R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Kalenderblatt erstellen

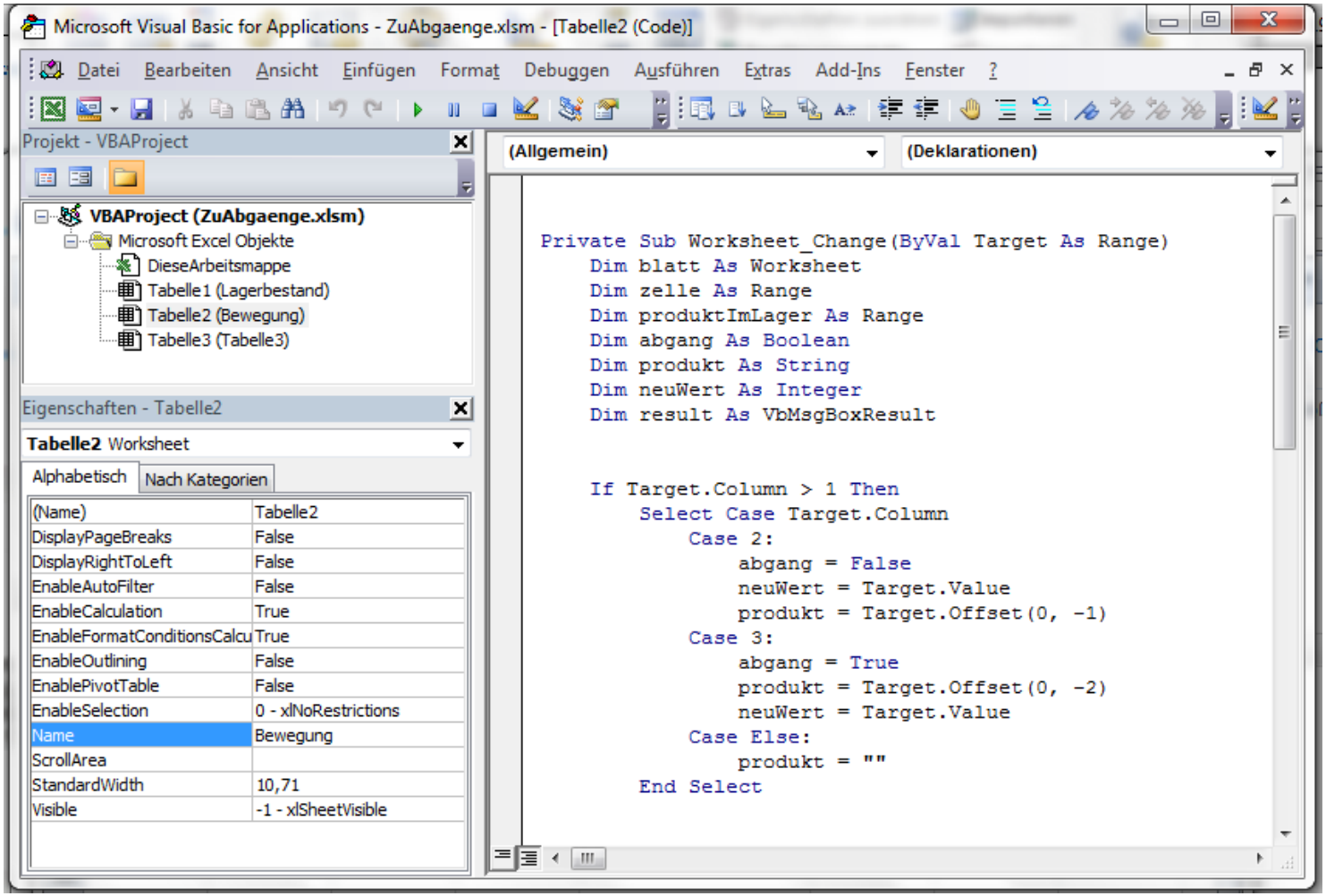
Aufgabe

- Erstellung eines Monatskalenders für ein bestimmtes Jahr.
- Das gewünschte Jahr wird von dem Benutzer in eine bestimmte Zelle geschrieben.
- Feiertage wie Ostern, Weihnachten etc. werden automatisiert gesetzt.
- Hinweis: Diese Aufgabe ist nicht durch die Aufzeichnung eines Makros lösbar.

VBA-Editor öffnen

- Das Menüband Entwicklertools ist eingeblendet.
- Mit einem Klick auf *Visual Basic* in der Gruppe Code wird der VBA-Editor geöffnet.

Beispiel


Microsoft Visual Basic for Applications - ZuAbgaenge.xlsm - [Tabelle2 (Code)]
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Projekt - VBAProject
VBAProject (ZuAbgaenge.xlsm)
Microsoft Excel Objekte
DieseArbeitsmappe
Tabelle1 (Lagerbestand)
Tabelle2 (Bewegung)
Tabelle3 (Tabelle3)
Eigenschaften - Tabelle2
Tabelle2 Worksheet
Alphabetisch
Nach Kategorien
(Name) Tabelle2
DisplayPageBreaks False
DisplayRightToLeft False
EnableAutoFilter False
EnableCalculation True
EnableFormatConditionsCalcu True
EnableOutlining False
EnablePivotTable False
EnableSelection 0 - xlNoRestrictions
Name Bewegung
ScrollArea
StandardWidth 10,71
Visible -1 - xlSheetVisible
(Allgemein)
(Deklarationen)
Private Sub Worksheet_Change(ByVal Target As Range)
Dim blatt As Worksheet
Dim zelle As Range
Dim produktImLager As Range
Dim abgang As Boolean
Dim produkt As String
Dim neuWert As Integer
Dim result As VbMsgBoxResult
If Target.Column > 1 Then
Select Case Target.Column
Case 2:
abgang = False
neuWert = Target.Value
produkt = Target.Offset(0, -1)
Case 3:
abgang = True
produkt = Target.Offset(0, -2)
neuWert = Target.Value
Case Else:
produkt = ""
End Select

VBA-Editor ...

- ist eine integrierte Entwicklungsumgebung (IDE) für die Programmiersprache V(isual)B(asic for)A(pplication).
- ist in jeder Office-Anwendung vorhanden.
- ist eine eigenständige Anwendung, die in der Taskleiste als Symbol eingeblendet wird.
- bietet die Möglichkeit VBA-Code zu lesen und zu bearbeiten.

Aufbau

- Titelleiste zur Anzeige von Informationen.
- Menüleiste sammelt alle Befehle.
- Symbolleiste zeigt die wichtigsten Befehle als Icon an.
- Projekt-Explorer als Schaltzentrale.
- Eigenschaftenfenster zeigt die Attribute eines gewählten Objekts an.
- Codefenster zeigt den Code zu dem gewählten Objekt an.
- Mit Hilfe des Rahmens um den Editor herum, wird das Fenster vergrößert oder verkleinert.

Titelleiste ...

- befindet sich am oberen Rand der Anwendung.
- hat in der linken Ecke ein Icon, welches die Anwendung symbolisiert. Mit einem Klick auf das Icon wird das dazugehörige Systemmenü geöffnet.
- zeigt den Namen der Excel-Datei sowie des aktiven Moduls an.
- hat am rechten Rand Schaltflächen zum Minimieren, Verkleinern / Maximieren oder Schließen der Anwendung.

Menüleiste ...

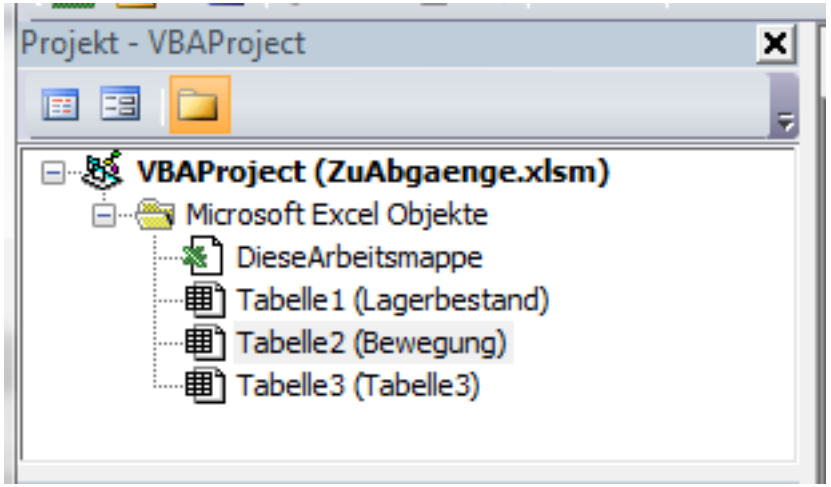
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle zusammengefasst. Der Name der Kategorie gibt einen ersten Hinweis auf die Benutzung der darin enthaltenen Befehle.

