

Sub DiagrammErstellen01()

Dim Bereich As Range

Dim s As String

```
Set Bereich = Range("A2:E6")
s = ActiveSheet.Name
Charts.Add
With ActiveChart
    .ChartType = xlColumnClustered
    .SetSourceData _
        Source:=Bereich, _
        PlotBy:=xlRows
    .HasTitle = True
    .ChartTitle.Text = s
End With
End Sub
```

For Each ... Next  
Worksheet\_Change  
ListView, IsError  
xlValidateDate  
Aut Show

Bernd Held

# VBA mit Excel

Das umfassende Handbuch



- Das Standardwerk für Einsteiger und Fortgeschrittene
- Konzepte und Techniken der VBA-Programmierung
- Bewährte Lösungen für die tägliche Praxis



Über 650 geprüfte Makros für den sofortigen Einsatz zum Download



Rheinwerk  
Computing

# Inhalt

Materialien zum Buch .....	25
Vorwort .....	27

## 1 Die Entwicklungsumgebung von Excel 31

---

1.1 Excel auf die Programmierung vorbereiten .....	31
1.1.1 Heruntersetzen der Sicherheitsstufe .....	31
1.1.2 Die Entwicklertools einbinden .....	32
1.2 Die Bestandteile der Entwicklungsumgebung .....	34
1.2.1 Der Projekt-Explorer und das »Eigenschaften«-Fenster .....	34
1.2.2 Der Direktbereich zum Testen verwenden .....	41
1.2.3 Das Überwachungsfenster .....	46
1.2.4 Das »Lokal«-Fenster .....	48
1.2.5 Die Symbolleiste »Bearbeiten« .....	49
1.2.6 Automatische Syntaxprüfung .....	54
1.2.7 Befehle in der nächsten Zeile fortsetzen .....	55
1.2.8 Automatische Anpassung der einzelnen Befehle .....	56
1.2.9 Schnelles Arbeiten über Tastenkombinationen .....	57
1.2.10 Der Objektkatalog .....	59
1.2.11 Der Makrorekorder .....	60
1.2.12 Sonstige wichtige Einstellungen in der Entwicklungsumgebung .....	65
1.2.13 Die Onlinehilfe .....	66

## 2 Datentypen, Variablen und Konstanten 69

---

2.1 Der Einsatz von Variablen .....	69
2.1.1 Regeln für die Syntax von Variablen .....	70
2.1.2 Variablen am Beginn des Makros deklarieren .....	71
2.1.3 Die wichtigsten Variabtentypen .....	72
2.1.4 Variablen Deklarationen erzwingen .....	73
2.1.5 Noch kürzere Deklaration von Variablen .....	74
2.1.6 Die unterschiedlichen Variabtentypen .....	75
2.1.7 Typische Beispiele für den Einsatz von Variablen .....	77
2.1.8 Die Objektvariablen .....	83

<b>2.2 Die Verwendung von Konstanten .....</b>	88
2.2.1 Typische Aufgaben für die Verwendung von Konstanten .....	88
<b>3 Die Sprachelemente von Excel-VBA</b>	103
<hr/>	
<b>3.1 Bedingungen .....</b>	103
<b>3.2 Typische Aufgaben aus der Praxis .....</b>	104
3.2.1 Wert in einer Spalte suchen .....	104
3.2.2 Liegt die aktive Zelle in einem vorgegebenen Bereich? .....	106
3.2.3 Prüfung, ob eine bestimmte Datei verfügbar ist .....	107
3.2.4 Spalteninhalte direkt nach der Eingabe umsortieren .....	108
3.2.5 Spalten mit Wochenenden kennzeichnen .....	110
<b>3.3 Die Kurzform einer Bedingung .....</b>	112
3.3.1 Den Doppelklick auf eine Zelle abfangen .....	113
<b>3.4 Die Anweisung »Select Case« einsetzen .....</b>	114
3.4.1 Zahlenwerte prüfen .....	115
3.4.2 Den Wochentag eines bestimmten Datums auslesen .....	117
3.4.3 Benotungen über einen Autotext durchführen .....	117
<b>3.5 Die »Enum«-Anweisung .....</b>	119
3.5.1 Umsatz klassifizieren mit »Enum« .....	120
<b>3.6 Schleifen erstellen und verstehen .....</b>	121
3.6.1 Die »For ... Next«-Schleife .....	121
3.6.2 Die »For Each ... Next«-Schleife .....	138
3.6.3 Die Schleife »Do Until ... Loop« .....	148
3.6.4 Die Schleife »Do While ... Loop« .....	152
<b>3.7 Sonstige Sprachelemente .....</b>	155
3.7.1 Die Anweisung »With« .....	155
<b>4 Zellen und Bereiche programmieren</b>	161
<hr/>	
<b>4.1 Zahlenformat einstellen und/oder konvertieren .....</b>	161
4.1.1 Zahlenformate einstellen (Datum und Zahl) .....	161
4.1.2 Zahlenformate einstellen (Text) .....	163
4.1.3 Zahlenformate übertragen .....	165
4.1.4 Zellen mit Nullen auffüllen .....	165
4.1.5 Einheitliches Datumsformat einstellen .....	167

4.1.6	Unerwünschte führende und nachgestellte Leerzeichen entfernen .....	169
4.1.7	Korrektur nach fehlerhaftem Import von Daten .....	171
4.1.8	Die Position des Minuszeichens umstellen .....	173
4.1.9	Daten umschlüsseln .....	175
4.1.10	Einen eindeutigen Schlüssel aus mehreren Spalten basteln .....	177
<b>4.2</b>	<b>Zellen, Rahmen und Schriften formatieren .....</b>	<b>179</b>
4.2.1	Schriftart ermitteln .....	179
4.2.2	Schriftart ändern .....	180
4.2.3	Zelleninhalte löschen .....	183
4.2.4	Schriftfarbe teilweise ändern .....	183
4.2.5	Automatisch runden und formatieren .....	185
4.2.6	Zwei Bereiche miteinander vergleichen .....	187
4.2.7	Einen Bereich »mustern« .....	189
4.2.8	Einen Bereich einrahmen .....	191
4.2.9	Einen Bereich umrahmen .....	193
<b>4.3</b>	<b>Die bedingte Formatierung von Excel .....</b>	<b>195</b>
4.3.1	Eine bedingte Formatierung als Standard einstellen .....	195
4.3.2	Duplikate mit dem bedingten Format aufspüren .....	199
4.3.3	Wertgrößen über einen Datenbalken darstellen .....	200
4.3.4	Eine Farbskala definieren und einsetzen .....	201
4.3.5	Daten über Pfeilsymbole bewerten .....	203
4.3.6	Die Top-Werte in einem Bereich hervorheben .....	204
4.3.7	Die einzugebende Textlänge überwachen .....	205
4.3.8	Eine bedingte Formatierung mit Wertgrenzen ausstatten .....	207
4.3.9	Sparklines einsetzen .....	208
<b>4.4</b>	<b>Bereiche und Zellen benennen .....</b>	<b>209</b>
4.4.1	Einen Bereich benennen .....	210
4.4.2	Mehrere Zellen einzeln benennen .....	211
4.4.3	Konstante als Namen vergeben .....	212
4.4.4	Verwendete Namen auslesen .....	213
4.4.5	Versteckte Namen sichtbar machen .....	214
4.4.6	Einen geheimen Namen anlegen .....	215
4.4.7	Einen dynamischen Namen anlegen .....	216
4.4.8	Ein Zellen-Dropdown auf Basis einer benannten Liste anlegen .....	217
4.4.9	Benannte Bereiche löschen .....	219
4.4.10	Benutzerdefinierte Listen erstellen .....	220
<b>4.5</b>	<b>Tabellenfunktionen einsetzen .....</b>	<b>221</b>
4.5.1	Bedingtes Zählen von Werten (ZÄHLENWENN) .....	221

