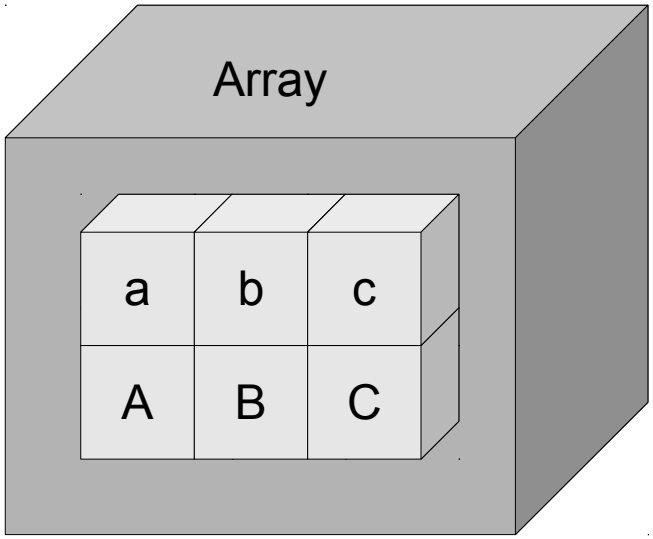
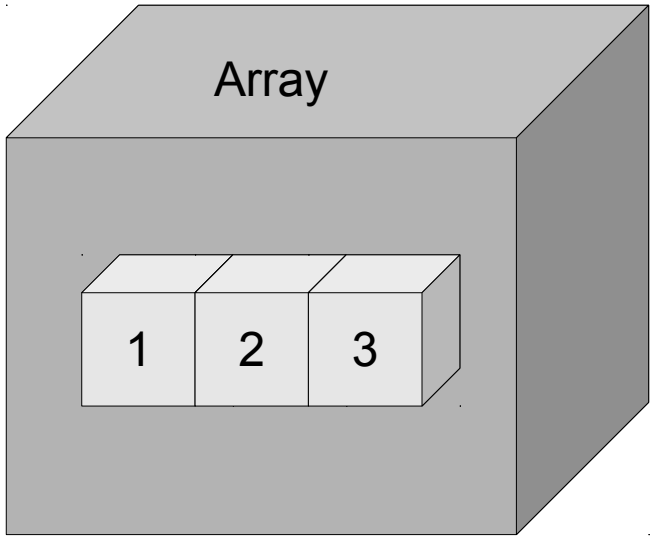
V(isual) B(asic for) A(pplication)
Arrays und Collections



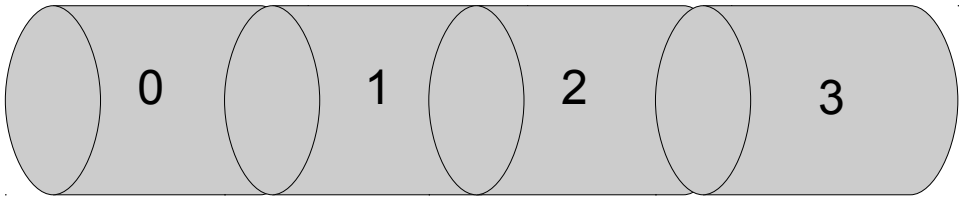


Array (Felder, Vektoren)

- Zusammenfassung von vielen Werten gleichen Datentyps.
- Gruppieren von Variablen zu einem Thema (zum Beispiel Temperaturwerte eines Monats).
- Speicherung einer festen Anzahl von Werten gleichen Datentyps.
- Ein- oder mehrdimensional.

Eindimensionale Arrays

- Folge von Werten gleichen Datentyps.
- Sammlung von Werten zu dem gleichen Thema, wie zum Beispiel durchschnittliche Temperaturen pro Monat. Der Monat wird durch die Position im Array symbolisiert. Der Wert an der Position bildet die Temperatur an.



… deklarieren

```
Dim intMonat(0 To 12) As Integer
Dim dblMessung(12) As Double
```

- Arrays werden mit Hilfe des Schlüsselwortes Dim lokal in einer Subroutine definiert.
- Arrays können auch als globale Variablen definiert werden.