Symbolleisten ...

- sammeln häufig genutzte Aktionen zu einem Thema. Die Aktionen werden durch Icons dargestellt.
- beginnen mit der senkrechten gestrichelten Linie am linken Rand. Sobald die Maustaste über diese Linie liegt, kann die Symbolleiste mit Hilfe von Drag (Maustaste gedrückt) & Drop (Maustaste loslassen) verschoben werden.
- werden über das Menü *Ansicht* – *Symbolleiste* ein- oder ausgeblendet.
- haben einen Pfeil nach unten am rechten Rand. Mit einem Klick auf diesen Pfeil wird ein Menü zum Ein- und Ausblenden von einzelnen Icons in der Symbolleiste angezeigt.

Projekt-Explorer ...

- ist die Schaltzentrale für die Programmierung einer Excel-Anwendung.
- verwaltet die, zu dem Projekt gehörende Arbeitsmappe sowie die darin enthaltenen Arbeitsblätter.
- zeigt aufgezeichnete Makros in Modulen an.
- ist frei platzierbar.
- kann über das Menü *Ansicht* ein- oder ausgeblendet werden.



Aufbau des Projekt-Explorers

- In der Titelleiste wird die Schließen-Schaltfläche am rechten Rand angezeigt.
- Darunter befindet sich eine Symbolleiste mit den folgenden Icons
  - „Zeige zum gewählten Element den Code an“.
  - „Zeige das passende Objekt zum Code an“.
  - „Sortierung mit Hilfe von Ordnern“.
- Unterhalb der Symbolleiste werden die Module alphabetisch sortiert oder in Abhängigkeit ihres Typs angezeigt.

Module ...

- kapseln Code zu einem Thema.
- sind Container für Code. In dem Container wird eine bestimmte Aufgabe gelöst.
- fassen Programmiercode und Deklarationen zu einem Thema zusammen.
- werden automatisch durch die Aufzeichnung eines Makros angelegt.
- können vom Entwickler oder der Anwendung angelegt werden.

Module im Projekt-Explorer

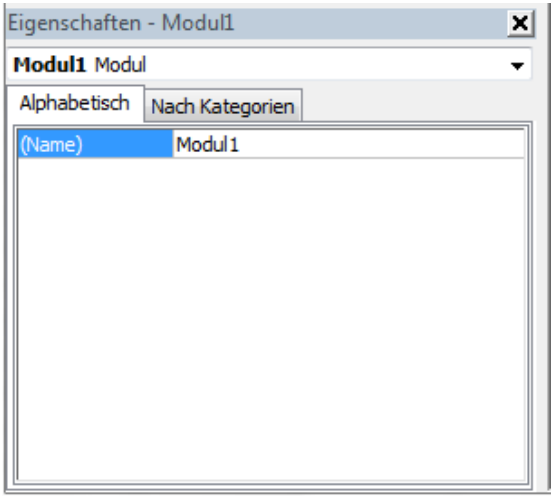
- werden mit Hilfe von Ordnern dargestellt.
- (Microsoft Excel Objekte). Die Module sind an die Arbeitsmappe oder an ein Tabellenblatt gebunden.
- (Module). In diesem Ordner werden allgemeine Module abgelegt. Allgemeine Module werden bei der Makro-Aufzeichnung automatisch erstellt oder vom Entwickler angelegt. Allgemeine Module stehen nicht in Abhängigkeit zu einem Excel-Objekt.

Hinzufügung eines allgemeinen Moduls

- Das Menüband Entwicklertools ist aktiv.
- Mit Hilfe des Symbols *Visual Basic* wird der VBA-Editor geöffnet.
- Mit Hilfe des Menüs *Einfügen* – *Modul* wird ein allgemeines Modul erzeugt. Das neu erzeugte Modul wird im Codefenster angezeigt.
- In diesem Modul wird der Code mit Hilfe der Tastatur eingegeben.

Benennung eines Moduls

- Im Eigenschaftenfenster wird der Name des Moduls eingegeben.
- Durch einen Klick in das Textfeld rechts von der Bezeichnung wird die Einfügemarke gesetzt.
- Mit Hilfe der Maus kann der vorhandene Name markiert werden.
- Mit Hilfe der Tastatur kann der vorgegebene Name durch einen benutzerdefinierten Namen ersetzt werden. Der Name sollte Auskunft über die Aufgabe des Moduls geben.



Die Anweisung ”Option Explicit„ ...

- sollte immer am Anfang eines Moduls stehen
- erzwingt die Deklaration von Variablen vor deren Verwendung. Variablen sind Platzhalter für veränderbare Werte.
- kann manuell mit Hilfe der Tastatur eingegeben werden.
- kann automatisch bei Einfügung des Moduls gesetzt werden (*Extras – Optionen*; Registerkarte *Editor*; *Variablendeklaration erforderlich*).

Schlüsselworte in VBA

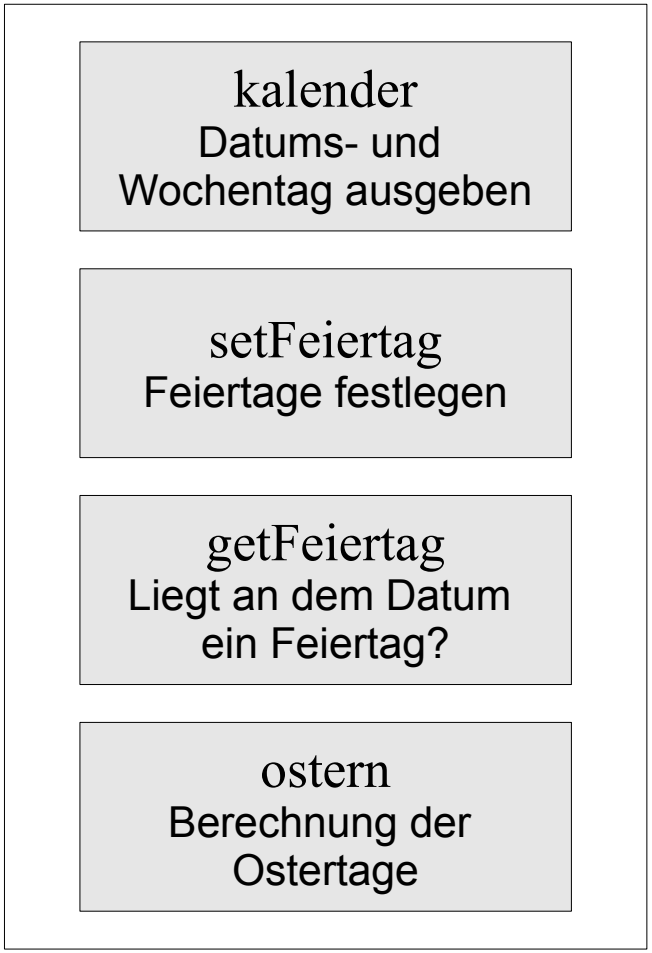
```
Sub KalenderErstellen()


End Sub
```

- Schlüsselworte werden in VBA standardmäßig in blauer Schrift dargestellt.
- Die Farben des Editors werden mit Hilfe von *Extras – Optionen;* Registerkarte *Editorformat* eingestellt.

Subroutinen ...

- fassen Anweisungen in einem Modul zusammen.
- können Werte übergeben bekommen. Diese Werte werden in der Subroutine verarbeitet.
- können einen Wert an den Aufrufer zurück geben.
- lösen ein Teilproblem der Gesamtaufgabe.
- können von verschiedenen Positionen aufgerufen werden.