4.5.2	Bedingtes Zählen von Werten bei mehreren Bedingungen (ZÄHLENWENNS) .....	223
4.5.3	Bedingte Summierung von Umsätzen .....	224
4.5.4	Bedingte Summierung von Umsätzen mit mehreren Kriterien .....	226
4.5.5	Den SVERWEIS im Makro einsetzen .....	228
4.5.6	Extremwerte finden und kennzeichnen .....	229
4.5.7	Prüfen, ob ein bestimmter Bereich leer ist .....	233
4.5.8	Einen Bereich mit Zahlenwerten mit vorangestellten Nullen auffüllen .....	234
4.5.9	Die Top-3-Werte in einem Bereich aufspüren .....	235
4.5.10	Automatische Prüfung und Überwachung eines Budgets .....	237
4.5.11	Mussfelder in einer Tabelle überprüfen .....	239
4.5.12	Mittelwert über eine InputBox ermitteln .....	240
4.5.13	Werte oberhalb und unterhalb des Durchschnitts ermitteln und kennzeichnen .....	241
<b>4.6</b>	<b>Matrixformeln in der Programmierung nutzen</b> .....	243
4.6.1	Werte bedingt zählen .....	243
4.6.2	Werte bedingt summieren .....	245
4.6.3	Mittelwert bilden ohne Berücksichtigung von Nullen .....	246
4.6.4	Extremwerte bedingt bilden .....	247
4.6.5	Den am meisten genannten Wert ermitteln .....	247
<b>4.7</b>	<b>Texte und Zahlen manipulieren</b> .....	248
4.7.1	Texte finden und umstellen .....	248
4.7.2	Mehrere Spalten anhand von Trennzeichen splitten .....	251
4.7.3	Daten nach einem Datentransfer bereinigen .....	253
4.7.4	Zeichenfolge(n) aus Zellen entfernen .....	254
4.7.5	Buchstaben aus Zellen entfernen .....	256
4.7.6	Dateinamen aus Pfad extrahieren .....	258
4.7.7	Alle Formelzellen einer Tabelle schützen .....	259
<b>4.8</b>	<b>Gültigkeitsprüfung in Excel</b> .....	260
4.8.1	Gültigkeitskriterien erstellen .....	261
4.8.2	Zellen mit Gültigkeitsfunktion auswählen .....	263
4.8.3	Datumsgrenzen festlegen .....	264
4.8.4	Nur Werkstage für die Eingabe zulassen .....	265
4.8.5	Eine Gültigkeitsliste aus einem Datenfeld befüllen .....	267
4.8.6	Uhrzeiten mit einer Gültigkeitsüberprüfung regeln .....	269
4.8.7	Ein Zellen-Dropdown aus einer Konstanten befüllen .....	270
4.8.8	Einen Standardeintrag für Zellen-Dropdowns setzen .....	272
4.8.9	Gültigkeitskriterien löschen .....	273

<b>4.9</b>	<b>Kommentare in Excel einsetzen</b>	273
4.9.1	Kommentare einfügen	274
4.9.2	Kommentare im Direktfenster auslesen	275
4.9.3	Kommentare formatieren	276
4.9.4	Kommentare aus der aktiven Tabelle löschen	280
4.9.5	Alle Kommentare einer Arbeitsmappe löschen	280
4.9.6	Einen Kommentar einem Namen in einer Arbeitsmappe zuweisen	281
4.9.7	Den Autor von Kommentaren anpassen	282
4.9.8	Kommentarkennzeichnung ein- und ausschalten	283
4.9.9	Kommentare einer Arbeitsmappe in eine Textdatei schreiben	284
4.9.10	Kommentare vor Veränderung schützen	285
4.9.11	Ein Bild in einen Kommentar einfügen	285

## 5 Spalten und Zeilen programmieren

287

---

<b>5.1</b>	<b>Zeilen und Spalten ansprechen, ansteuern und anpassen</b>	287
5.1.1	Mehrere Zeilen auf dem Tabellenblatt markieren	288
5.1.2	Mehrere Spalten auf dem Tabellenblatt markieren	288
5.1.3	Zeilen und Spalten markieren	289
5.1.4	Markierte Spalten zählen	290
5.1.5	Letzte freie Zelle in Spalte identifizieren	291
5.1.6	Anzahl der benutzten Spalten und Zeilen ermitteln	291
5.1.7	Zeilen und Spalten im umliegenden Bereich zählen	293
5.1.8	Zeilenhöhe und Spaltenbreite einstellen	294
<b>5.2</b>	<b>Zeilen einfügen und löschen</b>	297
5.2.1	Zeile einfügen	297
5.2.2	Mehrere Zeilen einfügen	298
5.2.3	Leere Zeilen dynamisch einfügen	298
5.2.4	Wirklich leere Zeilen löschen	301
5.2.5	Doppelte Sätze löschen	301
<b>5.3</b>	<b>Spalten einfügen, löschen und bereinigen</b>	303
5.3.1	Spalte einfügen	303
5.3.2	Mehrere Spalten einfügen	303
5.3.3	Spalte löschen	304
5.3.4	Mehrere Spalten löschen	304

<b>5.4</b>	<b>Zeilen ein- und ausblenden</b>	305
5.4.1	Leere Zeilen ausblenden	305
5.4.2	Alle Zellen einblenden	306
<b>5.5</b>	<b>Spalten ein- und ausblenden</b>	306
5.5.1	Bestimmte Spalten ausblenden	306
<b>5.6</b>	<b>Spalten und Zeilen formatieren</b>	307
<b>5.7</b>	<b>Daten sortieren</b>	309
5.7.1	Daten sortieren mit der klassischen Variante	309
5.7.2	Daten sortieren mit der modernen Methode	310
5.7.3	Daten sortieren nach Farbe der Zellen	313
5.7.4	Daten nach einer Überschrift spaltenweise sortieren	314
<b>5.8</b>	<b>Spalte(n) vergleichen</b>	315
5.8.1	Zelle mit Spalte vergleichen	315
5.8.2	Spalten über eine bedingte Formatierung miteinander vergleichen	316
<b>5.9</b>	<b>Zeilen filtern</b>	318
5.9.1	AutoFilter aktivieren und deaktivieren	318
5.9.2	Filterkriterien setzen	320
5.9.3	Nur Texte filtern	323
5.9.4	Daten filtern, die eine Zeichenfolge enthalten	324
5.9.5	Der Top-10-Filter	325
5.9.6	Filtern nach Zellenfarbe	327
5.9.7	Gefilterte Zeilen entfernen	328
5.9.8	Filterkriterien als Datenfeld übergeben	329
5.9.9	Alle gesetzten Filter sichtbar machen	335
5.9.10	Wie lauten die Filterkriterien?	337
5.9.11	Doppelte Werte mit dem Spezialfilter ermitteln	338
5.9.12	Doppelte Werte mit dem »Dictionary«-Objekt entfernen	340
5.9.13	Daten über einen Kriterienbereich filtern	341
5.9.14	Mehrere Spalten über einen Kriterienbereich filtern	343
5.9.15	Wildcards im Spezialfilter einsetzen	344
5.9.16	Filtern von Umsätzen in einem vorgegebenen Zeitraum	346
5.9.17	Gefilterte Daten transferieren	348
<b>5.10</b>	<b>Zeilen über das Teilergebnis gruppieren</b>	350