Deklaration

Dim	intMonat	(	0	To	12	)	As	Integer
Dim	[name]	(	[minIndex]	To	[maxIndex]	)	As	[datentyp]

- Der Name eines Arrays ist frei wählbar.
- Direkt im Anschluss an den Namen folgt die unterste und oberste Grenze der Indizes.
- Dem Schlüsselwort As folgt der Datentyp des Array. Jedes Feld in dem Array symbolisiert eine Variable von dem Datentyp ...

Name des Arrays

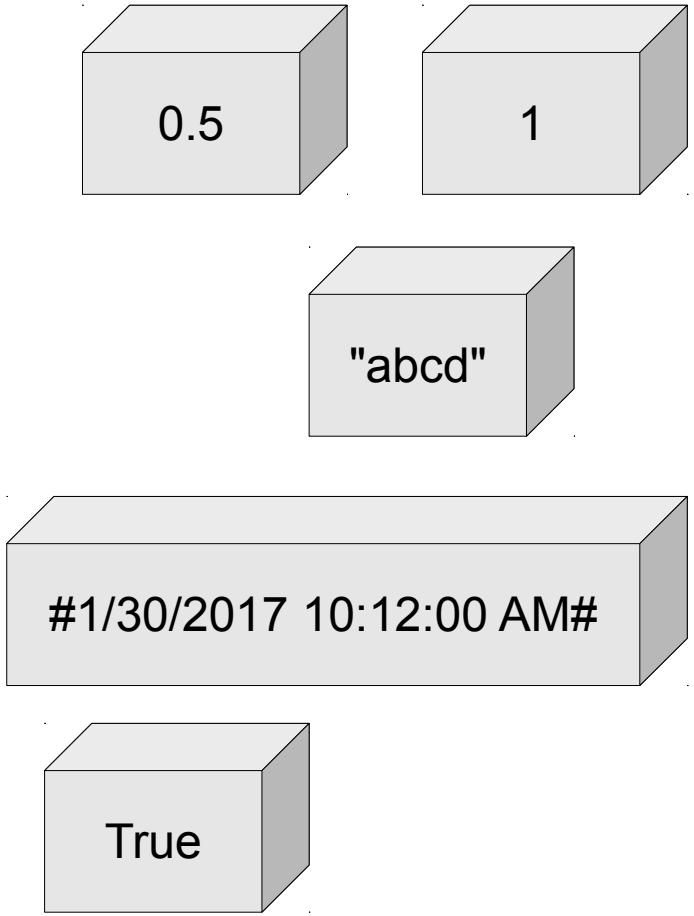
- Eindeutige Bezeichnung innerhalb der Subroutine oder des Moduls.
- Platzhalter für das erste Element in dem Array.

Regeln

- Jeder benutzerdefinierter Name beginnt mit einem Buchstaben.
- Leerzeichen, Punkt, Ausrufezeichen und die Zeichen @, &, $ und das Hash-Zeichen dürfen nicht in einem Namen genutzt werden.
- Der Name ist maximal 255 Zeichen lang.

Datentyp eines Arrays

- Ganz- oder Dezimalzahlen, die als Literale direkt in den Code geschrieben werden. Zahlen können mit Hilfe eines Ausdrucks berechnet werden.
- Einzelne Buchstaben, die zu einem Wort zusammengesetzt werden. Zahlen, die als Text interpretiert werden.
- Datums- und Zeitwerte.
- True (Wahr) oder False (Falsch) als Ergebnis eines Vergleichs von zwei Werten. Boolsche Werte können einer Variablen zugewiesen werden.



Index

Dim	intMonat	(	0	To	12	)	As	Integer
Dim	[name]	(	[minIndex]	To	[maxIndex]	)	As	[datentyp]

- Indizes von Feldern eines Arrays sind immer Ganzzahlen.
- Der Index des letzten Elements muss größer als der Index des ersten Elements sein.