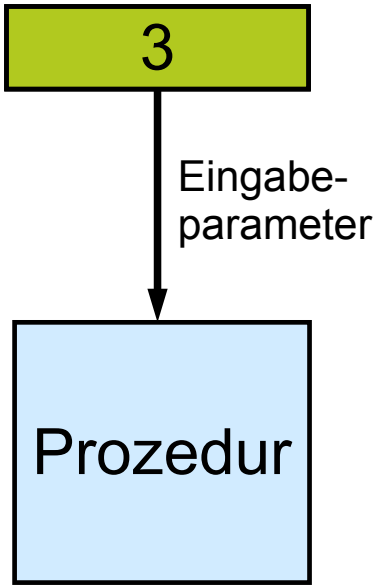
Erstellung eines Kalenders



Modul:
Monatskalender

Prozeduren ...

- können Argumente übergeben werden.
- haben Argumente als Startwerte, die in der Prozedur verarbeitet werden.
- Wie die Daten verarbeitet werden, ist dem Nutzer nicht bekannt. Der Nutzer kennt nur den Prozeduraufruf.



Gerüst einer Prozedur

```
Sub KalenderErstellen()


End Sub
```

- Jede Prozedur hat einen Kopf. Der Prozedurkopf beginnt mit Sub.
- Dem Prozedurkopf folgen die, zu der Prozedur gehörenden Anweisungen. Die Anweisungen werden eingerückt Zeile für Zeile im Prozedurrumpf geschrieben.
- Die Prozedur endet mit den Schlüsselwörtern End Sub. Sobald die Eingabe des Prozedurkopf mit Hilfe der Eingabetaste abgeschlossen wird, werden die Schlüsselworte End Sub automatisch gesetzt.

Prozedurkopf

```
Sub KalenderErstellen()
```

- Der Prozedurkopf beginnt mit dem Schlüsselwort Sub.
- Dem Schlüsselwort folgt ein benutzerdefinierter Name
- Dem benutzerdefinierten Namen folgen die runden Klammern. Die Klammern fassen die zu übergebenen Werte zusammen.

Benutzerdefinierte Namen

```
Sub KalenderErstellen()


End Sub
```

- Benutzerdefinierte Namen werden in VBA standardmäßig in schwarzer Schrift dargestellt.
- Prozedurnamen sollten über die Aufgabe der Prozedur Auskunft geben.
- Es sollten nur die Buchstaben des Alphabets sowie die Zahlen von 0 bis 9 genutzt werden.
- In zusammengesetzten Namen beginnt jedes Wort mit einem Großbuchstaben.

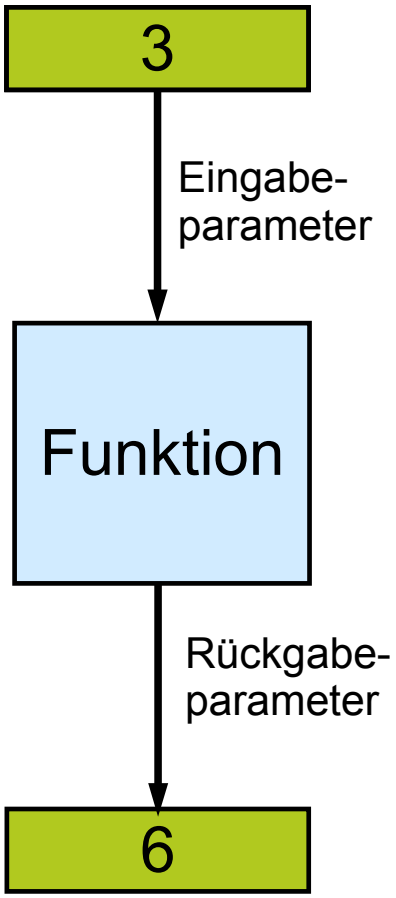
Prozeduren aufrufen

```
Call prozedur([argument], [argument])
prozedur [argument], [argument]
```

- Die Prozedur wird mit Hilfe ihres Namen in jeder beliebigen Zeile einer Subroutine aufgerufen
- Dem Namen folgt die Argumentliste. Die Argumentliste kann geklammert werden, muss aber nicht.
- Falls die Argumentliste geklammert wird, muss das Schlüsselwort `Call` für den Aufruf genutzt werden.

Funktionen ...

- können Argumente übergeben werden.
- haben Argumente als Startwerte, die in der Prozedur verarbeitet werden.
- Wie die Daten verarbeitet werden, ist dem Nutzer nicht bekannt. Der Nutzer kennt nur den Prozeduraufruf
- können einen Wert an den Aufrufer zurückgeben. Wie dieser Wert berechnet wurde, weiß der Nutzer nicht.



Gerüst einer Funktion

```
Function getFeiertag(datum As String, feiertag As Collection) As String


End Function
```

- Durch das Schlüsselwort Function wird der Beginn einer Funktion gekennzeichnet. Die Funktion endet mit den Schlüsselwörtern End Function.
- Dem Schlüsselwort Function folgt ein benutzerdefinierter Name. Mit Hilfe dieses Namens kann die Funktion aufgerufen werden.

Funktionskopf

```
Function getFeiertag(datum As String, feiertag As Collection) As String
```

- Dem Schlüsselwort Function folgt ein benutzerdefinierter Namen.
- Dem benutzerdefinierten Namen folgt in runden Klammern die Argumentliste.
- Am Ende des Kopfes wird der Funktion mit Hilfe von As ein bestimmter Typ zugewiesen. Diese Funktion gibt einen Text zurück. Der Datentyp der Funktion sollte mit dem Datentyp des Rückgabewertes übereinstimmen.

Argumentliste

```
Function getFeiertag(datum As String, feiertag As Collection)
```

- Eine Argumentliste besteht aus vielen verschiedenen Argumenten unterschiedlichsten Datentyps.
- Die Argumente in der Liste werden durch Kommata getrennt.
- Jedes Argument wird durch einen eindeutigen Namen gekennzeichnet. Mit Hilfe des Schlüsselwortes `As` wird jedem Argument ein bestimmter Datentyp zugewiesen.

Datentypen ...

- beschreiben den Rückgabewert einer Funktion oder den Inhalt eines Arguments.
- legen Regeln für die Verwendung des Wertes fest.
- beschreiben den Datenbereich des Platzhalters.
- legen den Speicherbedarf fest.
- Standard-Datentypen sind für Ganzzahlen, Dezimalzahlen, Datums- und Zeitwerte sowie Zeichenfolgen vorhanden.

… für Ganzzahlen

	Speicherbedarf in Bytes	Datenbereich
As Byte	1	0 - 255
As Integer	2	-32.768 - +32.767
As Long	4	-2.147.483.648 - +2.147.483.647
As Boolean	2	0 (falsch, false) <> 0 (wahr, true)

Datentypen für Dezimalzahlen

- As Single beschreibt ein Wert von einfacher Genauigkeit. Der Platzhalter enthält eine Näherung an eine reelle Zahl.
- As Double beschreibt ein Wert von doppelter Genauigkeit. Der Platzhalter enthält eine Näherung an eine reelle Zahl. Double-Zahlen benötigen mehr Speicher als Single-Zahlen.
- As Currency besitzt 15 Stellen vor dem Dezimalkomma und vier Stellen nach dem Dezimalkomma. Dieser Wert sollte für Geldbeträge etc. genutzt werden.

Weitere Datentypen

- As Date für Datums- und Zeitwerte.
- As String für die Speicherung von Zeichenketten
- As Boolean für die Speicherung des Zustandes ”wahr„ oder ”falsch„.

Rückgabewert festlegen

```
Function getFeiertag(datum As String, feiertag As Collection) As String
        GetFeiertag = "Neujahr"
End Function
```

- Der Name der Funktion ist ein Platzhalter für einen Speicherbereich in einer definierten Größe.
- Mit Hilfe des Gleichheitszeichens wird der Funktion ein Wert zugewiesen.
- Diesen Wert kann der Aufrufer weiterverarbeiten

Funktionen aufrufen

```
strFeiertag = getFeiertag(aktuellDatum, allFeiertage)
```

- Die Funktion wird mit ihren Namen aufgerufen.
- In runden Klammern folgt die Argumentliste so wie im Kopf angegeben.
- Mit Hilfe des Gleichheitszeichens wird einer Variablen der Wert der Funktion zugewiesen.

Variablen ...

