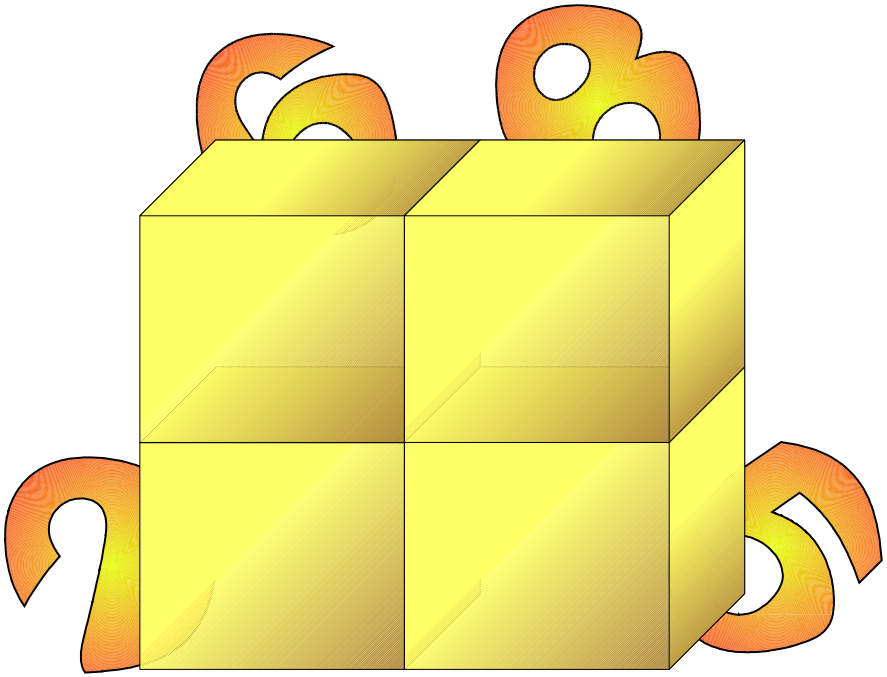
R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – VBA
Arrays, Enumeration und benutzerdefinierte Typen



Array (Feld)

- ... ist aus vielen Variablen, die den gleichen Datentyp besitzen, zusammengesetzt.
- ... fasst Elemente vom gleichen Datentyp zusammen.
- ... gruppiert Variablen zu einem bestimmten Thema.
- ... können bis zu 60 Dimensionen besitzen.

Beispiele

- Die Monatsnamen werden für die Ausgabe gespeichert.
- Temperaturwerte eines Jahres werden für jeden Monat gespeichert.
- Schrauben werden in verschiedenen Größen geliefert.
- Labormesswerte werden zusammengefasst.
- Matrizenberechnungen werden durchgeführt.
- Die Tabelle "Umsatz pro Bezirk und Quartal" wird abgebildet.

Eindimensionale Arrays

- ... sind aufeinander gestapelte Behälter gleicher Größe, aber unterschiedlichen Inhalts.
- ... sind Listen mit einer bestimmten Anzahl von Einträgen. Die Einträge haben alle den gleichen Datentyp.