Index des ersten Feldes

Dim	intMonat	(	0	To	12	)	As	Integer
Dim	[name]	(	[minIndex]	To	[maxIndex]	)	As	[datentyp]

- In der runden Klammern wird der Index des ersten Feldes angegeben.
- Falls keine Angaben zu dem Index gemacht werden, beginnt ein Array immer mit dem Index 0.

Index des letzten Feldes

Dim	intMonat	(	0	To	12	)	As	Integer
Dim	[name]	(	[minIndex]	To	[maxIndex]	)	As	[datentyp]

- Der Index des letzten Elements muss immer angegeben werden. Falls keine Angaben zu dem Index des ersten Element gemacht werden, beginnt ein Array immer mit dem Index 0.
- Der Index des ersten und letzten Elements werden mit dem Schlüsselwort To verbunden.

Füllen von Feldern

```
Dim intMonat(0 To 12) As Integer

intMonat(0) = 1
intMonat(1) = 2
```

- Dem Namen des Arrays folgt in runden Klammer der Index.
- Der Index verweist auf ein Feld in dem Array.
- Standardmäßig hat das erste Feld den Index 0. Das zweite Feld hat den Index des ersten Feldes plus eins und so weiter.

Variablen als Index nutzen

```
Dim dblMessung(12) As Double
Dim index As Integer

index = 2
dblMessung(index) = index Mod 2
```

- Der Index kann auch durch eine Variable vom Datentyp „Ganzzahl“ angegeben werden.

Anzahl der Elemente

```
Dim wochentage As Variant
Dim anzahl As Integer

wochentage = Array("So", "Mo", "Di", "Mi", "Do", "Fr", "Sa")

anzahl  = UBound(wochentage) - LBound(wochentage) + 1
```

- Hinweis: Für die Berechnung der Anzahl gibt es keine vordefinierte Funktion.

Untere und obere Index-Grenze

- LBound([arrayname]) ermittelt den Index des ersten Elements in einem Array.
- UBound([arrayname]) ermittelt den Index des letzten Elements in einem Array.
- Ein Unter- und Überschreiten der Grenze löst einen Laufzeitfehler aus.

Nutzung der Funktion Array()

```
Dim wochentage As Variant
Dim index As Integer

wochentage = Array("So", "Mo", "Di", "Mi", "Do", "Fr", "Sa")
```

- Arrays, die immer die gleichen Werte enthalten, können mit Hilfe der Funktion Array() erstellt werden.
- Als Parameter werden der Funktion die zu speichernden Werte übergeben. Die Parameter werden durch Kommata getrennt.
- Der Rückgabewert der Funktion muss einer Variablen vom Typ Variant zugewiesen werden. Eine Aussage über den Datentyp der Felder kann nicht getroffen werden.

Nutzung von Schleifen

```
Dim wochentage As Variant
Dim index As Integer

wochentage = Array("So", "Mo", "Di", "Mi", "Do", "Fr", "Sa")

For index = LBound(wochentage) To UBound(wochentage)
    Debug.Print wochentage(index)
Next
```

Für jedes Elemente in einem Array

```
Dim wochentage As Variant
Dim element As Variant

wochentage = Array("So", "Mo", "Di", "Mi", "Do", "Fr", "Sa")

For Each element In wochentage
    Debug.Print element
Next
```

- Hinweis: In einer foreach-Schleife kann nur eine Variable vom Datentyp Variant oder eine Objekt-Variable genutzt werden.

Array ohne Indexangaben

Dim	menge	(	)	As	Integer
Dim	[name]	(	)	As	[datentyp]

- Die runden Klammern sind leer. Eine Unter- und Obergrenze wurde nicht angegeben.
- Das Array kann beliebig viele Elemente haben.