- sind Platzhalter für veränderbare Werte.
- speichern Zahlen, Datums- und Zeitwerte sowie Texte.
- bekommen einen berechneten Wert zugewiesen.
- sind von einem bestimmten Standard-Datentyp. Die verschiedenen Datentypen werden im Internet auf der Seite http://msdn.microsoft.com/en-us/library/gg278937.aspx aufgelistet.

Beispiel

```
Sub KalenderErstellen()

    Dim aktuellDatum As Date
    Dim ersterTagImJahr As Date
    Dim jahr As String
    Dim monat As String
    Dim countTag As Integer
    Dim zeile As Long
    Dim spalte As Long

End Sub
```

… definieren

```
Dim aktuellDatum As Date
[Zugriff] nameVariable As [Datentyp]
```

- Jede Variable hat einen eindeutigen Namen (aktuellDatum). Der Name ist ein Platzhalter für einen Wert von einem bestimmten Datentyp.
- Mit Hilfe des Schlüsselwortes As wird der Variablen ein Datentyp zugewiesen. In diesem Beispiel wird eine Variable vom Typ Date (Datum) erzeugt.
- Auf die Variablen kann nur innerhalb der Prozedur KalenderErstellen zugegriffen werden (Dim). Diese Prozedur ist Besitzer dieser Variablen. Die Variable ist lokal in dieser Prozedur bekannt.

Zugriff auf Variablen

```
Dim aktuellDatum As Date
[Zugriff] nameVariable As [Datentyp]
```

- Das Schlüsselwort vor dem Namen der Variablen legt den Zugriff auf die Variable fest.
- In diesem Beispiel kann nur die Prozedur KalenderErstellen auf die Variable aktuellDatum zugreifen.
- Das Schlüsselwort Dim beschreibt immer einen lokalen Zugriff auf eine Variable in einer Subroutine.

Regeln für den Variablennamen

- Jeder benutzerdefinierter Name beginnt mit einem Buchstaben.
- In der Programmierung sollten nur die Buchstaben a..z, A..Z, der Unterstrich und die Zahlen 0..9 genutzt werden.
- Es dürfen keine Schlüsselwörter wie zum Beispiel Dim, Sub als Variablennamen genutzt werden.
- Jeder Variablenname wird in einer Subroutine oder Modul nur einmal genutzt.
- Ein Variablenname hat maximal 255 Zeichen, sollte aber nicht länger als 40 Zeichen sein.

Geeignete Variablennamen ...

- geben Auskunft über die Nutzung des Platzhalters.
- beschreiben den zu speichernden Wert.
- entsprechen dem Sprachraum des Entwicklers.
- sind an die Sprache der realen Welt angelehnt.

Beispiele

Variable für …	Geeignet	Nicht lesbar
Bestellnummer	bestellnummer strBestellNr	bnr bn
Rechnungsdatum	rechnung_Datum dateRechnung	rgD dat1203
Bestellmenge	bestellmenge intMenge	m bm
Warenpreis	preisProEinheit artikelpreis	cur pr

In diesem Beispiel genutzte Datentypen

Datentyp	Erläuterung
As String	Eine beliebige Anzahl von Zeichen.
As Date	Datums- und Zeitwerte im Bereich von 1.Januar 100 bis 31. Dezember 9999 00:00:00 – 23:59:59.
As Long	Ganzzahlen im Bereich -2.147.483.648 bis +2.147.483.647.
As Integer	Ganzzahlen im Bereich von -32.768 bis +32.767.

Variablen initialisieren

```
ersterTagImJahr = ("1.1." & Range("B1").Value)
aktuellDatum = ersterTagImJahr
zeile = 3
spalte = 2
monat = 0
countTag = 1
```

- Der Wert rechts vom Gleichheitszeichen wird der Variablen links zugewiesen.
- Speichere in zeile den Wert 3. Die Variable ist ein Platzhalter für einen bestimmten Wert.
- Die Variablen bekommen einen definierten Startwert zugewiesen.

Anweisungen ...

- verändern den Wert einer Variablen.
- berechnen einen Wert mit Hilfe von Ausdrücken.
- rufen Prozeduren und Funktionen auf.
- symbolisieren einen bestimmten Arbeitsschritt.
- werden in VBA mit dem Ende der Zeile abgeschlossen.
- werden Zeile für Zeile ausgeführt.

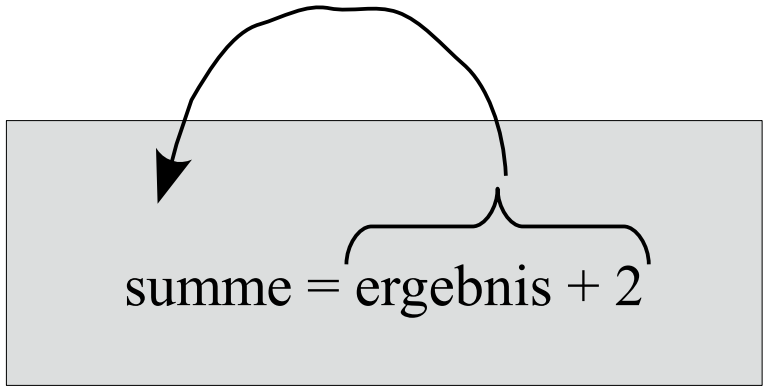
Ausdrücke ...

- berechnen mit Hilfe von Operatoren und Operanden einen neuen Wert. Der berechnete Wert wird einer Variablen oder einer Eigenschaft eines Objekts mit Hilfe des Gleichheitszeichens zugewiesen.
- verändern den Wert von Variablen oder Attributen entsprechend des angegebenen Datentyps.
- nutzen arithmetische Operatoren, um Werte zu berechnen.
- nutzen Vergleichsoperatoren, um zwei Werte zu vergleichen.

Operator „Gleichheitszeichen“

- in Vergleichen / Bedingungen: [ausdruck] ist gleich [ausdruck]
- in Berechnungen: [variable] ← [ausdruck]. Der, in einem Ausdruck, berechnete Wert, wird einer Variablen zugewiesen.

(summe = ergebnis) => wahr / falsch



Arithmetische Operatoren

Operator	Rechenart	Beispiel
+	Positives Vorzeichen	ergebnis = +3
-	Negatives Vorzeichen	ergebnis = -3
+	Addition	ergebnis = 3 + 4
-	Subtraktion	ergebnis = 3 - 4
^	Potenzrechnung	ergebnis = 3^4

Arithmetische Operatoren

Operator	Rechenart	Beispiel
*	Multiplikation	ergebnis = 3 * 4
/	Division	ergebnis = 3 / 4 ergebnis = 0.75 Fehler = 3 / 0
\	Ganzzahlige Division	ergebnis = 3 \ 4 ergebnis = 0
Mod	Modula (Division mit Rest). Nur für Ganzzahlen.	ergebnis = 3 % 4 ergebnis = 3 ergebnis = 4 % 3 ergebnis = 1

Nutzung von Dezimalzahlen

- Dezimaltrennzeichen: Punkt.
- Führende Nullen werden entfernt.
- Eine Null als Nachkommastelle wird durch das #-Zeichen ersetzt.
- Das Programm kennt keine Maßeinheiten, Längenangaben etc.
- Dezimalzahlen, die in Variablen vom Datentyp Ganzzahlen gespeichert werden, verlieren ihre Nachkommastellen.

Konvertierungsfunktionen

- CInt([variable]) konvertiert einen beliebigen Wert in einen Interger-Wert.
- CLng([variable]) konvertiert einen beliebigen Wert in einen Long-Wert.
- CSng([variable]) konvertiert einen beliebigen Wert in einen Single-Wert.
- CDbl([variable]) konvertiert einen beliebigen Wert in einen Double-Wert.
- CCur([variable]) konvertiert einen beliebigen Wert in einen Currency-Wert.
- Fix([variable]) gibt den ganzzahligen Anteil einer Zahl zurück. Die Funktion gibt bei einer negativen Zahl einen Wert zurück, der größer oder gleich der Variablen ist.