---

<b>6 Tabellen und Diagramme programmieren</b>	353
<b>6.1 Tabellen einfügen</b>	353
<b>6.2 Tabellenblätter benennen</b>	354
6.2.1 Eine neue Mappe erstellen, zwölf Monatstabellen anlegen und benennen	355
6.2.2 Eine neue Mappe mit den nächsten 14 Tagen anlegen	356
6.2.3 Tabelle einfügen und gleichzeitig benennen	357
<b>6.3 Tabellen markieren</b>	357
<b>6.4 Tabellenblätter gruppieren</b>	359
6.4.1 Mehrere Tabellen gruppieren	359
6.4.2 Alle Tabellen gruppieren	359
6.4.3 Gruppierte Tabellen übertragen	360
6.4.4 Gruppierte Tabellen ermitteln	360
<b>6.5 Tabellenblätter löschen</b>	361
6.5.1 Eine Tabelle löschen	361
6.5.2 Bestimmte Tabellen aus einer Mappe entfernen	363
6.5.3 Tabellen mit gefärbten Registerlaschen entfernen	364
6.5.4 Leere Tabellen aus Arbeitsmappen entfernen	365
<b>6.6 Tabellenblätter ein- und ausblenden</b>	365
6.6.1 Tabellenblätter sicher ausblenden	366
6.6.2 Tabellen je nach Status ein- oder ausblenden	367
6.6.3 Alle Tabellenblätter anzeigen	367
6.6.4 Alle Tabellen außer der aktiven Tabelle ausblenden	368
<b>6.7 Tabellenblätter schützen</b>	369
6.7.1 Tabellenschutz aufheben	370
6.7.2 Alle Tabellen einer Arbeitsmappe schützen	370
6.7.3 Weitere Schutzfunktionen ab Excel 2002	371
6.7.4 Passwort – Einstellungsdialog mit verschlüsseltem Passwort aufrufen	372
<b>6.8 Tabellen einstellen</b>	373
6.8.1 Registerlaschen ein- und ausblenden	373
6.8.2 Tabellenansicht anpassen	374
6.8.3 Eine einheitliche Zoomeinstellung vornehmen	374
6.8.4 Tabellenblätter sortieren	375
6.8.5 Kopf- und Fußzeilen einrichten	376

6.8.6	Druckbereiche festlegen .....	385
6.8.7	Das Tabellengitternetz ein- und ausschalten .....	386
6.8.8	Zeilen- und Spaltenköpfe ein- und ausblenden .....	386
6.8.9	Cursor einstellen auf Zelle A1 .....	387
<b>6.9</b>	<b>Tabellenblätter drucken und PDF erstellen .....</b>	<b>387</b>
6.9.1	Mehrere Kopien drucken .....	388
6.9.2	Markierte Bereiche drucken .....	388
6.9.3	Mehrere Tabellenblätter drucken .....	389
6.9.4	Tabelle als PDF ablegen .....	389
<b>6.10</b>	<b>Tabelleninhaltsverzeichnis erstellen .....</b>	<b>390</b>
<b>6.11</b>	<b>Intelligente Tabellen .....</b>	<b>392</b>
6.11.1	Tabelle umwandeln .....	393
6.11.2	Tabelle um eine Spalte ergänzen .....	394
6.11.3	Tabelle um eine Zeile ergänzen .....	395
6.11.4	Tabelle filtern .....	396
6.11.5	Tabellen sortieren .....	397
6.11.6	Tabelle um Ergebniszeile erweitern .....	399
6.11.7	Tabelle entfernen .....	399
<b>6.12</b>	<b>Pivot-Tabellen erstellen .....</b>	<b>400</b>
6.12.1	Pivot-Tabellen aktualisieren .....	404
6.12.2	Eine einzelne Pivot-Tabelle aktualisieren .....	404
6.12.3	Mehrere Pivot-Tabellen auf einem Tabellenblatt aktualisieren .....	404
6.12.4	Alle Pivot-Tabellen in einer Arbeitsmappe aktualisieren .....	405
6.12.5	Pivot-Tabellen dynamisch erweitern .....	405
6.12.6	Pivot-Tabellen formatieren .....	406
6.12.7	Slicer einfügen und bedienen .....	407
<b>6.13</b>	<b>Diagramme programmieren .....</b>	<b>409</b>
6.13.1	Umsätze in einem Säulendiagramm darstellen .....	410
6.13.2	Tagesumsätze im Liniendiagramm anzeigen .....	415
6.13.3	Tagesgenaue Formatierung im Punktdiagramm .....	417
6.13.4	Diagramme als Grafiken speichern .....	419
6.13.5	Gewinn und Verlust in einem Säulendiagramm präsentieren .....	421
6.13.6	Linienstärke unabhängig von den Markierungssymbolen formatieren .....	422
6.13.7	Sparklines automatisch erstellen .....	424
<b>6.14</b>	<b>Tabellen blitzschnell vergleichen und Unterschiede dokumentieren .....</b>	<b>427</b>

---

<b>7</b>	<b>Arbeitsmappen und Dateien programmieren</b>	433
<b>7.1</b>	<b>Arbeitsmappen ansprechen</b>	433
<b>7.2</b>	<b>Arbeitsmappen anlegen</b>	434
7.2.1	Eine Arbeitsmappe auf Basis einer Dokumentvorlage erstellen .....	434
7.2.2	Arbeitsmappe mit x Tabellen anlegen .....	435
7.2.3	Mappe mit Wochentabellen anlegen .....	436
<b>7.3</b>	<b>Arbeitsmappen speichern</b>	437
7.3.1	Arbeitsmappe unter aktuellem Tagesdatum speichern .....	438
7.3.2	Alle Tabellen einer Mappe als eigenständige Mappen speichern .....	438
7.3.3	Mappe erstellen und »Speichern unter«-Dialog aufrufen .....	440
7.3.4	Individuellen Speichern-Dialog aufrufen .....	441
7.3.5	Kopie der aktuellen Mappe zur Laufzeit erstellen .....	442
<b>7.4</b>	<b>Arbeitsmappen öffnen</b>	443
7.4.1	Die Argumente der Methode »Open« .....	443
7.4.2	Mehrere Arbeitsmappen öffnen .....	444
7.4.3	Die aktuellste Datei in einem Verzeichnis öffnen .....	445
7.4.4	Regelmäßig das Vorhandensein der Datei prüfen .....	446
7.4.5	Alle verknüpften Mappen automatisch öffnen .....	447
<b>7.5</b>	<b>Arbeitsmappen schließen</b>	448
7.5.1	Arbeitsmappe schließen – Änderungen akzeptieren .....	448
7.5.2	Alle Arbeitsmappen bis auf eine schließen .....	449
<b>7.6</b>	<b>Arbeitsmappe löschen</b>	450
7.6.1	Arbeitsmappe nach Verfallsdatum löschen .....	451
7.6.2	Alle Excel-Mappen in einem Verzeichnis löschen .....	451
7.6.3	Mappe löschen, die älter als 14 Tage ist .....	452
<b>7.7</b>	<b>Arbeitsmappen drucken</b>	453
7.7.1	Nur bestimmte Tabellen drucken .....	453
7.7.2	Alle Mappen eines Verzeichnisses drucken .....	454
7.7.3	Nur sichtbare Blätter ausdrucken .....	455
<b>7.8</b>	<b>Dokumenteigenschaften verarbeiten</b>	456
7.8.1	Dokumenteigenschaftsnamen abfragen .....	457
7.8.2	Letztes Speicherdatum abfragen .....	458
7.8.3	Erstelldatum herausfinden und manipulieren .....	458
7.8.4	Zugriffsdaten einer Arbeitsmappe ermitteln .....	459
7.8.5	Eigene Dokumenteigenschaften verwenden .....	460
<b>7.9</b>	<b>Arbeitsmappen und Verknüpfungen</b>	461
7.9.1	Verknüpfungen in Hyperlinks umwandeln .....	462

