom *Typ String* indiziert werden → Kapitel 7.5 Collection. Nur Elemente vom Datentyp 'Variant' können in einer Collection gespeichert werden. Die Werte werden intern in einer Hash-Tabelle gespeichert, die dynamisch wächst, wenn mehr Elemente in sie eingefügt werden.

7.4.0.1.2 Array

Ein Array ist eine Menge von Elementen, die über einen Schlüssel vom *Typ Integer* indiziert werden und in einem Speicher-Block gespeichert sind. Alle Elemente in einem Array haben den gleichen Datentyp und es gibt eine Array-Klasse für jeden nativen Datentyp.

Arrays können ein- und mehrdimensional sein. Sind sie mehrdimensional, werden die Elemente von mehr als einem Integer-Wert indiziert. Wenn ein Array genau eine Dimension besitzt, dann kann es mit der Resize()-Methode dynamisch verkleinert oder erweitert werden.

Die Kapitel 7.4.0 bis 7.4.9 widmen sich nicht nur der allgemeinen Klasse *Array* mit seinen Eigenschaften und Methoden, sondern geben auch einen Ein- und Überblick zu den folgenden Themen, wobei die Reihenfolge keine Rangfolge impliziert:

- Native Arrays
- Abgeleitete Arrays
- Eindimensionale Arrays
- Mehrdimensionale Arrays (→ Dimensionen)
- Dynamisches Array
- Inline-Arrays
- Embeded-Arrays
- Deklaration von Arrays
- Einfügen von Elementen in ein Array
- Zugriff auf Elemente eines Arrays; eingeschlossen deren Anzeige
- Array-Kopie
- Export und Import von Arrays
- Konvertierung: Array ↔ Vektor
- Sortierung der Elemente eines Arrays

7.4.0.2 Klasse Array

Die Klasse *Array (gb)* ist die Eltern-Klasse für jede andere Array-Klasse. Ein Array ist als Daten-Container aufzufassen, auf dessen Elemente mit einem numerischen Schlüssel (Index) zugegriffen werden kann.

7.4.0.3 Eigenschaften der Klasse Array

Die Klasse *Array (gb)* verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Bounds	Array.Bounds	Es wird eine virtuelle Klasse zurückgegeben. Die Anzahl der Dimensionen eines Arrays wird durch die Eigenschaft *Array.Bounds.Count* angegeben und die Größe der N-ten Dimension kann man über Array.Bounds[N] abfragen.
Count	Integer	Anzahl der Elemente im Array.
Length	Integer	Anzahl der Elemente im Array. Die Eigenschaften *Count* und *Length* werden synonym verwendet.

Eigenschaft	Datentyp	Beschreibung
Data	Pointer	Zeiger auf die Array-Daten.
Dim	Integer	Anzahl der Dimension eines Arrays.
Max	Integer	Größter Index eines Arrays. Dabei gilt: Array.Max = Array.Count - 1.
Type	Integer	Typ der Daten, die in ein Array eingefügt werden können → http://gambaswiki.org/wiki/cat/constant

Tabelle 7.4.0.3.1: Eigenschaften der Klasse Array (gb)

7.4.0.4 Methoden der Klasse Array

Die Klasse *Array (gb)* verfügt über grundlegende Methoden und spezielle Methoden, die jedoch nicht für alle Array-Klassen zutreffen. Im nächsten Abschnitt werden die grundlegenden Methoden vorgestellt und beschrieben:

Methode	Beschreibung
Clear()	Löscht *alle* Elemente in einem Array.
Remove (Index As Integer [, Length As Integer])	Löscht ein oder mehrere Elemente in einem Array.
Resize (Size As Integer)	Ändert die Größe (Anzahl der Elemente) eines Arrays.

Tabelle 7.4.0.4.1 : Methoden 1 der Klasse Array (gb)

Hinweise:

- Clear(): Löscht alle Elemente in einem eindimensionalen Array und Array.Count gibt danach 0 zurück. Wenn das Array mehrdimensional (→ Kapitel 7.4.4) ist, dann werden die Werte aller Array-Elemente – je nach Datentyp – auf den Default-Wert gesetzt.
- Remove(i,k): Löscht das Element mit dem angegebenen Index=i aus dem Array. Wenn auch der optionale Parameter Length=k mit k>0 gesetzt ist, werden k Elemente aus dem Array ab Array[i] gelöscht. Ist k negativ, dann werden alle Elemente Array[i..Max] im Array entfernt.
- Resize(s): Mit Array.Resize(s) bringt man die Anzahl der Elemente in einem Array auf die Anzahl=s. Die Anzahl der Elemente im Array wird so verkleinert oder vergrößert. Entweder werden überschüssige Elemente *vom Ende entfernt* oder *zusätzliche Elemente angehängt* – entsprechend dem Typ des Arrays mit ihrem Default-Wert bei nativen Arrays (→ Kapitel 7.4.3.1).