Interne Speicherung von Datums- und Zeitwerten

- Der Wert 0 entspricht dem Datum 30.12.1899. Das Datum 15.12.2010 entspricht einem Wert von 40527.
- Negative Ganzzahlen beziehen sich auf ein Datum vor dem 30.12.1899.
- Ein Tag entspricht dem Wert 1. Die Nachkommastellen bilden die Uhrzeit ab.
  - 1 Stunde ⇨ 1/ 24 = 0.0417
  - Mitternacht ⇨ 0
  - Mittag ⇨ 12 * 1/24 = 0.5

Konstante Datums- und Zeitwerte

```
datum = #2/23/2011#
uhrzeit = #1:20:00 PM#
```

- Konstante Datums- und Zeitwerte werden durch das #-Zeichen begrenzt.
- Datumswerte werden in dem Format Monat/Tag/Jahr angegeben. Es wird das kurze Datumsformat genutzt. Wenn möglich, sollten keine zweistelligen Jahresangaben genutzt werden.
- Zeitwerte werden in dem Format Stunde : Minute : Sekunde eingegeben. Zeitangaben werden in einem 12- oder 24-Stunden-Format in Abhängigkeit des Computers dargestellt.

Tag, Monat aus einem Datum extrahieren

- Day([datum]) extrahiert die Tagesangabe aus einem Datumswert (Datentyp Date).
- Month([datum]) extrahiert die Monatsangabe aus einem Datumswert (Datentyp Date).
- Year([datum]) extrahiert die Jahresangabe aus einem Datumswert (Datentyp Date).

Addition von Datumswerten

```
aktuellDatum = DateAdd("d", countTag, ersterTagImJahr)
aktuellDatum = DateAdd([Intervall], [Anzahl], [Datum])
```

- Datum + x von Tagen, Monaten, Jahren etc.
- Das Intervall wird als String angegeben. In diesem Beispiel wird eine bestimmte Anzahl von Tagen zu dem angegebenen Datum addiert.
- Die verschiedenen Intervall-Möglichkeiten werden in der Hilfe der Funktion beschrieben.

Datumswert erstellen

```
LichtenbergOstern = DateSerial(CInt(jahr), 3, ostern)
LichtenbergOstern = DateSerial([jahr], [monat], [tag])
```

- Die Funktion DateSerial gibt ein Wert vom Datentyp Date zurück.
- Die Jahresangabe sollte immer vierstellig sein. Die Jahresangabe kann Werte von 100 bis 9999 annehmen
- Die Monats- und Tag-Angaben sind Integer-Werte. Für den Monat sollten Werte zwischen 1 und 12 sowie für den Tag Werte zwischen 1 und 31 genutzt werden.
- Standardmäßig wird der gregorianischen Kalender genutzt. Mit Hilfe der Eigenschaft VBA.Calendar kann der Wert verändert werden.

Berechnung von Ostern

- Die Gaußsche Osterformel berechnet die Daten aus der Jahresangabe. Die Osterformel kann für Datumsberechnung ab 1583 (Umstellung auf den Gregorianische Kalender) genutzt werden.
- 1997 Modifizierung der Formel durch Heiner Lichtenberg.
- Eine Erläuterung der beiden Formeln befindet sich in der Datei kalenderFeiertage.xlsm.

Strings verbinden

```
ersterTagImJahr = ("1.1." & Range("B1").Value)
person = vorname & " " & nachname
```

- Mit Hilfe des kaufmännischen Unds werden Texte verbunden.
- In dem ersten Beispiel wird ein konstanter Text ("1.1.") mit den Inhalt der Zelle B1 verbunden.
- In dem zweiten Beispiel werden Variablen vom Datentyp String mit einem Leerzeichen verbunden.

Hinweise zu Strings

- Konstante Strings werden durch die Anführungsstriche begrenzt.
- Strings können alle Zeichen der ANSI-Zeichentabelle enthalten.
- Telefonnummern, Postleitzahlen werden in diesem Format gespeichert.
- Zahlen, die als Strings definiert sind, können nicht in Berechnungen genutzt werden.

Text formatieren

```
varString = Format(aktuellDatum, "DDD")
varString = Format([String], [Formatzeichen])
```

- Der Funktion Format wird der zu formatierenden String sowie ein Formatstring übergeben.
- Der Formatstring besteht aus verschiedenen Zeichen, die die Art der Formatierung beschreiben.

Formatstring

```
varString = Format(aktuellDatum, "DDD")
varString = Format([String], [Formatzeichen])
```

- Der Formatierungsstring kann vorgefertigt sein (z.B. „Currency“).
- Der Formatierungsstring kann sich aus verschiedenen Formatzeichen zusammengesetzt werden. In dem Code werden der Formatstring mmmm für den Monatsnamen und DDD für die Angabe der Wochentage, abgekürzt.
- Eine Auflistung finden Sie im Web unter http://msdn.microsoft.com/en-us/library/gg251755.aspx.

Bedingte Anweisungen ...

- haben einen Kopf If [Bedingung] Then und enden mit dem Schlüsselwort End If.
- führen Anweisungen in Abhängigkeit eines Vergleiches zweier Werte aus. Der Vergleich ist wahr oder falsch.
- können verschachtelt werden.
- Wenn (If) die Bedingung wahr ist, dann (Then) führe die dazugehörigen Anweisungen aus.
- In dem nachfolgenden Code-Beispiel wird abgefragt, ob der Wert der Variablen kleiner ist als der aktuelle Monat. Wenn ja, wird zu der nächsten Spalte gesprungen.

Beispiel

```
If (monat < Month(aktuellDatum)) Then
        zeile = 3                       '  Anfangszeile setzen

        If monat > 0 Then               ' Neue Spalte berechnen
            spalte = spalte + 2
        End If

End If
```

Bedingungen ...

- sind Ausdrücke, die wahr oder falsch sind.
- vergleichen Werte.
- nutzen Vergleichsoperatoren.
- überprüfen das boolsche Ergebnis von vordefinierten Funktionen, die mit „Is“ beginnen.
- können verknüpft werden.
- werden häufig in runde Klammern gesetzt. Mit Hilfe der Klammern wird eine Rangfolge der Auswertung vorgegeben. Runde Klammern erhöhen die Lesbarkeit von Ausdrücken.

Vergleichsoperatoren

Operator	Erläuterung	Beispiel
=	ist gleich	3 = 4 *falsch*
<>	ungleich	3 <> 4 *richtig*
<	kleiner als	3 < 4 *richtig*
<=	kleiner gleich	3 <= 4 *richtig*
>	größer	3 > 4 *falsch*
<=	größer gleich	3 >= 4 *falsch*

Hinweise

- Der zu vergleichende Wert und der Vergleichswert sollten den gleichen Datentyp besitzen.
- Gleitkommazahlen sollten nicht für Vergleiche „ist gleich“ genutzt werden. Gleitkommazahlen speichern immer nur einen Nährungswert.

Anweisungen wiederholen

```
Do


Loop Until (Day(aktuellDatum) = 31) And (Month(aktuellDatum) = 12)
```

- Die Schleife beginnt mit Do.
- Die Anweisung zwischen Do und Loop werden x-Mal wiederholt.
- Falls keine Abbruchbedingung angegeben wird, läuft die Schleife endlos. Das Programm stürzt ab.

Abbruchbedingung

```
Do


Loop Until (Day(aktuellDatum) = 31) And (Month(aktuellDatum) = 12)
```

- Eine Abbruchbedingung kann dem Schlüsselwort Do oder Loop folgen.
- In diesem Beispiel werden die, zu der Schleife gehörenden Anweisungen mindesten einmal durchlaufen. Die Schleifenbedingung befindet sich im Fuß der Schleife.
- Die Abbruchbedingung
  Until (Day(aktuellDatum) = 31) And (Month(aktuellDatum) = 12)
  wird am Ende des Schleifen-Durchlaufs überprüft.

Until- oder While-Bedingungen

```
Do


Loop Until (Day(aktuellDatum) = 31) And (Month(aktuellDatum) = 12)
```

- Die Abbruchbedingung kann mit den Schlüsselwörtern While oder Until beginnen.
- While (Solange): Solange die Abbruchbedingung wahr ist, werden die Anweisungen wiederholt.
- Until (Bis): Bis die Abbruchbedingung erfüllt ist, werden die Anweisungen wiederholt.