Dynamische Dimensionierung eines Arrays

```
Dim menge() As Integer
Dim index As Integer

index = 0
ReDim menge(index)
```

- Mit Hilfe des Schlüsselwortes ReDim wird ein Array dimensioniert.
- In diesem Beispiel wird die Obergrenze auf 0 gesetzt. Die Untergrenze ist 0. Das Array enthält ein Element.
- Jedes Element des Arrays enthält entsprechend des Datentyps einen Standardwert. Vorhandene Werte werden gelöscht.

Weitere Möglichkeit

```
Dim menge() As Integer
Dim index As Integer

index = 0
ReDim menge(0 To index)
```

- Mit Hilfe des Schlüsselwortes ReDim wird ein Array dimensioniert.
- In den runden Klammern, die direkt dem Array-Namen folgen, wird die Untergrenze und die Obergrenze angegeben. Die Angaben werden durch das Schlüsselwort To verbunden.

Neu-Dimensionierung eines Arrays

```
Dim menge() As Integer
Dim index As Integer

index = index + 1
ReDim Preserve menge(index)
```

- Mit Hilfe des Schlüsselwortes ReDim Preserve wird ein Array neu dimensioniert.
- In diesem Beispiel wird die Obergrenze durch die Variable index festgelegt. Die Untergrenze ist 0.

Hinweise

- Die Obergrenze wird erhöht. Die, in dem Array gespeicherten Werte bleiben erhalten.
- Die Obergrenze wird vermindert. Elemente, die oberhalb der neuen Obergrenze liegen werden gelöscht. Es findet ein Informationsverlust statt.

Trennung von Strings

```
Dim aktuell As Date
Dim strAktuell As String
Dim datumsangaben() As String

aktuell = Date
strAktuell = CStr(aktuell)
datumsangaben = Split(strAktuell, ".")

Debug.Print "Tag: " & datumsangaben(0)
```

Funktion split()

- Split([string],[trennzeichen]).
- Der erste Parameter definiert den zu trennenden String.
- Der zweite Parameter legt das Trennzeichen fest.
- Der String wird an den Trennzeichen in beliebig viele Bestandteile getrennt.
- Die Funktion gibt die Bestandteile als Array vom Datentyp String zurück.

Verknüpfung von Elemente aus Strings

```
Dim aktuell As Date
Dim strAktuell As String
Dim datumsangaben() As String

aktuell = Date
strAktuell = CStr(aktuell)
datumsangaben = Split(strAktuell, ".")

strAktuell = Join(datumsangaben, ".")
Debug.Print "Datum: " & strAktuell
```

Funktion split()

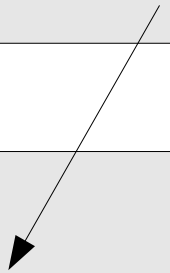
- Join([array],[trennzeichen]).
- Der erste Parameter definiert die zu verknüpfende Array-Elemente.
- Der zweite Parameter legt das Trennzeichen zwischen den Elementen fest.
- Die Funktion gibt einen String, der alle Array-Elemente enthält, zurück.

Verknüpfung von Elementen aus Strings

```
Dim aktuell As Date
Dim strAktuell As String
Dim datumsangaben() As String

aktuell = Date
strAktuell = CStr(aktuell)
datumsangaben = Split(strAktuell, ".")

strAktuell = Join(datumsangaben, ".")
Debug.Print "Datum: " & strAktuell
```

Array als Parameter (call by reference)

```
Dim feldInfo() As String

gesamtsumme = getGesamtsumme(feldInfo)
```



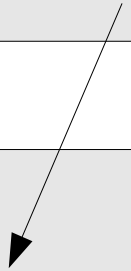
```
Function getGesamtsumme(ByRef info() As String) As Currency

End Function
```

Hinweise

- Als Parameter werden dimensionslose Parameter genutzt.
- Der Name des Arrays verweist auf das erste Element.
- Die Elemente im Array können durch den Aufrufer und die aufgerufene Funktion verändert werden.
- Die Elemente im Array sind nicht vor Veränderungen geschützt.