Graphische Darstellung

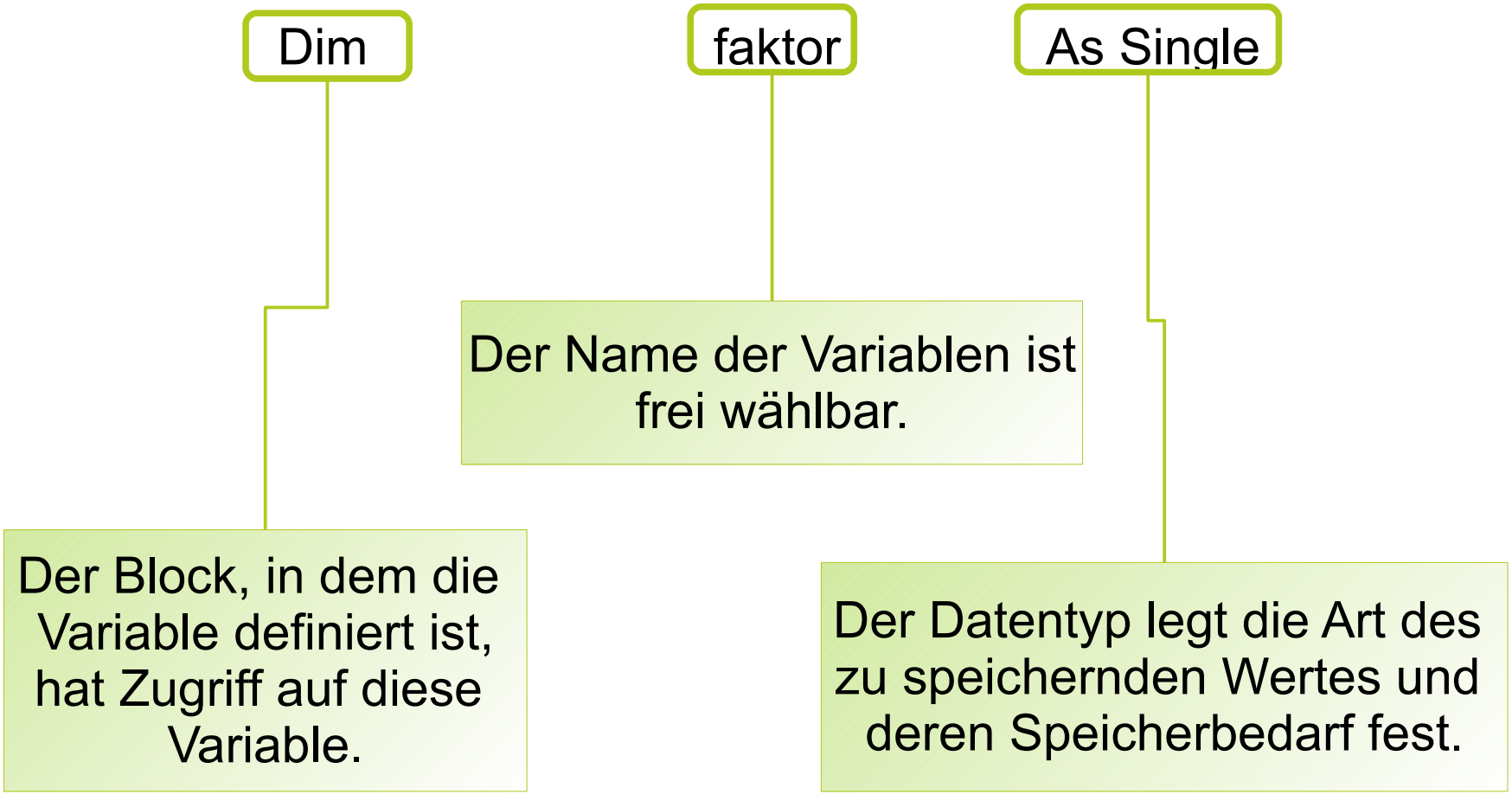
Dim zahl As Integer = 1

1

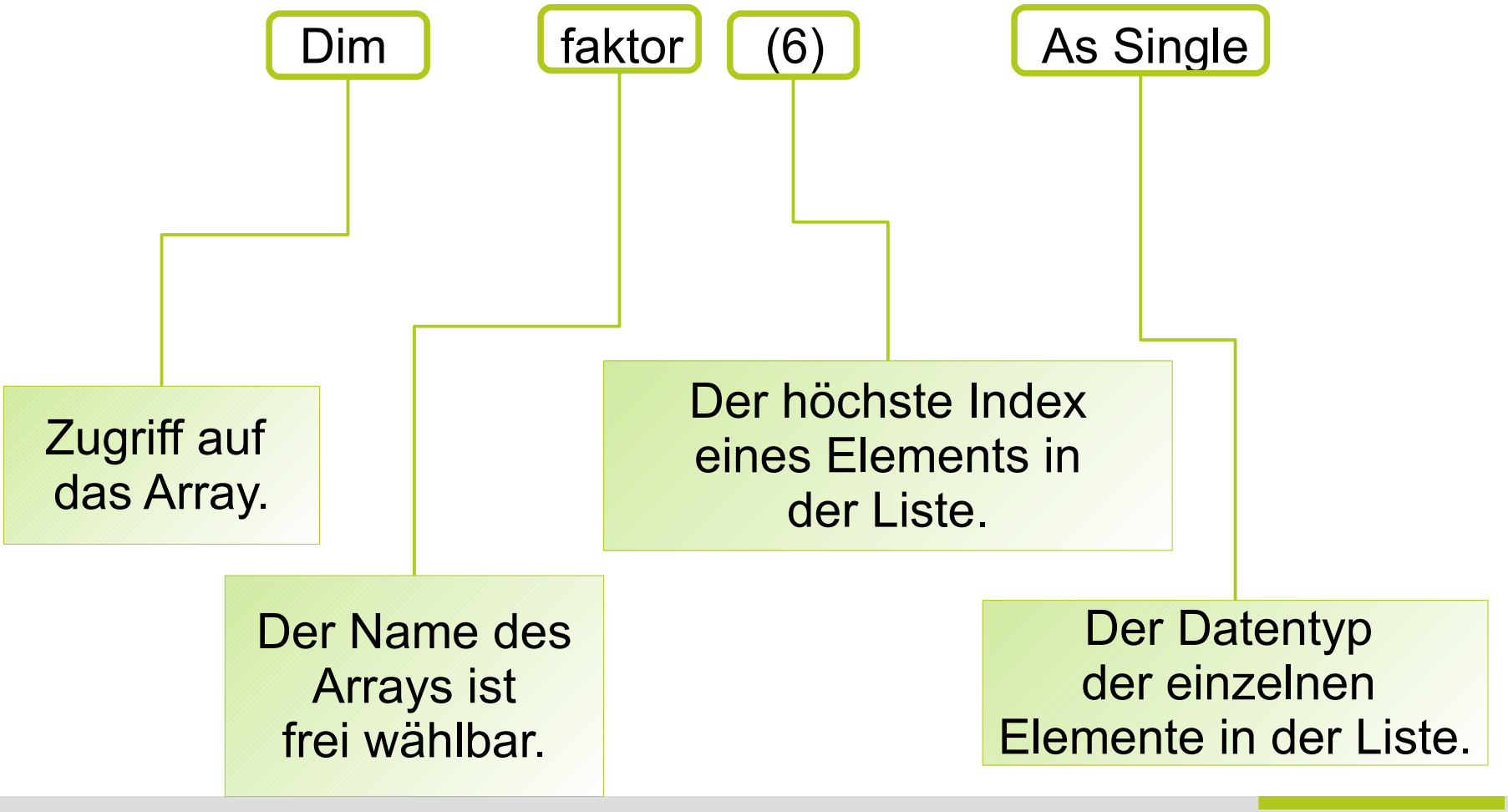
Dim zahl() As Integer = {1, 2, 3, 4, 5}

zahl(0)	1
zahl(1)	2
zahl(2)	3
zahl(3)	4
zahl(4)	5

Variablen deklarieren



Array deklarieren



Möglichkeiten

Dim faktor(6) As Single

- In den runden Klammern wird der höchste Index des Arrays festlegt.
- Der Index des letzten Elements in dem Array wird angegeben.
- In diesem Beispiel hat das Array sieben Elemente. Das erste Element hat den Index 0 und das letzte Element in der Liste hat den Wert 6.

Dim faktor(0 To 6) As Single

- Es wird der Index für das erste und das letzte Element angegeben.
- Es wird ein Bereich für die Indizes der einzelnen Elemente angegeben.

Untergrenze in VBA

- Standardmäßig hat das erste Element in einem Array den Index null.
- Mit Hilfe der Anweisung `Option Base 1` am Anfang eines Moduls kann der Index des ersten Elements für alle Arrays in einem Modul auf eins gesetzt werden.
- Eine Standard-Untergrenze von eins wird aber von einigen VBA-Funktionen nicht beachtet!