Abbruchbedingung in diesem Beispiel

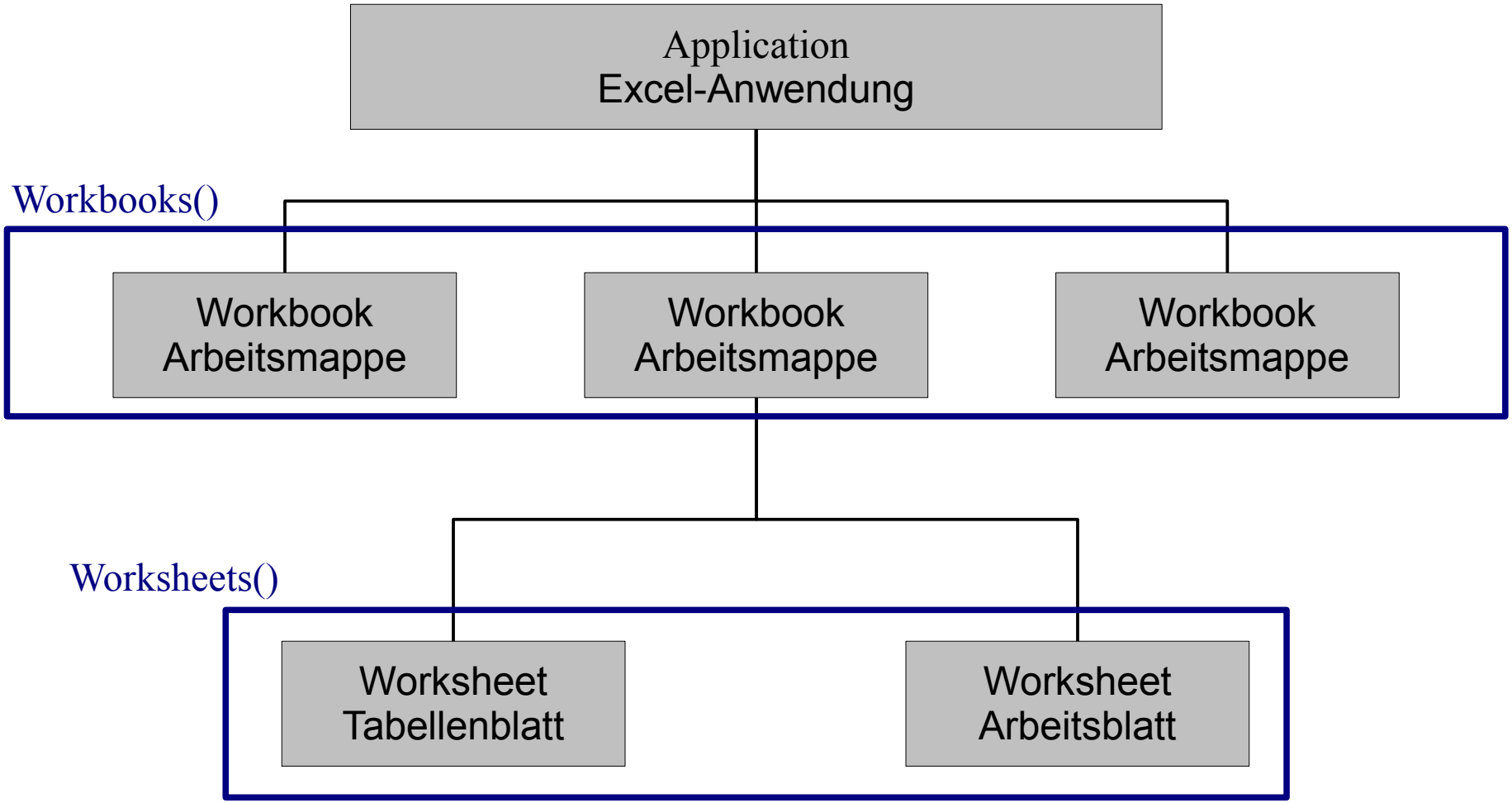
```
Do


Loop Until (Day(aktuellDatum) = 31) And (Month(aktuellDatum) = 12)
```

- Abbruch: Am 31. Tag des zwölften Monats.
- AND verknüpft zwei Bedingungen. Beide Bedingungen müssen wahr sein. Andernfalls bricht die Schleife nicht ab.
- Die zwei Bedingungen werden, um die Lesbarkeit zu erhöhen, geklammert. Bei der Verknüpfung von Bedingungen muss die Rangfolge der Operatoren beachtet werden.

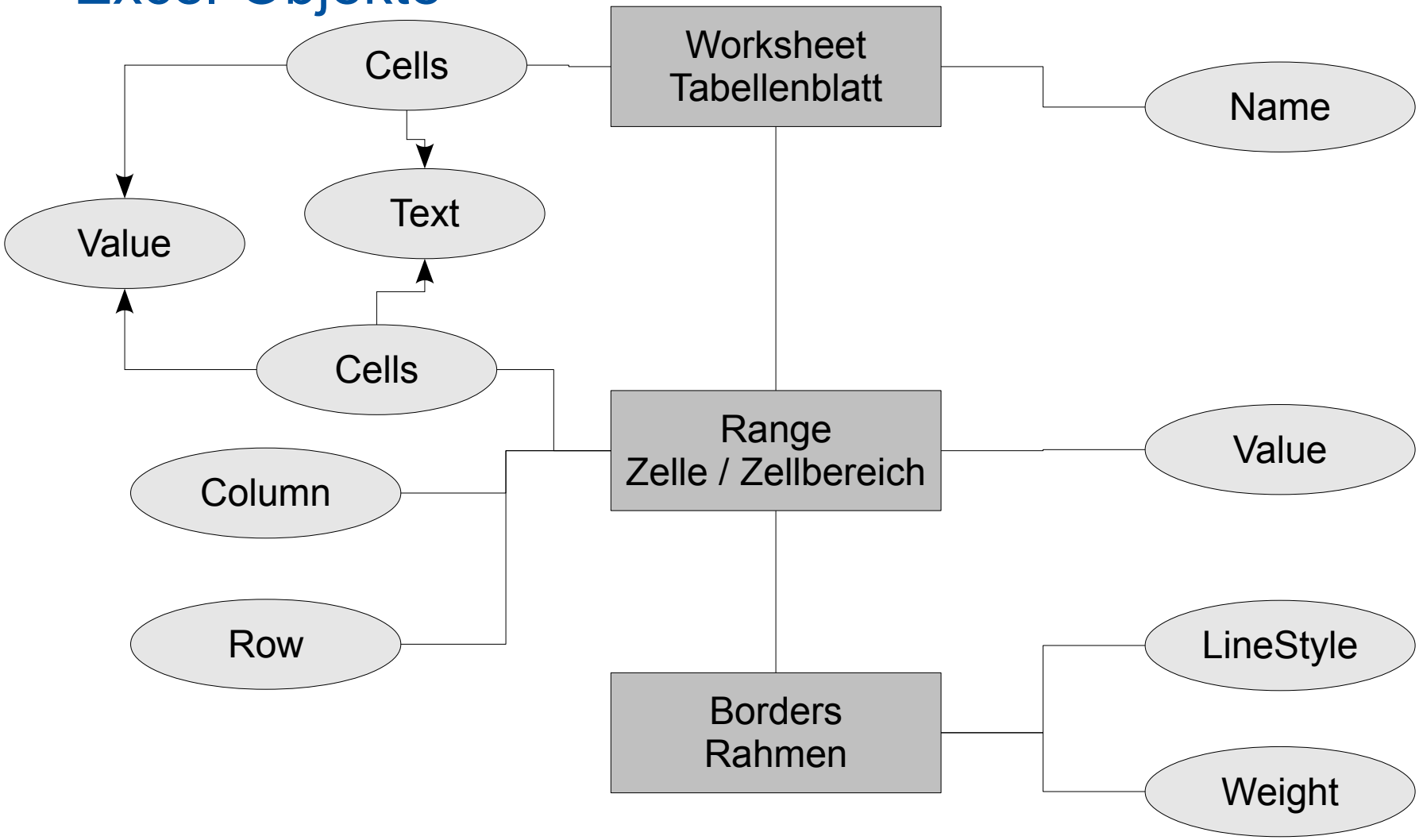
Verknüpfungsoperatoren für Bedingungen

Operator	Erläuterung	Beispiel
And	und	(3 > 4) And (3 < 10) *richtig* (5 > 6) And (5 < 4) *falsch* Alle Bedingungen müssen zutreffen.
Or	oder	(3 > 4) Or (3 < 10) *richtig* (5 > 6) Or (5 < 4) *richtig* Eine der Bedingungen muss zutreffen.
Not	nicht	Negation einer Bedingung Not(3 > 4) *richtig* Not(3 < 4) *falsch*

Excel-Objekte



Excel-Objekte



Objekte ...