7.9.2	Verknüpfungen aus der Arbeitsmappe entfernen .....	463
7.9.3	Verknüpfungen ändern .....	465
7.9.4	Verknüpfungen aktualisieren .....	466
<b>7.10</b>	<b>Arbeitsmappe durchsuchen .....</b>	<b>467</b>
<b>7.11</b>	<b>Arbeitsmappen miteinander vergleichen .....</b>	<b>468</b>
<b>7.12</b>	<b>Arbeitsmappenübersicht erstellen .....</b>	<b>471</b>
<b>7.13</b>	<b>Textdateien importieren .....</b>	<b>473</b>
7.13.1	Eine durch Semikolon getrennte CSV-Datei öffnen .....	473
7.13.2	Eine durch Tabstopps getrennte Textdatei öffnen .....	474
7.13.3	Eine Textdatei mit festen Spaltenbreiten öffnen .....	476
7.13.4	Eine Datenverbindung zu einer Textdatei mit Trennzeichen herstellen .....	478
7.13.5	Eine Datenverbindung zu einer Textdatei mit festen Spaltenbreiten herstellen .....	480
7.13.6	Textdateien zeilenweise einlesen .....	482
<b>7.14</b>	<b>Makros für das Dateimanagement .....</b>	<b>483</b>
7.14.1	Ein Jahresverzeichnis automatisch anlegen .....	484
7.14.2	Eine bestimmte Datei nach Rückfrage löschen .....	485
7.14.3	Einen Ordner archivieren .....	486
7.14.4	Eine bestimmte Datei kopieren .....	488
7.14.5	Ordner anlegen und entfernen .....	489
7.14.6	Verzeichnisstruktur in einer Tabelle anzeigen .....	490

---

## **8 Eigene Funktionen und reguläre Ausdrücke** 495

---

<b>8.1</b>	<b>Benutzerdefinierte Funktionen .....</b>	<b>495</b>
8.1.1	Aktive Arbeitsmappe ermitteln .....	496
8.1.2	Aktives Tabellenblatt ermitteln .....	497
8.1.3	Ist eine Tabelle leer? .....	498
8.1.4	Ist eine Tabelle geschützt? .....	498
8.1.5	Befinden sich Daten in einer bestimmten Spalte? .....	499
8.1.6	Den letzten Wert einer Spalte ermitteln .....	500
8.1.7	Den letzten Wert einer Zeile ermitteln .....	500
8.1.8	Den aktiven Bearbeiter identifizieren .....	501
8.1.9	Funktion zum Umsetzen von Schulnoten .....	501
8.1.10	Rangfolge als Text ausgeben .....	502
8.1.11	Enthält eine bestimmte Zelle ein Gültigkeitskriterium? .....	503
8.1.12	Enthält eine Zelle einen Kommentar? .....	504

8.1.13	Ist eine bestimmte Zelle verbunden? .....	504
8.1.14	Initialen aus Namen erstellen .....	505
8.1.15	Nur Zellen mit Fettdruck addieren .....	506
8.1.16	Mit Uhrzeiten rechnen .....	507
8.1.17	Erweitertes Runden durchführen .....	509
8.1.18	Schnelles Umrechnen von Geschwindigkeiten .....	510
8.1.19	Extremwerte berechnen .....	510
8.1.20	Erste Ziffer in einer Zelle ermitteln .....	512
8.1.21	Buchstaben aus Zellen entfernen .....	514
8.1.22	Anzahl der Ziffern einer Zelle ermitteln .....	515
8.1.23	Römische Zahlen in arabische umwandeln .....	515
8.1.24	Einen Kommentartext in eine Zelle holen .....	517
8.1.25	Angabe eines optionalen Parameters bei einer Funktion .....	518
8.1.26	Leerzeichen in einen String integrieren .....	519
<b>8.2</b>	<b>Jahresbericht mit nur einer Funktion blitzschnell erstellen</b> .....	520
<b>8.3</b>	<b>Modulare Funktionen schreiben</b> .....	526
8.3.1	Dateien in einem Verzeichnis zählen .....	526
8.3.2	Fehlerüberwachung umleiten .....	527
8.3.3	Prüfen, ob eine bestimmte Datei vorhanden ist .....	528
8.3.4	Prüfen, ob eine bestimmte Datei geöffnet ist .....	529
8.3.5	Prüfen, ob eine Datei gerade bearbeitet wird .....	530
8.3.6	Prüfen, ob ein bestimmter Name in der Arbeitsmappe verwendet wird .....	531
8.3.7	Dokumenteigenschaften einer Arbeitsmappe ermitteln .....	531
8.3.8	Den letzten Wert einer Spalte ermitteln .....	534
8.3.9	Grafikelemente in einem definierten Bereich löschen .....	535
8.3.10	Kalenderwoche nach DIN ermitteln .....	537
8.3.11	Unerwünschte Zeichen aus Zellen entfernen .....	537
<b>8.4</b>	<b>Funktionen verfügbar machen</b> .....	539
8.4.1	Speichern der Funktionen in der persönlichen Arbeitsmappe .....	539
8.4.2	Speichern der Funktionen in einem Add-in .....	540
8.4.3	Ein Add-in einbinden .....	541
<b>8.5</b>	<b>Mit regulären Ausdrücken programmieren</b> .....	541
8.5.1	Funktionen für die Arbeit mit regulären Ausdrücken erstellen .....	542
8.5.2	Bestimmte Zeichenfolgen in Texten finden .....	545
8.5.3	Spezielle Zeichen nutzen .....	546
8.5.4	Zeichenfolgen aus Texten extrahieren .....	548
8.5.5	Eine E-Mail-Adresse prüfen .....	550
8.5.6	Konten prüfen .....	552
8.5.7	Zahlen aus Texten extrahieren .....	553