Indizes eines Array-Elements

- Voraussetzung: Die Standard-Untergrenze ist null.
- $\text{zahl}(0)$ bezeichnet das erste Element in einem Array.
- $\text{zahl}(3)$ bezeichnet das vierte Element in einem Array.
- $\text{zahl}(\text{anzahl} - 1)$ bezeichnet das letzte Element in einem Array.
- Der Index folgt dem Variablennamen in runden Klammern ohne Leerzeichen.
- Ein Index außerhalb des gültigen Bereichs liefert einen Fehler.

Zugriff mit Hilfe einer for-Schleife

```
Dim intFeld(6) As Integer
Dim nMax As Integer
Dim nMin As Integer
Dim count As Integer

nMin = 0
nMax = 6

For count = nMin To nMax
    intFeld(count) = count
Next count
```

Jedes Element ausgeben

```
Dim intFeld(6) As Integer
Dim nMax As Integer
Dim nMin As Integer
Dim count As Integer
Dim element As Variant

For count = nMin To nMax
    intFeld(count) = count
Next count

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

Es werden alle Elemente nach und nach durchlaufen.
Der Durchlauf beginnt immer beim ersten Element.
Die Variable muss als Variant deklariert werden.

Anzahl der Elemente

```
Dim intFeld(6) As Integer
Dim nMax As Integer
Dim nMin As Integer

nMin = LBound(intFeld)
nMax = UBound(intFeld)

anzahl = nMax - nMin + 1
```

Es ist keine Funktion zur Ermittlung der Länge vorhanden.

Stringfunktionen

- Filter. Datenfelder vom Datentyp String werden nach einem bestimmten Muster durchsucht.
- Split. Eine Zeichenkette wird an einem bestimmten Zeichen getrennt. Die Bestandteile werden in einem Array gespeichert.
- Join. Ein Array von Strings wird zu einer Zeichenkette zusammengefügt.

Nutzung von Filter

```
Sub einlesen()

    For count = 0 To UBound(
        If Range(spalteA & (cou
            bezirk(count) = Rang
                                & Ra
        Else
            Exit For
        End If
    Next count

    found = Filter(bezirk, "Hannover", True)
End Sub
```

Es wird nach dem String Hannover in dem Array bezirk gesucht. In dem Array found werden alle Elemente gespeichert, die dem Suchmuster entsprechen. Falls der dritte Parameter den Wert False hat, werden alle Elemente gespeichert, die nicht dem Suchmuster entsprechen.

Nutzung von Join

```
Sub einlesen()

    For count = 0 To UBound(be
        If Range(spalteA & (cou
            bezirk(count) = Range
                          & Ra
        Else
            Exit For
        End If
    Next count

    element = Join(bezirk, vbCrLf)
End Sub
```

Alle Elemente des Arrays bezirk werden in einer Zeichenkette gespeichert. In diesem Beispiel werden die Elemente in dem String durch einen Zeilenumbruch getrennt. Das Trennzeichen ist beliebig wählbar.

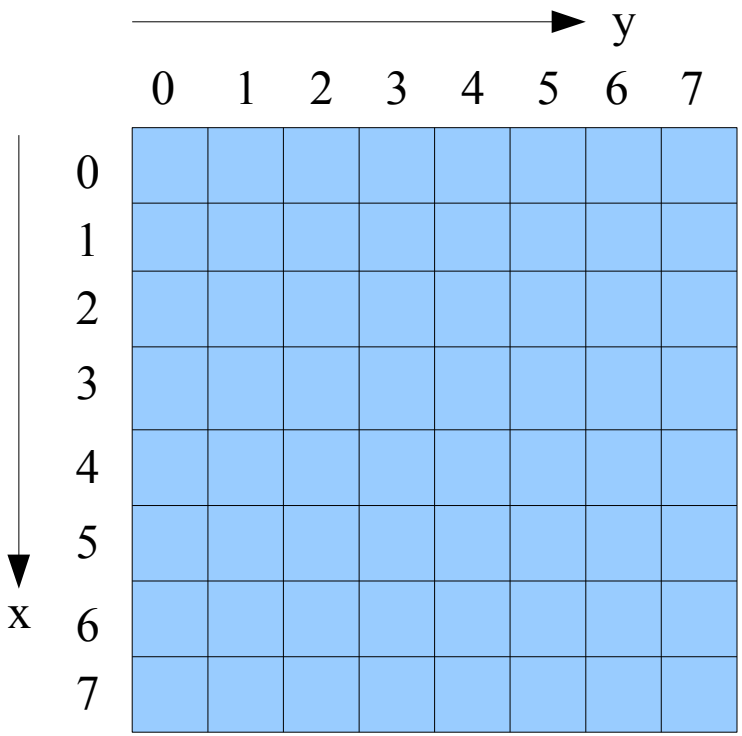
Nutzung von Split