Array als Parameter (call by value)

```
Dim feldInfo() As String

Call berechneBestellpreis(feldInfo)
```



```
Sub berechneBestellpreis(ByVal info As Variant)

End Sub
```

Hinweise

- Die Parameter müssen vom Datentyp Variant sein.
- Problem: Ist der Parameter ein Array oder eine Variable? Von welchem Datentyp sind die Elemente des Arrays?

Ist der Parameter ein Array?

```
Sub berechneBestellpreis(ByVal info As Variant)
    Dim index As Integer

    If IsArray(info) Then

        For index = LBound(info) To Ubound(info)
                ...
        Next

    End If

End Sub
```

Erläuterung

- Der Funktion IsArray() wird der Name einer Variablen vom Datentyp Variant als Parameter übergeben.
- Falls die Variable als Array interpretiert werden kann, gibt die Funktion True zurück.

Rückgabe eines Arrays

```
Function getBestellmengeAndPreis() As Variant
    Dim info() As String

    getBestellmengeAndPreis = info

End Function
```

- Die Funktion ist vom Datentyp Variant.
- Dem Funktionsnamen wird ein Array übergeben.

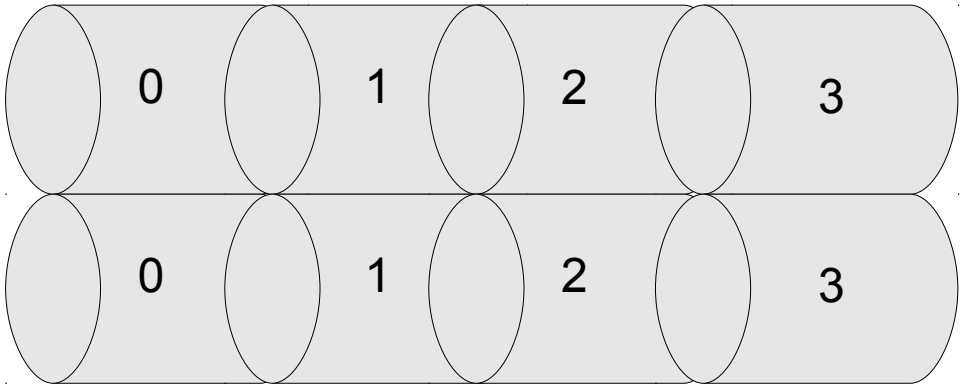
Aufruf der Funktion

```
Dim info As Variant

info = getBestellmengeAndPreis()

If IsArray(info) Then
```

- Der Rückgabewert wird einer Variablen vom Datentyp Variant zugewiesen.
- Mit Hilfe der Funktion IsArray() wird überprüft, ob der „variable Behälter“ als Array interpretiert werden kann.

Mehrdimensionale Arrays

- Darstellung von mehreren Dimensionen.
- Zwei Dimensionen: x-, y-Koordinatensystem; Schachbrett; Matrizen.
- Drei Dimensionen: x-, y-, z-Koordinatensystem.
- Es können bis zu maximal 60 Dimensionen angegeben werden.



Ein- und zweidimensionale Arrays

Tage

(0)	(1)	(2)	...	(30)