<b>9 Ereignisse programmieren</b>	555
<b>9.1 Ereignisse für die Arbeitsmappe</b>	555
9.1.1 Allgemeine Vorgehensweise beim Erstellen von Arbeitsmappen-Ereignissen	556
9.1.2 Die wichtigsten Ereignisse für die Arbeitsmappe im Überblick	557
9.1.3 Zugriff beim Öffnen der Mappe festhalten (»Workbook_Open«)	559
9.1.4 Das Schließen der Arbeitsmappe bedingt verhindern (»Workbook_BeforeClose«)	564
9.1.5 Letztes Bearbeitungsdatum festhalten (»Workbook_BeforeSave«)	565
9.1.6 Die Lösung für das sparsame Drucken (»Workbook_BeforePrint«)	565
9.1.7 Einfügen von Blättern verhindern (»Workbook_NewSheet«)	566
<b>9.2 Ereignisse für das Tabellenblatt</b>	567
9.2.1 Allgemeine Vorgehensweise bei der Einstellung von Tabellenereignissen	567
9.2.2 Die wichtigsten Ereignisse für Tabellen im Überblick	568
9.2.3 Passworteingabe beim Aktivieren einer Tabelle (»Worksheet_Activate«)	569
9.2.4 Vergleich von zwei Spalten (»Worksheet_Change«)	569
9.2.5 AutoTexte über Kürzel abrufen (»Worksheet_Change«)	571
9.2.6 Symbole nach der Eingabe verändern (»Worksheet_Change«)	573
9.2.7 Die Spaltensumme überwachen (»Worksheet_Change«)	574
9.2.8 Nur einmalige Eingabe zulassen (»Worksheet_Change«)	575
9.2.9 Die Eingabe von Dubletten verhindern (»Worksheet_Change«)	576
9.2.10 Eingabe verhindern (»Worksheet_SelectionChange«)	577
9.2.11 Markierung überwachen (»Worksheet_SelectionChange«)	578
9.2.12 Mausklicks überwachen (»Worksheet_BeforeRightClick«)	579
9.2.13 Die Aktualisierung von Pivot-Tabellen überwachen (»Worksheet_PivotTableUpdate«)	580
<b>9.3 Reaktion auf Tastendruck</b>	580
9.3.1 Texte einfügen	582
9.3.2 Blattsperre ohne Blattschutz erstellen	584
9.3.3 Nur Werte einfügen	585
<b>9.4 Zeitsteuerung in Excel</b>	585
9.4.1 Regelmäßig die Uhrzeit anzeigen	586
9.4.2 Die Zeit läuft ...	587
9.4.3 Zellen blinken lassen	588
9.4.4 Eingaben nach Ablauf von 1 Minute löschen	589

## 10 Die VBE-Programmierung 591

---

<b>10.1</b>	<b>Die VBE-Bibliothek einbinden</b>	592
10.1.1	Die VBE-Bibliothek deaktivieren	593
10.1.2	Weitere Bibliotheken einbinden	594
10.1.3	Objektbibliotheken deaktivieren	595
10.1.4	Informationen zu Objektbibliotheken ausgeben	595
10.1.5	VBE-Editor aufrufen	596
10.1.6	Das Direktfenster aufrufen	597
<b>10.2</b>	<b>Die VBE ein- und ausschalten</b>	597
10.2.1	Neue Module einfügen	598
10.2.2	Einzelne Module löschen	599
<b>10.3</b>	<b>Einzelnes Makro löschen</b>	599
<b>10.4</b>	<b>Alle Makros aus einer Arbeitsmappe entfernen</b>	600
10.4.1	Module mit Makros bestücken	601
10.4.2	Makro zeilenweise in ein Modul übertragen	602
10.4.3	Makros aus einer Textdatei in ein Modul überführen	603
10.4.4	Export von VBA-Modulen in Textdateien	604
<b>10.5</b>	<b>Identifikation von Komponenten</b>	605
<b>10.6</b>	<b>Ein bestimmtes Makro auskommentieren</b>	606
<b>10.7</b>	<b>Das Direktfenster löschen</b>	607
<b>10.8</b>	<b>Den Status des VBA-Projekts abfragen</b>	607
<b>10.9</b>	<b>Makros und Ereignisse dokumentieren</b>	608

## 11 Dialoge, Meldungen und UserForms programmieren 611

---

<b>11.1</b>	<b>»MsgBox«-Meldung</b>	612
11.1.1	Welche Schaltfläche wurde angeklickt?	614
11.1.2	Löschrückfrage	614
<b>11.2</b>	<b>Die »InputBox«-Eingabemaske</b>	615
11.2.1	Einen Suchbegriff über eine InputBox abfragen	616
11.2.2	Abfrage des Spaltenbuchstabens	618
<b>11.3</b>	<b>Integrierte Dialoge einsetzen</b>	619
11.3.1	Den »Öffnen«-Dialog aufrufen	620

11.3.2	Den Dialog »Optionen« aufrufen .....	623
<b>11.4</b>	<b>Eigene UserForms entwerfen .....</b>	623
11.4.1	UserForm einfügen .....	623
11.4.2	UserForm beschriften .....	624
11.4.3	UserForm aufrufen .....	625
11.4.4	Die verfügbaren Steuerelemente .....	625
11.4.5	Steuerelemente einfügen .....	627
11.4.6	Die wichtigsten Eigenschaften .....	628
11.4.7	Ereignisse einstellen .....	631
<b>11.5</b>	<b>Programmierung von Textfeldern .....</b>	631
11.5.1	Passwort über einen Dialog abfragen .....	632
11.5.2	Textfelder leeren .....	636
11.5.3	Textfelder kennzeichnen .....	638
11.5.4	Prüfung auf numerischen Inhalt .....	640
11.5.5	Länge eines Textfeldes prüfen .....	642
11.5.6	Prüfen von und Rechnen mit Textfeldern .....	643
11.5.7	Prüfen und widerrufen .....	648
11.5.8	Eine AutoAusfüllen-Funktion programmieren .....	650
11.5.9	Rechtschreibprüfung vornehmen .....	653
11.5.10	Daten über ein Textfeld suchen .....	655
<b>11.6</b>	<b>Programmierung von Listenfeldern .....</b>	659
11.6.1	Listenfeld mit Tabellen füllen .....	659
11.6.2	Listenfeld mit Monaten füllen .....	662
11.6.3	Mehrspaltiges Listenfeld mit Daten aus Tabelle füllen .....	664
11.6.4	Listenfeld transponiert füllen .....	669
11.6.5	Listenfelder im Duett .....	671
11.6.6	Listenfeld und Textfelder im Zusammenspiel .....	675
<b>11.7</b>	<b>Programmierung von Kombinationsfeldlisten .....</b>	678
11.7.1	Dropdown mit Tagen füllen .....	678
11.7.2	Eindeutige Einträge im Dropdown anzeigen .....	680
11.7.3	Dropdowns synchronisieren .....	684
11.7.4	Dropdown und Listenfeld im Duett .....	687
<b>11.8</b>	<b>Die Programmierung von Optionsschaltflächen .....</b>	690
11.8.1	Mehrwertsteuersatz als Option anwenden .....	690
11.8.2	Optionsfelder und Listenfeld im Zusammenspiel .....	694
<b>11.9</b>	<b>Die Programmierung von Kontrollkästchen .....</b>	696
11.9.1	Kontrollkästchen über eine Tabelle speisen .....	697
11.9.2	Ansichtseinstellungen über Kontrollkästchen vornehmen .....	700