```
Sub einlesen()

    For count = 0 To UBound(
        If Range(spalteA & (cou
            bezirk(count) = Rang
                                & Ra
        Else
            Exit For
        End If
    Next count

    element = Join(bezirk, vbCrLf)
    tmpBezirk = Split(element, vbCrLf, 3)
End Sub
```

Die Zeichenkette wird mit Hilfe eines Zeilenumbruchs in die einzelnen Bestandteile getrennt. Es kann jedes beliebige Zeichen für die Trennung genutzt werden. Der dritte Parameter gibt die Anzahl der Arrayelemente an. Die Angabe -1 trennt die Zeichenkette vollständig.

Zweidimensionale Arrays

- ... entsprechen einem Rechteck, welches durch seine Breite und Länge beschrieben wird.
- ... ist eine Tabelle mit Zeilen und Spalten.



Zweidimensionales Array deklarieren

```
Dim bezirk(5, 1) As String
```

- Die Liste der Dimensionen wird durch runde Klammern begrenzt.
- Die einzelnen Dimensionen werden durch Kommata getrennt angegeben.
- Für jede Dimension wird der Index des letzten Elements angegeben. Es wird der höchste Index angegeben.
- Für jede Dimension kann ein anderer maximaler Index gewählt werden.

Zeilenweise Werte einlesen

```
Sub einlesen()
    Const spalteA As String = "A"
    Const spalteB As String = "B"

    Dim bezirk(5, 1) As String
    Dim zeile As Integer

    For zeile = 0 To UBound(bezirk, 1)
        bezirk(zeile, 0) = Range(spalteA & (zeile + 1))
        bezirk(zeile, 1) = Range(spalteB & (zeile + 1))
    Next zeile
End Sub
```

Alle Werte ausgeben

```
Sub ausgeben()
    Const spalteA As String = "A"
    Const spalteB As String = "B"
    Dim bezirk(5, 1) As String
    Dim zeile As Integer
    Dim spalte As Integer

    For zeile = 0 To UBound(bezirk, 1)
        For spalte = 0 To UBound(bezirk, 2)
            Debug.Print "(" & zeile & ", " & spalte & ") : " _
                                & bezirk(zeile, spalte)
        Next spalte
    Next zeile
End Sub
```

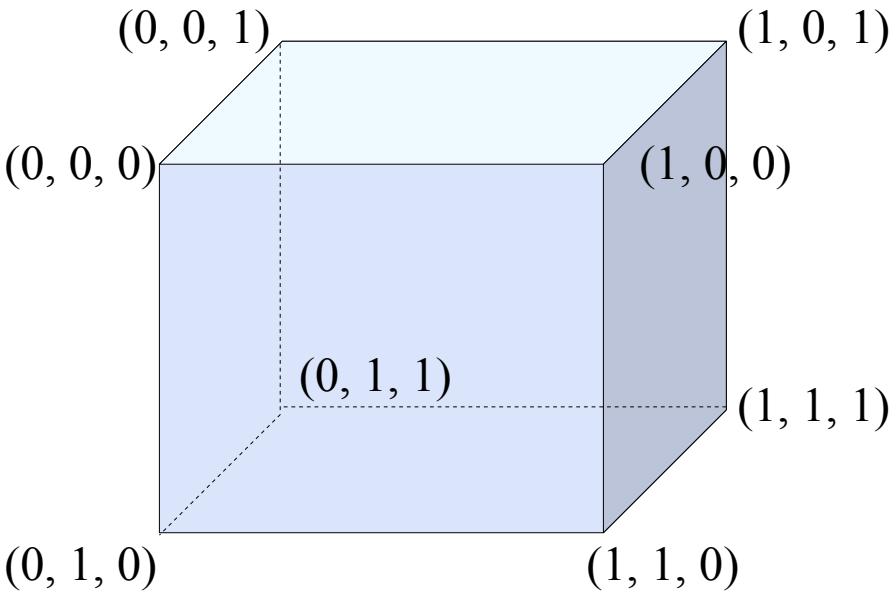
Obergrenze in Abhängigkeit der Dimension

UBound(bezirk, 2)

- Hier wird die Obergrenze der zweiten Dimension abgefragt.
- Die verschiedenen Dimensionen werden von 1 ... n durchnummeriert.
- Wenn keine Dimension angegeben wird, wird die Obergrenze der ersten Dimension abgefragt.
- Die Untergrenze kann für jede Dimension ermittelt werden.

Dreidimensionale Arrays

- ... entsprechen den Ecken eines Würfels.
- ... können mit Hilfe eines Koordinatensystems dargestellt werden.



Dreidimensionales Array erstellen