Umsätze

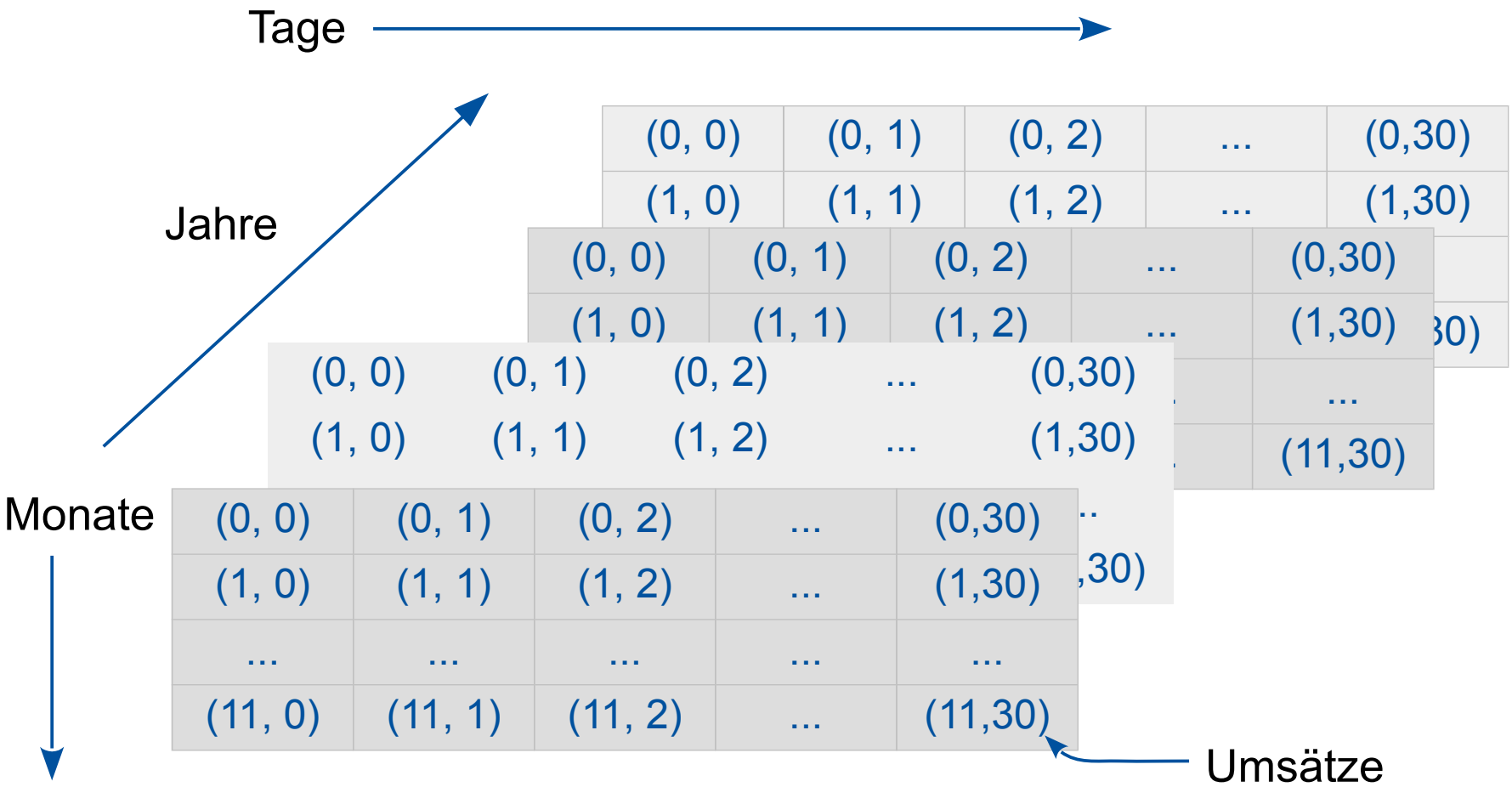
Tage →

Monate ↓

(0, 0)	(0, 1)	(0, 2)	...	(0,30)
(1, 0)	(1, 1)	(1, 2)	...	(1,30)
...	...	...	...	...
(11, 0)	(11, 1)	(11, 2)	...	(11,30)

Umsätze

Dreidimensionale Arrays



… deklarieren

```
Dim monatTag(0 To 30, 0 To 11) As Boolean
Dim temperaturDurchschnitt(30, 11) As Double
```

- Dem Namen des Arrays folgt in runden Klammer die Obergrenze für jede Dimension.
- Die Dimensionen werden durch Kommata getrennt.
- Für jede Dimension kann eine Untergrenze angegeben werden. Untergrenze und Obergrenze werden mit dem Schlüsselwort `To` verbunden.

… füllen

```
monatTag(0, 0) = True
monat(30,1)  = False
```

- Dem Namen des Arrays folgt in runden Klammer der Index für jede Dimension.
- Die Indizes werden durch Kommata getrennt.