In der folgenden Tabelle steht 'Array-Typ' immer für den Daten-Typ eines Elements im Array.

Methode	Beschreibung
Add (Value As Array-Typ [Index As Integer])	Es wird ein neues Element in das Array an der Stelle 'Index' eingefügt. Alle nachfolgenden Elemente werden automatisch um eine Position (nach unten) verschoben. Wird 'Index' nicht angegeben, wird ein neues Element am Ende des Arrays eingefügt.
Copy ([Start As Integer, Length As Integer])	Die Funktion gibt als Funktionswert die 1:1-Kopie des Originals in einem neuen Array-Objekt zurück. Falls das optionale Argument Start=s angegeben ist, wird nur ein Array mit den Elementen ab Startposition s generiert. Ist Length=k angegeben, werden ab der Startposition s nur k Elemente in das neue Array(-Objekt) kopiert.
Delete (Start As Integer [,Length As Integer])	Delete wird synonym für *Extract(..)* verwendet.
Extract (Start As Integer [,Length As Integer])	Es werden ein oder mehrere Elemente im Original-Array ab der Position 'Start=s' gelöscht und (als Funktionswert) in einem Array zurückgegeben. Wird das optionale Argument 'Length=k' angegeben, dann werden ab Position s genau k Elemente aus dem Array gelöscht und als Funktionswert zurückgegeben. Ist k negativ, werden die (-k) Elemente von s an rückwärts extrahiert und zurückgegeben.
Exist (Value As Array-Typ [,Mode As Integer])	Gibt True zurück, wenn das angegebene Element im Array vorhanden ist. Das optionale Argument 'Mode' bestimmt die Vergleichsmethode.

Methode	Beschreibung
Fill (Value As Array-Typ [, Start As Integer, Length As Integer])	Füllt ein Array mit dem vorgegebenen Wert. Ist der optionale Parameter 'Start=s' angegeben, dann wird der vorgegebene Wert ab Start-Position s eingefügt. Ist der Parameter 'Length=k' angegeben (optional), wird der vorgegebene Wert k-fach eingefügt.
Find (Value As Array-Typ [, Mode As Integer, Start As Integer])	Gibt die Position zurück, an welcher das Such-Element zum ersten Mal im Array gefunden wird. Existiert der Such-String nicht im Array, wird als Funktionswert -1 zurück gegeben. Wird das optionale Argument 'Start=s' angegeben, so wird erst ab Index s gesucht. Das Argument 'Mode' bestimmt die Vergleichsmethode.
Insert (Array As Array-Typ[] [, Pos As Integer]) As Array-Typ[]	Fügt die Elemente eines vorgegebenen Arrays in das Original-Array am Ende ein (Standard). Wird jedoch das optionale Argument 'Pos=p' angegeben, so wird das vorgegebene Array in das Original-Array erst ab Position p eingefügt.
Pop( )	Entfernt das *letzte Element* eines Arrays und gibt dieses Element als Funktionswert zurück. Wenn das originale Array leer ist, wird ein Fehler ausgelöst.
Push (Value As Array-Typ)	Fügt ein Element am Ende eines Arrays ein.
Reverse( )	Die Reihenfolge der Array-Elemente wird invertiert – die Array-Elemente $E_0..E_n$ werden umsortiert auf $E_n..E_0$. Das invertierte Array wird als Funktionswert zurückgegeben.