```
' Für eine bestimmte Anzahl von Orten...
For xCount = LBound(klima, 1) To UBound(klima, 1)
        yCount = 0

        If (Range(spalteA & (tabZeile))) <> "" Then
            klima(xCount, yCount, 0) = Range(spalteA & (tabZeile))
        Else
            Exit For
        End If

        tabZeile = tabZeile + 1
        ...
        tabZeile = tabZeile + 1
Next xCount
```

Dreidimensionales Array erstellen

```
' ... wird pro Monat die Niederschläge und

If (Range(spalteA & (tabZeile))) = "Niederschlag" Then
    For yCount = 1 To quartal
        tabSpalte = Chr(Asc("A") + yCount)
        klima(xCount, yCount, 1) = Range(tabSpalte & tabZeile)
    Next yCount
End If

tabZeile = tabZeile + 1
```

Dreidimensionales Array erstellen

```
' und die Temperatur eingelesen

If (Range(spalteA & (tabZeile))) = "Temperatur" Then
    For yCount = 1 To quartal
        tabSpalte = Chr(Asc("A") + yCount)
        klima(xCount, yCount, 2) = Range(tabSpalte & tabZeile)
    Next yCount
End If
```

Dynamische Arrays deklarieren

```
Dim nFeld() As Single
```

- In den Klammern darf kein maximaler Index angegeben werden.
- Es kann jeder Datentyp genutzt werden.
- Die Größe wird dynamisch im Programm je nach Wunsch festgelegt.

Dimension eines dynamischen Arrays festlegen

```
ReDim nFeld(10)
```

- Der maximale Index wird in den Klammern angegeben.
- Die Werte der Felder werden automatisch gelöscht.
- Es wird ein neues Feld mit dem angegebenen maximalen Index erstellt. In diesem Beispiel hat das Feld 11 Elemente.
- Es können auch mehrdimensionale Felder dynamisch erzeugt werden. Die Dimensionen werden durch Kommata getrennt angegeben.

Dynamische Arrays vergrößern oder verkleinern

ReDim Preserve nFeld(10)

- Der maximale Index wird in den Klammern angegeben.
- Falls der angegebene maximale Index größer ist als der maximale Ursprungsindex, bleiben alle Werte der vorhandenen Felder erhalten. Es werden nur neue Felder ergänzt.
- Falls der angegebene maximale Index kleiner ist als der maximale Ursprungsindex, werden Felder gelöscht. Der Inhalt der Felder geht verloren.
- In einem dynamischen, mehrdimensionalen Feld kann nur die zuletzt angegebene Dimension verändert werden.

Feldelemente an eine Prozedur übergeben

```
Sub addition(zahl1 As Integer, zahl2 As Integer)
    Dim summe As Integer

    summe = zahl1 + zahl2
    Debug.Print zahl1 & " + " & zahl2 & " = " & summe
End Sub