Unter- und Obergrenze

```
LBound(monatTag, 1)
UBound(monatTag, 2)
```

- Die Funktion LBound() gibt die Untergrenze eines Arrays zurück.
- Die Funktion UBound() gibt die Obergrenze eines Arrays zurück.
- Den Funktionen wird als erster Parameter der Name des Arrays übergeben. Als zweiter Parameter wird die Dimension übergeben.

Angaben zur Dimension

```
LBound(monatTag, 1)
UBound(monatTag, 2)
```

- Die erste Dimension hat den Index 1. Die zweite Dimension den Index 2 und so weiter.

Nutzung von Schleifen

```
Dim monatTag(0 To 30, 0 To 11) As Boolean
Dim indexMonat As Integer
Dim indexTag As Integer

For indexTag = 0 To UBound(monatTag, 1)

    For indexMonat = 0 To UBound(monatTag, 2)
        ...
    Next indexMonat

Next indexTag
```

Collection (Sammlung)

- Sammlung von Elementen mit unterschiedlichen Datentypen.
- Gruppe von Elementen zu einem Thema.
- Jedes Element in der unsortierten Auflistung hat einen eindeutigen Schlüssel, der einen beliebigen Wert identifiziert.

… deklarieren

```
Dim wochentage As Collection
```

- Die Variable ist vom Datentyp Collection.
- Häufig werden für Variablen Namen in der Mehrzahl genutzt.
- Zum Beispiel die Auflistung .Fields eines Recordsets ist vom Datentyp Collection.
- Die Variable verweist auf ein Objekt.

… anlegen

```
Set wochentage = New Collection
```

- Die Initialisierung einer Objekt-Variablen beginnt mit dem Schlüsselwort Set.
- Das Schlüsselwort New erzeugt ein neues Objekt vom Datentyp Collection.
- Der Verweis auf die neue Sammlung wird der Objekt-Variablen zugewiesen.

Hinzufügung eines Elements

wochentag	.	Add		"Sonntag"	,	"So"
varCollection	.	Add		[value]	,	[key]

- Mit Hilfe der Methode .Add() einer Objekt-Variablen vom Typ Collection wird ein Element der Sammlung hinzugefügt.
- Der Punkt-Operator verbindet eine Methode mit der passenden Objekt-Variablen.

Elemente einer Sammlung

wochentag	.	Add		"Sonntag"	,	"So"
varCollection	.	Add		[value]	,	[key]

- Jedes Element besteht aus einem Schlüssel-Wert-Paar.
- Der Methode wird als erster Parameter ein Wert und als zweiter Parameter ein Schlüssel übergeben.
- Die Parameter werden durch Kommata getrennt.

Wert eines Elements

wochentag	.	Add		"Sonntag"	,	"So"
varCollection	.	Add		[value]	,	[key]

- Der Wert kann von jedem Datentyp sein.

Schlüssel eines Elements

wochentag	.	Add		"Sonntag"	,	"So"
varCollection	.	Add		[value]	,	[key]

- Der Schlüssel identifiziert eindeutig ein Element in einer Sammlung.
- Der Schlüssel muss vom Datentyp String sein.

Anzahl der Elemente in einer Sammlung

```
wochentage.Count
```

- Die Methode .Count gibt die Anzahl der Elemente in der Sammlung zurück.

Rückgabe eines Elements

```
wochentage.Item("Mo")
```

- Mit Hilfe der Methode .Item() wird ein Element aus der Sammlung zurückgegeben.
- Der Methode wird in runden Klammern der Index des gewünschten Elements übergeben.

Alle Elemente einer Sammlung

```
Dim wochentage As Collection
Dim element As Variant

For Each element In wochentage
    Debug.Print element

Next
```

Löschen eines Elements

```
wochentage.Remove("Mo")
```

- Mit Hilfe der Methode .Remove() wird ein Element aus der Sammlung gelöscht.
- Der Methode wird in runden Klammern der Index des zu löschenden Elements übergeben.