- werden häufig durch ein Rechteck dargestellt.
- beschreiben ein Element der Excel-Anwendung.
- sind hierarchisch geordnet.
- haben Attribute (Eigenschaften), die das Aussehen des Objekts beschreiben. Zum Beispiel .Name gibt die Bezeichnung eines Arbeitsblattes wieder.
- haben Methoden, die die Attribute verändern. Zum Beispiel .Close schließt die Arbeitsmappe.

Attribute (Eigenschaften) ...

- beschreiben ein Excel-Objekt.
- Jedes Objekt hat die gleichen Attribute. Jedes Objekt unterscheidet sich aber von anderen Objekten durch die Attributwerte.
- werden mit dem dazugehörigen Objekt durch ein Punkt verbunden.

Range ...

Range("A1")				
	Range("B3:C4")			

- beschreibt eine Zelle, Zeile, Spalte oder Zellbereich.
- nimmt auf einen Zellbereich Bezug.

Möglichkeiten

- Range("A1") bezieht sich auf die angegebene Zelle „A1“. Der Range-Bereich umfasst eine Zelle.
- Range("A1:D6") bezieht sich auf den zusammenhängenden Zellbereich „A1:D6“. Der Range-Bereich umfasst alle Zellen („A1“, „A2“, ..., „B1“, „B2“, ...) des angegebenen Zellbereichs.
- .UsedRange bezieht sich auf den verwendeten Bereich in einer Arbeitsmappe.

Borders

```
Range(Cells(zeile - 1, spalte), Cells(zeile - 1, spalte + 2))
        .Borders(xlEdgeLeft).LineStyle = xlContinuous
        .Borders(xlEdgeLeft).Weight = xlThin
        .Borders(xlEdgeRight).LineStyle = xlContinuous
```

- Beschreibung des Rahmens um eine Zelle oder Zellbereich.
- Der einzelnen Ränder werden durch einen Parameter in den runden Klammern (xlEdgeLeft, xlEdgeRight, xlEdgeTop, xlEdgeBottom) festgelegt.
- .Borders ohne eine Randangabe bezieht sich auf den gesamten Rahmen.

Eigenschaften des Rahmens

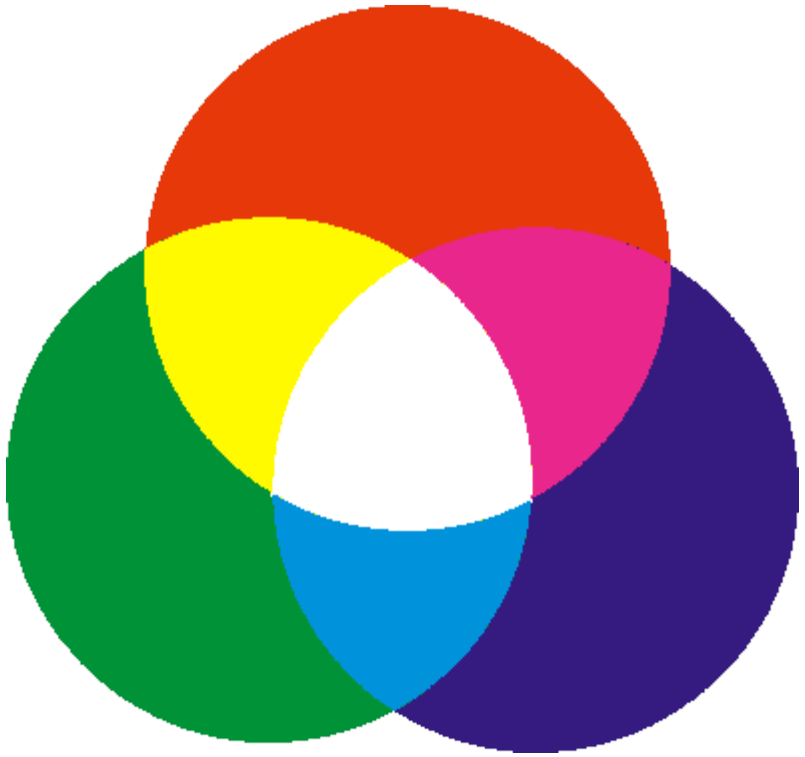
- .Colors legt die Rahmenfarbe (RGB) fest.
- .ThemeColor nutzt Designfarben in Abhängigkeit des gewählten Designschemas.
- .ThinAndShade. Aufhellung oder Abdunklung der Rahmenfarbe.
- .LineStyle legt die Rahmenart fest.
- .Weight legt die Rahmendicke fest.

Farben angeben

- Die Prozedur RGB([R], [G], [B]) mischt eine Farbe für den Bildschirm. Der Funktion wird der Rotanteil, der Grün-Anteil und der Blau-Anteil der Farbe übergeben.
- Designfarben sind auf der Webseite http://msdn.microsoft.com/en-us/library/office/bb216356%28v=office.12%29.aspx aufgelistet.

RGB-Farben

- ... werden für die Darstellung von Farben am Bildschirm genutzt.
- Das RGB-Farbsystem addiert (mischt) Licht in verschiedenen Farben.
- Jede Farbe (Rot, Grün, Blau) wird in 256 Helligkeitsstufen unterteilt.
- Um so mehr sich eine Farbe Weiß annährt, um so heller wird diese.



Beispiele

Farbe	RGB	Farbkonstante
	(0, 0, 0)	vbBlack
	(255, 255, 255)	vbWhite
	(255, 0, 0)	vbRed
	(0, 255, 0)	vbGreen
	(0, 0, 255)	vbBlue
	(255, 255, 0)	vbYellow
	(255, 0, 255)	vbMagenta
	(0, 255, 255)	vbCyan

Farben abdunkeln oder aufhellen

- .ThinAndShade = -1. Dunkelste Farbe.
- .ThinAndShade = 0. Orignalfarbe.
- .ThinAndShade = 0.5. Hellere Farbe.
- .ThinAndShade = 1. Hellste Farbe.

Auflistung Cells ...

Cells(1)				
		Cells(4,3)		

- ist eine Auflistung aller Zellen eines Tabellenblattes oder Zellbereichs.
- beschreibt mit Hilfe des Zeilen- und Spaltenindex eindeutig eine Zelle.
- Der Zeilen und Spaltenindex kann mit Hilfe einer Variablen angegeben werden.

Range mit Cells mischen

```
Range(Cells(zeile - 1, spalte), Cells(zeile - 1, spalte + 2))
Range("A1", Cells(2, 3))
```

- Der erste Parameter legt die Position der oberen, linken Zelle fest.
- Der zweite Parameter legt die Postion der unteren, rechten Zelle fest.
- Durch die Angabe mit Hilfe von Cells wird ein Zellbereich definiert.

Eigenschaften einer Zelle

- .Value speichert den aktuellen Wert einer Zelle ohne Formatierungen. Der Wert einer Zelle kann gelesen oder verändert werden.
- .Text enthält den, am Bildschirm angezeigten Inhalt einer Zelle. Der Inhalt einer Zelle kann formatiert sein. Die Eigenschaft kann nur ausgelesen, aber nicht verändert werden.
- .HorizontalAlignment bestimmt die horizontale Ausrichtung des Inhaltes einer Zelle. Der Wert xlCenterAcrossSelection zentriert einen Wert in einem selektierten Zellbereich. Die Zellen werden aber nicht verbunden. Mit Hilfe der Methode .Merge kann ein Zellbereich verbunden werden.