Tabelle 7.4.0.4.2 : Methoden 2 der Klasse Array

7.4.0.5 Array-Klassen für native Datentypen

Gambas hat eine vordefinierte Array-Klasse für *jeden nativen* Datentyp → Kapitel 7.4.3.1. Der Name dieser Klassen ist der Name des Datentyps, dem eckige Klammern [ ] folgen.

```
Boolean[], Byte[], Short[], Integer[], Long[], Single[], Float[], Date[], String[], Object[],
Pointer[] und Variant[]
```

Bitte beachten Sie die folgenden Hinweise, da nicht alle Arrays über alle u.a. Methoden verfügen:

- Für alle (nativen) Arrays – außer Variant[ ] – gibt es die Methode Sort([Mode As Integer]).
- Nur String[ ] hat die Methode *Join([Separator As String, Escape As String])* .
- Für die nativen Arrays Boolean[ ], Byte[ ], Short[ ], Integer[ ], Long[ ], Single[ ], Float[ ], Date[ ], Pointer[ ] existieren die beiden Methoden *Read(Stream As Stream [, Start As Integer, Length As Integer])* und *Write(Stream As Stream [, Start As Integer, Length As Integer])*.
- Die Methode *ToString([Start As Integer, Length As Integer])* kennt nur Byte[ ].
- Nur bei den Arrays Object[ ] können Sie die beiden Methoden *FindByRef(Value As Object [, Start As Integer])* und *ExistByRef(Value As Object)* einsetzen.

Arrays können über diese Methoden in der folgenden Tabelle verfügen:

Methode	Beschreibung
Sort ([Mode As Integer])	Die Elemente im Array werden sortiert. Über Mode kann optional die Sortierreihenfolge festgelegt werden. → gb.Ascent (a..Z , Standard) und gb.Descent (z..A)
Join ([Separator As String, Escape As String]) As String	Verkettet die in den Arrays gespeicherten Zeichenketten. Separator ist eine Zeichenfolge, die zwischen je zwei Strings des Arrays eingefügt werden soll. Das Standard-Trennzeichen ist ein Komma. Escape wird zum Einschließen jeder Zeichenfolge verwendet. Besteht Escape aus zwei Zeichen, dann ist das erste das Start-Escape-Zeichen und das zweite das Ende-Escape-Zeichen. Einzelne Escape-Zeichen in den Strings werden dupliziert. Die Join-Funktion ist das Gegenteil der Split-Funktion.
Read (Stream As Stream [, Start As Integer, Length As Integer])	Füllt ein Array, indem die Daten direkt aus einem Stream gelesen werden. Start gibt an, wo die Daten im Array gespeichert werden. Standardmäßig werden die Daten von Anfang an gespeichert. Length ist die Anzahl der Elemente, die aus der Datei gelesen werden. Standardmäßig werden alle Daten gelesen bis das Ende des Arrays erreicht ist.

Methode	Beschreibung
Write (Stream As Stream [, Start As Integer, Length As Integer])	Schreibt den Array-Inhalt in einen Stream. Start gibt die Position des ersten Elements im Array an, von dem geschrieben wird. Standardmäßig werden die Daten ab erstem Array-Element geschrieben. Length ist die Anzahl der Elemente, die in die Datei geschrieben werden. Standardmäßig werden die Daten geschrieben bis das Ende des Arrays erreicht ist.
ToString ([Start As Integer, Length As Integer]) As String	Der Funktionswert gibt den Array-Inhalt als String zurück. Start ist der Index des ersten zu extrahierenden Bytes. Standardmäßig ist Start gleich 0. Length ist die Anzahl der zu extrahieren Bytes. Standardmäßig wird alles bis zum Ende des Arrays extrahiert.
FindByRef (Value As Object [, Start As Integer]) As Integer	Als Funktionswert wird die Position des ersten Auftretens des spezifizierten Objekts im Array zurück gegeben. Ist Start angegeben, dann beginnt die Suche an der Start-Position. Standardmäßig wird das gesamte Array durchsucht. Wenn das spezifizierte Objekt nicht gefunden werden kann, dann wird -1 als Funktionswert zurückgegeben. Im Gegensatz zu Find() vergleicht die Funktion FindByRef(Value As Object [,Start As Integer]) die Objekte stets anhand ihrer Identität (d.h. ihrer Adresse im Speicher). Die spezielle _compare()-Methode wird nicht benutzt.
ExistByRef (Value As Object) As Boolean	Die Funktion gibt True als Funktionswert zurück, wenn das spezifizierte Objekt im Array gespeichert ist. Im Gegensatz zu Exist() vergleicht die Funktion ExistByRef(Value As Object) die Objekte stets anhand ihrer Identität (d.h. ihrer Adresse im Speicher). Die spezielle _compare()-Methode wird nicht benutzt.

Tabelle 7.4.0.5.1 : Methoden 3 der Klasse Array