<b>11.10 Die Programmierung von Bildelementen</b> .....	703
11.10.1 Der eigene Bildbetrachter .....	704
<b>11.11 Die Programmierung sonstiger Steuerelemente</b> .....	708
11.11.1 Bilder in MultiPage laden .....	708
11.11.2 Umschaltfläche programmieren .....	710
11.11.3 Drehfeld programmieren .....	714
11.11.4 Die Programmierung des »ListView«-Steuerelements .....	717
11.11.5 Die Programmierung des »TreeView«-Steuerelements .....	723
11.11.6 Die Programmierung des »ProgressBar«-Steuerelements .....	727
<b>11.12 Das Verwaltungstool</b> .....	729
11.12.1 Die hinterlegte Datentabelle .....	730
11.12.2 Die beteiligten Steuerelemente .....	731
11.12.3 Vorbereitende Aufgaben .....	731
11.12.4 Daten suchen .....	733
11.12.5 Mit Klick auf das Listenfeld die Textfelder ausfüllen .....	735
11.12.6 Den Dialog initialisieren .....	736
11.12.7 Datensatz löschen .....	737
11.12.8 Datensatz ändern .....	738
11.12.9 Neue Kundennummer ermitteln .....	739
11.12.10 Datensatz anlegen .....	739
<b>11.13 Ist das ListView-Steuerelement die bessere Listbox?</b> .....	741
11.13.1 Die Befüllung des ListView-Elements .....	742
11.13.2 Das Sortieren von Spalten in einem ListView-Element .....	746
11.13.3 Markierte Elemente in Tabelle zurückschreiben .....	747

---

## 12 Excel im Umfeld von Office programmieren

---

<b>12.1 Excel im Zusammenspiel mit PowerPoint</b> .....	751
12.1.1 Excel-Bereich nach PowerPoint exportieren .....	751
12.1.2 Bereich aus Excel in eine bestehende Präsentation einfügen .....	753
12.1.3 Excel-Bereich verknüpft in eine neue Präsentation integrieren .....	755
12.1.4 PowerPoint-Folie als Objekt in Excel einbinden .....	758
12.1.5 Diagrammobjekte in eine Präsentation exportieren .....	760
<b>12.2 Excel im Zusammenspiel mit Word</b> .....	762
12.2.1 Excel-Bereich in Dokument exportieren .....	763
12.2.2 Excel-Tabelle in ein leeres Dokument überführen .....	765
12.2.3 Markierten Bereich einer Excel-Tabelle in ein Dokument exportieren .....	769

12.2.4	Bereich als Grafik an einer bestimmten Stelle eines Dokuments einfügen .....	771
<b>12.3</b>	<b>Excel im Zusammenspiel mit Outlook .....</b>	<b>773</b>
12.3.1	Kontaktdaten aus Excel nach Outlook exportieren .....	773
12.3.2	Termine aus Excel in den Outlook-Kalender schieben .....	778
12.3.3	Aktive Tabelle aus Excel heraus versenden .....	782
12.3.4	Aktive Tabelle als Anhang aus Excel heraus versenden .....	783
12.3.5	Aktive Arbeitsmappe per E-Mail versenden .....	784
12.3.6	Alle Dokumente aus einem Verzeichnis per E-Mail versenden .....	787
<b>12.4</b>	<b>Excel im Zusammenspiel mit Access .....</b>	<b>790</b>
12.4.1	Toolfrage und Randbedingungen .....	790
12.4.2	Anforderungen an das Tool .....	790
12.4.3	Die Umsetzung der Kernfunktionen .....	791
12.4.4	Befüllung der UserForm mit den wichtigsten Daten .....	792
12.4.5	Suche nach Therapeut über das Kürzel/den Patientennamen .....	794
12.4.6	Suche nach Datum .....	800
12.4.7	Termine erfassen .....	802
12.4.8	Änderung von Terminen .....	804
12.4.9	Termine löschen .....	806
12.4.10	Felder löschen .....	808
<b>12.5</b>	<b>Excel im Zusammenspiel mit dem Internet Explorer .....</b>	<b>809</b>
12.5.1	Eine Internetseite aus Excel aufrufen .....	809
12.5.2	Texte übersetzen mit Google .....	809

---

## **13 Datenfelder, ADO, Dictionaries und Collections programmieren**

---

<b>13.1</b>	<b>Aufgaben mithilfe von ADO und SQL-Statements lösen .....</b>	<b>813</b>
13.1.1	Daten filtern und in einer anderen Tabelle ausgeben .....	814
13.1.2	Umsätze nach Datum verdichten .....	817
13.1.3	Umsätze nach Datum und Warengruppe verdichten .....	820
13.1.4	Daten aus einer Arbeitsmappe beziehen, ohne diese zu öffnen .....	823
13.1.5	Daten aus einer Tabelle löschen .....	825
13.1.6	Top-Werte ermitteln .....	828
13.1.7	Mehrere Tabellen zusammenfassen .....	830
13.1.8	Eine Unikatsliste bilden .....	832
13.1.9	Excel-Daten per ADO verändern .....	835

<b>13.2 Arbeiten mit Arrays .....</b>	838
13.2.1 Aktionen im Arbeitsspeicher ausführen lassen .....	839
13.2.2 Bestimmte Daten aus einer Tabelle löschen .....	842
13.2.3 Daten konvertieren .....	845
<b>13.3 Arbeiten mit dem »Dictionary«-Objekt .....</b>	849
13.3.1 Daten verdichten .....	850
13.3.2 Bedingte Summierung mit mehreren Kriterien .....	853
13.3.3 Eine Unikatsliste erstellen .....	857
13.3.4 Anzahl von Bestellungen ermitteln .....	859
13.3.5 Doppelte Daten in einem Bereich ermitteln .....	864
<b>13.4 Arbeiten mit Collections .....</b>	865
13.4.1 Eindeutige Einträge über eine Collection bilden .....	866
13.4.2 Eine Collection aus einer Tabelle befüllen .....	869
<b>13.5 Intelligente Arrays mit Zusatzfunktionen .....</b>	871
13.5.1 Stack füllen und prüfen .....	871
13.5.2 Eine SortedList füllen und übergeben .....	873

---

## **14 Die Programmierung der Excel-Oberfläche**

---

<b>14.1 Die Programmierung von Kontextmenüs .....</b>	877
14.1.1 Kontextmenüs deaktivieren .....	878
14.1.2 Das Zellenkontextmenü erweitern .....	879
14.1.3 Kontextmenü aufbauen (dreistufig) .....	881
14.1.4 Kontextmenü zurücksetzen .....	882
<b>14.2 Die Menübandprogrammierung .....</b>	883
14.2.1 Der Custom UI Editor .....	883
14.2.2 Weitere wichtige Quellen und Hilfen .....	885
14.2.3 Menüband mit Schaltflächen erstellen .....	887
14.2.4 Menüband mit »ComboBox« erstellen .....	888
14.2.5 Menüband mit bereits verfügbaren Funktionen bestücken .....	890
14.2.6 Den Backstage-Bereich programmieren .....	892
14.2.7 Eine Galerie mit Fotos erstellen .....	893