Eigenschaften für ein Objekt zusammenfassen

```
With Range(Cells(zeile - 1, spalte), Cells(zeile - 1, spalte + 1))
        .HorizontalAlignment = xlCenterAcrossSelection
        .Borders(xlEdgeLeft).LineStyle = xlContinuous
        .Borders(xlEdgeLeft).Weight = xlThin
        .Borders(xlEdgeRight).LineStyle = xlContinuous
        .Borders(xlEdgeRight).Weight = xlThin
        .Borders(xlEdgeTop).LineStyle = xlContinuous
        .Borders(xlEdgeTop).Weight = xlThin
        .Borders(xlEdgeBottom).LineStyle = xlDouble
        .Borders(xlEdgeBottom).Weight = xlThick
End With
```

Erläuterung

- Die Zusammenfassung beginnt mit dem Schlüsselwort With.
- Dem Schlüsselwort folgt der Name eines Objekts. In diesem Beispiel werden die Eigenschaften für den Zellbereich (Range(Cells(zeile - 1, spalte), Cells(zeile - 1, spalte + 1))) verändert. Der Zellbereich wird durch Angabe der linken oberen und der rechten unteren Zelle begrenzt.
- Die Zusammenfassung endet mit End With.
- Zwischen dem Kopf und dem Ende werden alle benötigten Eigenschaften und Methoden zusammengefasst. Der Punkt als Trennzeichen zum Objekt muss gesetzt werden.

ActiveSheet

- ist ein Platzhalter für das momentan aktive Arbeitsblatt
- ist ein Synonym für das am Bildschirm angezeigte Arbeitsblatt.
- nutzt alle Eigenschaften eines Worksheets.

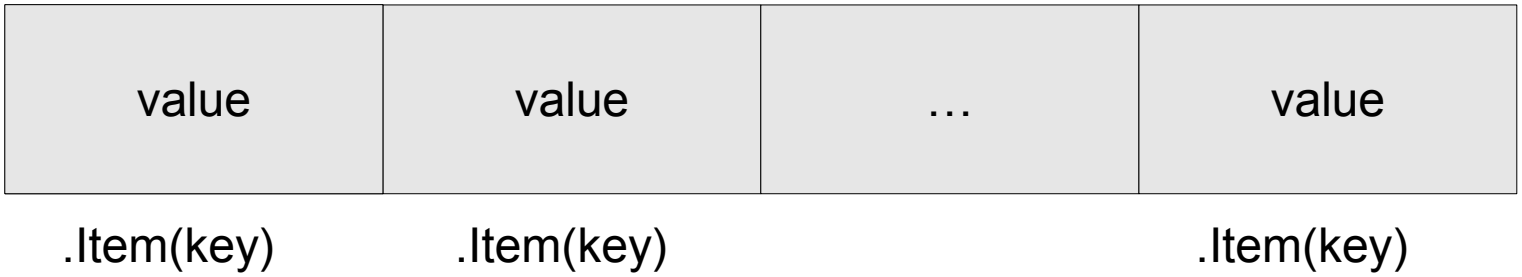
Auflistung Columns

- Auflistung aller Spalten in einem Arbeitsblatt. Die Spalte wird mit Hilfe des Index oder einer Bezeichnung eindeutig definiert.
- Columns(4) bezeichnet die vierte Spalte in einem Arbeitsblatt.
- Columns("B") nutzt die Bezeichnung einer Spalte.
- Columns("M:P") bezieht sich auf die Spalten M, N, O und P.
- Columns(spalte) nutzt eine Variable, um eine Spalte eindeutig zu kennzeichnen.
- Columns.Item(spalte) gibt ein bestimmtes Element (Item) aus der Auflistung zurück.

Eigenschaften

- .AutoFit passt die Spaltengröße dem Inhalt der Spalte an.
- .ColumnWidth gibt einen Wert für die Spaltengröße vor. Als Einheit wird die Breite eines Zeichens genutzt.

Collection



- Auflistung von beliebig vielen Elementen.
- Sammlung von Elementen mit unterschiedlichen Datentypen.
- Gruppe von Elementen zu einem Thema.
- Jedes Element in der Auflistung wird eindeutig über ein Index oder einen benutzerdefinierten Schlüssel identifiziert. Der Schlüssel ist vom Datentyp „String“.

Collection deklarieren

```
Dim feiertag As Collection
```

- Die Variable hat einen benutzerdefinierten Namen. Der Name gibt Auskunft über die Art der Sammlung.
- Die Variable kann nur lokal in der Subroutinen genutzt werden, in der sie definiert ist (Schlüsselwort Dim).
- Das Schlüsselwort As weist der Variablen einen Daten- oder Objekttyp zu. Hier wird eine Variable für ein Objekt Collection erstellt.

Objektvariable initialisieren

```
Dim feiertag As Collection
Set feiertag = New Collection
```

- Mit Hilfe des Gleichheitszeichens wird der Objektvariablen ein Verweis auf ein Objekt zugewiesen.
- Eine Zuweisung an eine Objektvariable beginnt immer mit dem Schlüsselwort Set (Setze). Falls das Schlüsselwort fehlt, wird ein Fehler gemeldet.
- Mit Hilfe des Schlüsselwortes New wird ein neues Objekt vom angegebenen Typ erzeugt. Eine Referenz auf das neue Objekt wird in der Objektvariablen gespeichert.

Definieren und initialisiern

```
Dim feiertag As New Collection
```

- Durch das Schlüsselwort New wird ein neues Objekt erzeugt.
- Ein Verweis auf dieses Objekt wird der Objektvariablen übergeben.

Element der Collection hinzufügen

feiertag.Add "Neujahr", ("01.01." & jahr)

Objektvariable.Add [value], [key], [before], [after]

- Die Methode .Add fügt ein Element der Sammlung hinzu. Standardmäßig wird das neue Element am Ende angehängt.
- Das Element kann vor oder nach einem bestimmten Element eingefügt werden.
- Für das Element kann ein eindeutiger Schlüssel vom Datentyp String festgelegt werden.

Identifizierung der Elemente ...

feiertag.Item([key])

feiertag([index])

- über einen Index: der Index wird in runden Klammern gesetzt. Die Klammern folgen direkt dem Namen der Objektvariablen.
- mit Hilfe des Schlüssels: Der Eigenschaft Item wird ein vorhandener Schlüssel übergeben. Die Angabe des Schlüssels wird durch die runden Klammern begrenzt.
- Falls der Index so groß gewählt wurde oder der Schlüssel nicht vorhanden ist, wird ein Fehler gemeldet.

Laufzeitfehler ...

- sind Fehler, die nach dem Starten des Codes auftreten.
- können einen Programmabsturz verursachen. Das Programm wird unkontrolliert beendet.
- ist zum Beispiel eine „Division durch Null“, ein falscher Umgang mit Datentypen, Endlosschleifen etc..

… abfangen

```
Function getFeiertag(datum As String, feiertag As Collection) As String
        Dim tagName As String
        Dim myFeiertag As Collection

        On Error Resume Next
        Set myFeiertag = New Collection
        Set myFeiertag = feiertag
        tagName = myFeiertag.Item(datum)

         If Err.Number = 5 Then
            getFeiertag = ""
        Else
            getFeiertag = tagName
        End If
End Function
```

Möglichkeiten

- On Error Resume Next. Wenn ein Fehler aufgetreten ist, wird die nächste Zeile ausgeführt. Der Fehler muss in der nächsten Zeile vom Entwickler behoben werden.
- On Error GoTo markeFehler. Falls ein Fehler auftritt, wird zu einer bestimmten Sprungmarke in der Subroutine gesprungen. An der Marke markeFehler: wird der Fehler aufgefangen.

Sprungmarken ...

markeFehler:

- bestehen aus einen benutzerdefinierten Namen und dem Doppelpunkt.
- sind Lesezeichen in einer Prozedur.
- sind eindeutig in einer Prozedur.
- werden bei der Ausführung des Programms nicht beachtet. Das Programm bricht nicht automatisch vor einer Sprungmarke ab. Eine Prozedur kann vor einer Sprungmarke mit Hilfe von Exit Sub und eine Funktion mit Hilfe von Exit Function abgebrochen werden.

Informationen zu Laufzeitfehlern ...

- bietet das vordefinierte Objekt Err.
- Die Eigenschaft Err.Number gibt die Fehlernummer zurück.
- Die Eigenschaft Err.Description enthält den Standard-Fehlertext.
- Die Methode Err.Clear löscht alle aufgetretenen Fehlermeldungen.