Sub FelderDefinition()
    Dim feld(5) As Integer
    Dim count As Integer

    For count = 0 To UBound(feld) Step 2
        Call addition(feld(count), feld(count + 1))
    Next count
End Sub
```

Felder an eine Prozedur übergeben (Call by Value)

```
Sub additionFeldByVal(ByVal feld As Variant)
    Dim summe As Integer
    Dim count As Integer

    If IsArray(feld) Then
        If TypeName(feld) = "Integer()" Then
            summe = 0
            For count = 0 To UBound(feld)
                summe = summe + feld(count)
            Next count

            Debug.Print "Summe = " & summe
        End If
    End If
End Sub
```

... an eine Prozedur übergeben (Call by Reference)

```
Sub additionFeldByRef(ByRef feld() As Integer)
    Dim summe As Integer
    Dim count As Integer

    summe = 0
    For count = 0 To UBound(feld)
        summe = summe + feld(count)
    Next count

    Debug.Print "Summe = " & summe
End Sub
```

Enumeration

- ... ist eine Aufzählung von Elementen.
- ... fasst Konstanten zu einem bestimmten Thema zusammen.
- Jede Bezeichnung in einer Enumeration symbolisiert einen ganzzahligen Wert.
- ... müssen am Anfang eines Moduls deklariert werden.
- ... sind in Standardmodulen öffentlich (Public).
- ... beginnt mit Enum und endet mit End Enum.

Enumeration deklarieren

```
Public Enum myErrorNumber

    ERR_NOTANUMBER
    ERR_ISDECIMAL

End Enum
```

Die Konstanten einer Enumeration werden von 0 bis n durchnummeriert.

Werte für die Elemente setzen

```
Public Enum myErrorNumber

    ERR_NOTANUMBER = 100
    ERR_ISDECIMAL = 101

End Enum
```

Der Konstanten kann mit Hilfe des Gleichheitszeichens eine positive oder negative Ganzzahl zugewiesen werden.

Anfangswert setzen

```
Public Enum myErrorNumber

    ERR_NOTANUMBER = 100
    ERR_ISDECIMAL

End Enum
```

In diesem Fall wird der ersten Konstanten ein Wert zugewiesen. Der Wert des Nachfolgers wird automatisch um eins erhöht.

Zuweisung an Variablen

```
Dim fehler As myErrorNumber
Dim nr As Long

fehler = myErrorNumber.ERR_NOTANUMBER
' oder
fehler = ERR_NOTANUMBER

nr = fehler
```

Der Wert einer Enumeration-Konstanten wird einer Variablen zugewiesen.
Der Name der Enumeration wird von seiner Konstanten durch ein Punkt getrennt.

Benutzerdefinierte Typen

- Struktur- oder Verbundvariablen.
- ... können ein bestimmtes Objekt beschreiben.
- ... bieten einen Verbund von Variablen, die einer bestimmten Kategorie angehören.
- ... bilden eine bestimmte Struktur von Daten ab.
- ... können nur am Anfang eines Moduls deklariert werden.
- ... sind in Standardmodulen öffentlich (Public).
- beginnen mit Type ... enden mit End Type.

... deklarieren

```
Type KlimaDaten

    stadt As String
    temperatur As Single
    niederschlag As Single

End Type
```

Für die Variablen innerhalb der Struktur kann jeder Standard-Datentyp, ein Array oder ein benutzerdefinierter Typen genutzt werden.
Die Variablen innerhalb der Struktur sind lokal.

... nutzen

```
Dim klima() As KlimaDaten

Select Case tmpText

  Case "Niederschlag"
      klima(count).niederschlag = Range(spalteB & (tabZeile))

  Case "Temperatur"
      klima(count).temperatur = Range(spalteB & (tabZeile))

  Case Else
      count = count + 1
      ReDim Preserve kli
      klima(count).stadt =
End Select
```

Die Variable ist von einem benutzerdefinierten Typ.

Die Variable wird von der Strukturvariablen mit einem Punkt getrennt.