<b>15 Fehlerbehandlung, Tuning und Schutz von VBA-Projekten sowie Support durch KI</b>	895
<b>15.1 Kleinere Fehler beheben</b>	895
15.1.1 Stimmt die Syntax? .....	896
15.1.2 Ist die Variablendefinition gegeben? .....	896
15.1.3 Objekt vorhanden? .....	896
15.1.4 Methode, Eigenschaft verfügbar? .....	897
<b>15.2 Schwerwiegendere Fehler</b>	897
15.2.1 Fehler im Vorfeld erkennen und reagieren .....	897
15.2.2 Fehler ignorieren .....	898
15.2.3 Fehlerursache ermitteln .....	898
15.2.4 Die Funktion »IsError« .....	899
<b>15.3 Das Add-in MZ-Tools</b>	900
15.3.1 Zeilenummern automatisch einfügen .....	901
15.3.2 Eine Fehlerbehandlung mit den MZ-Tools hinzufügen .....	902
<b>15.4 Laufzeiten verkürzen</b>	903
15.4.1 Variablen und Konstanten einsetzen .....	903
15.4.2 Berechnung und Bildschirmaktualisierung ausschalten .....	904
15.4.3 Integrierte Tabellenfunktionen anwenden .....	905
<b>15.5 VBA-Projekte schützen</b>	905
<b>15.6 KI und Excel-VBA</b>	906
15.6.1 Codevorschläge und Generierung von vollständigem Code .....	906
15.6.2 Fehlererkennung und Debugging .....	908
15.6.3 Codeübersetzung .....	910
15.6.4 Codeanalyse und Optimierung, Fehlerbehandlung integrieren, Codehygiene .....	911
15.6.5 Dokumentation und Kommentierung .....	912
15.6.6 Codeverständnis .....	913
15.6.7 Recherche und Weiterbildung .....	913
<b>15.7 Mein Fazit</b>	914
<b>16 Typische Verarbeitungsaufgaben aus der Praxis</b>	915
<b>16.1 Daten übertragen</b>	915
<b>16.2 Daten im Batch verarbeiten</b>	918

<b>16.3</b>	<b>Daten verteilen</b>	920
16.3.1	Die Tabellen entfernen	921
16.3.2	Die Verteilung der Zeilen auf die Tabellen	922
16.3.3	Die Plausibilität prüfen	925
16.3.4	Der Export der Tabellen	927
<b>16.4</b>	<b>Berichtsfilterseiten erstellen</b>	928
<b>16.5</b>	<b>Daten löschen</b>	931
16.5.1	Daten entfernen – Variante 1	931
16.5.2	Daten entfernen – Variante 2	932
16.5.3	Daten entfernen – Variante 3	933
<b>16.6</b>	<b>Daten kennzeichnen</b>	934
16.6.1	Doppelte Daten kennzeichnen (der Standard)	935
16.6.2	Doppelte Daten kennzeichnen (die Erweiterung)	936
16.6.3	Top-10-Werte aus einem Bereich ermitteln	938
<b>16.7</b>	<b>Diagramme automatisch formatieren</b>	941
<b>16.8</b>	<b>Daten mithilfe von Wildcards finden</b>	945
<b>16.9</b>	<b>Zwei identische Bereiche miteinander vergleichen</b>	946
<b>16.10</b>	<b>Suche nach einem Begriff unter Berücksichtigung der Formatierung</b>	948
<b>16.11</b>	<b>Automatische Sicherung von E-Mails in einer Access-Datenbank</b>	949
<b>16.12</b>	<b>Einen Durchschnitt aus den Top-5-Werten ermitteln</b>	952
<b>16.13</b>	<b>Arbeitsmappen auf Knopfdruck automatisch erstellen</b>	954
<b>16.14</b>	<b>Alle Formeln einer Tabelle schützen und verstecken</b>	955
<b>16.15</b>	<b>Eine Unikatsliste über den Einsatz von SQL erstellen</b>	956
<b>16.16</b>	<b>Erstellung eines Kalenders mit VBA – Schritt für Schritt</b>	959
<b>16.17</b>	<b>Kriterien für eine Mehrfachfilterung aus Zellen beziehen</b>	961
<b>16.18</b>	<b>Bestimmte Zeichenfolge in einem Bereich entfernen</b>	962
<b>16.19</b>	<b>Automatisch eine Kopie der Mappe erstellen</b>	963
<b>16.20</b>	<b>Einen Excel-Bereich als Objekt nach PowerPoint übertragen</b>	964
<b>16.21</b>	<b>Einen formatierten Text in eine PowerPoint-Folie übertragen</b>	966
<b>16.22</b>	<b>Top-Werte ermitteln und kennzeichnen</b>	968
<b>16.23</b>	<b>Daten aus einer geschlossenen Mappe ziehen</b>	970
<b>16.24</b>	<b>Ein PDF in Excel erstellen und direkt versenden</b>	973
<b>16.25</b>	<b>Daten verdichten</b>	974
16.25.1	Daten verdichten mithilfe von SUMMEWENNNS	976

16.25.2 Daten mithilfe einer SQL-Anweisung verdichten .....	977
16.25.3 Daten verdichten mithilfe des Datenfilters .....	979
<b>16.26 Daten in Mappen aktualisieren, ohne diese zu öffnen .....</b>	<b>980</b>
<b>16.27 Eine Suchfunktion für eine ListBox erstellen .....</b>	<b>983</b>
<b>16.28 Mehrere Bilder dynamisch in eine Tabelle einfügen (Bildergalerie) .....</b>	<b>986</b>
<b>16.29 Bedingte Summierung direkt im Arbeitsspeicher vornehmen .....</b>	<b>989</b>
<b>16.30 Über zwei Arrays Jahresumsätze pro Monat auswerten .....</b>	<b>994</b>
<b>16.31 Bedingte Summierung mit mehreren Kriterien im Speicher durchführen .....</b>	<b>997</b>
<b>16.32 Pfeilsymbol oberhalb einer bestimmten Säule in einem Diagramm einfügen .....</b>	<b>1001</b>
<b>16.33 Excel erweitern – doppelte Werte finden und unterschiedlich kennzeichnen</b>	<b>1004</b>
<b>16.34 Bestimmte Zeilen übertragen – Kriterien dynamisch zusammenstellen .....</b>	<b>1007</b>
<b>16.35 Der Vollautomat – alle Dateien eines Verzeichnisses schnell verarbeiten .....</b>	<b>1010</b>
<b>16.36 Eine Mehrfachsuche in einer Tabelle mit einem Array beschleunigen .....</b>	<b>1013</b>
<b>Index .....</b>	<b>1017</b>

# Kapitel 4

## Zellen und Bereiche programmieren

*In diesem Kapitel dreht sich alles um Zellen, die durch das Objekt »Range« angesprochen werden. Das »Range«-Objekt kann aus einer einzigen Zelle oder aus mehreren Zellen bzw. aus einem Zellbereich oder mehreren Zellbereichen bestehen.*

Das wichtigste Objekt in der Excel-Programmierung ist meiner Ansicht nach das Objekt Range. Wenn Sie mit diesem Objekt umgehen können, dann ist schon viel gewonnen.

In diesem Kapitel behandle ich die folgenden Themen:

- ▶ Zellen formatieren und konvertieren
- ▶ Zellen benennen und mit Namen arbeiten
- ▶ Formeln und Tabellenfunktionen einsetzen
- ▶ Gültigkeitsprüfungen durchführen
- ▶ Kommentare in Zellen verarbeiten

### Die Beispiele aus diesem Kapitel zum Download

Sie finden alle Beispiele zu diesem Kapitel in der Datei *Zellen.xlsxm*, die Sie auf der Verlagswebsite unter [www.rheinwerk-verlag.de/6028](http://www.rheinwerk-verlag.de/6028) herunterladen können.



### 4.1 Zahlenformat einstellen und/oder konvertieren

Beginnen wir, zunächst einige Zahlenformate für Zellen einzustellen.

In vielen Fällen reicht jedoch eine einfache Umformatierung von Daten leider nicht aus. Oft muss man Daten mithilfe von Konvertierungsfunktionen nachbereiten.

#### 4.1.1 Zahlenformate einstellen (Datum und Zahl)

Die Zahlenformate finden Sie in der Oberfläche von Excel, wenn Sie beispielsweise einen Bereich markieren und die Tastenkombination **Strg** + **1** drücken. Daraufhin öffnet sich der Dialog ZELLEN FORMATIEREN, in dem Sie sämtliche Zahlenformate einstellen können.

Alle diese Formate können Sie aber auch automatisiert über Makros einstellen. Die dazu notwendige Eigenschaft heißt `NumberFormat`. Allerdings müssen Formate in der Entwicklungsumgebung so eingestellt werden, wie es im Land des Gründers von Microsoft üblich ist. Denken Sie also daran, dass das Tausendertrennzeichen in Amerika das Komma ist und das Dezimaltrennzeichen der Punkt. Auch bei den Datumsformatierungen wird mit Buchstabenkürzeln gearbeitet. So gelten die Kürzel aus Tabelle 4.1 bei den Datumsformaten, wenn wir von dem Datum 27.12.2021 ausgehen.

Englisch	Deutsch	Bedeutung	Ergebnis auf Deutsch
DD	TT	Tagesangabe zweistellig	27
DDD	TTT	Tagesangabe dreistellig	Mo
DDDD	DDDD	Tagesangabe vierstellig	Montag
MM	MM	Monatsangabe zweistellig	12
MMM	MMM	Monatsangabe dreistellig	Dez
MMMM	MMMM	Monatsangabe vierstellig	Dezember
YY	JJ	Jahresangabe zweistellig	21
YYYY	JJJJ	Jahresangabe vierstellig	2021

**Tabelle 4.1** Die wichtigsten Formatkürzel für das Datum

Die Konvertierung der in Englisch einzugebenden Formatcodes erfolgt automatisch. Sehen Sie sich dazu das Makro aus Listing 4.1 an. Dieses Makro versieht Spalte A von TABELLE1 mit dem Datumsformat DD.MM.YYYY.

```
Sub ZahlenFormatEinstellen()
```

```
With Tabelle1
```

```
'Datumsformat einstellen
.Range("A:A").NumberFormat = "DD.MM.YYYY"

'Zahlenformat: (Tausenderpunkt, Dezimalkomma mit zwei Nachkommastellen)
.Range("B:B").NumberFormat = "#,##,.00"
```

```
End With
```

```
End Sub
```

**Listing 4.1** Zahlenformat richtig einstellen

Mithilfe der Eigenschaft NumberFormat geben Sie das gewünschte Zahlenformat gleich für die komplette Spalte A an.

Für Spalte B wählen wir ein Zahlenformat, das den Tausenderpunkt und das Dezimalkomma enthält. Auch in diesem Fall müssen Sie die Formatierung so verwenden, wie sie in den USA üblich ist.

	A	B	C	D	E	F	G
1	Datum	Umsatz					
2	01.10.2024	9.273,00					
3	02.10.2024	6.693,00					
4	03.10.2024	4.592,00					
5	04.10.2024	3.598,00					
6	05.10.2024	3.560,00					
7	06.10.2024	8.512,00					
8	07.10.2024	1.733,00					
9	08.10.2024	8.907,00					
10	09.10.2024	5.967,00					
11	10.10.2024	4.233,00					
12	11.10.2024	7.603,00					
13	12.10.2024	7.653,00					
14	13.10.2024	7.946,00					
15	14.10.2024	7.578,00					
16	15.10.2024	9.807,00					
17	16.10.2024	2.346,00					
18	17.10.2024	9.392,00					
19	18.10.2024	1.920,00					
20	19.10.2024	6.980,00					

Abbildung 4.1 Die Formatierung wurde vorgenommen.

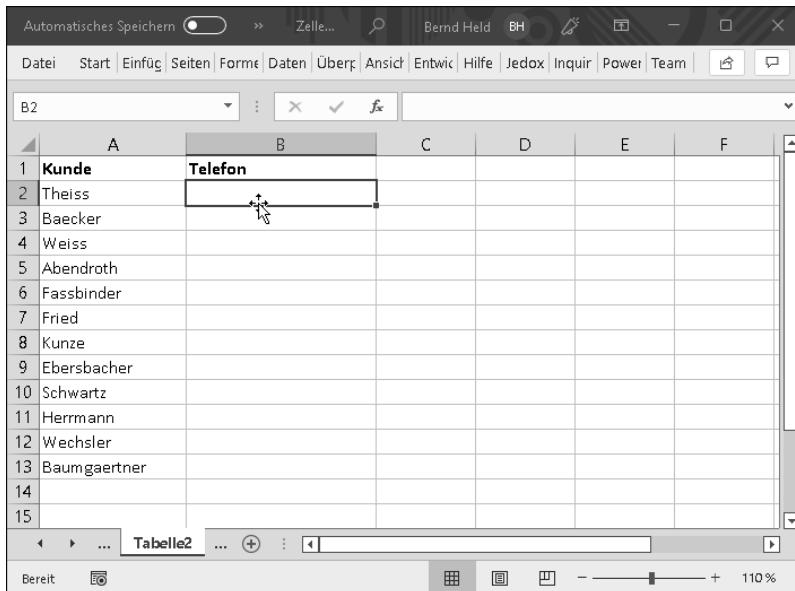
### Wichtiger Hinweis

Verwenden Sie nicht die Eigenschaft NumberFormatLocal. Diese Eigenschaft erlaubt es Ihnen zwar, die landesüblichen Zahlenformate zu verwenden, aber wenn Sie dann die Arbeitsmappe über den Teich schicken, klappt das nicht mehr.

#### 4.1.2 Zahlenformate einstellen (Text)

Etwas beschwerlich ist es auch, wenn Sie in Excel beispielsweise Telefonnummern mit führenden Nullen erfassen. Excel schluckt diese einfach. Sie wissen dann im Zweifelsfall nicht, wie viele Nullen Sie beim Telefonieren vorwählen müssen. Speziell in diesem Fall und in ähnlich gelagerten Fällen ist es sinnvoll, den entsprechenden

Bereich vorher über das Textformat zu formatieren. In diesem Fall bleiben die führenden Nullen erhalten.



	A	B	C	D	E	F
1	Kunde	Telefon				
2	Theiss					
3	Baecker					
4	Weiss					
5	Abendroth					
6	Fassbinder					
7	Fried					
8	Kunze					
9	Ebersbacher					
10	Schwartz					
11	Herrmann					
12	Wechsler					
13	Baumgaertner					
14						
15						

Abbildung 4.2 In Spalte B sollen Telefonnummern eingetragen werden.

Auch diese Einstellung wird über das Makro aus Listing 4.2 vorgenommen, bevor überhaupt Daten erfasst oder importiert werden. Denn sind die Nullen einmal weg, dann dauerhaft. Stellen Sie daher zuerst das Zahlenformat ein, und nehmen Sie danach die Befüllung vor.

```
Sub TextFormatEinstellen()
    Dim lngZeileMax As Long

    With Tabelle2
        lngZeileMax = .Cells(.Rows.Count, 1).End(xlUp).Row
        .Range("B2:B" & lngZeileMax).NumberFormat = "@"
    End With

End Sub
```

Listing 4.2 Einen verwendeten Bereich mit dem Zahlenformat »Text« formatieren

Das Formatkürzel @ steht für das Zahlenformat Text. Dieses wird der Eigenschaft NumberFormat übergeben und dem vorher ermittelten Bereich zugewiesen. Alle Eingaben nach diesem Makrolauf werden als Text vorgenommen, d. h., die führenden Nullen werden von Excel nicht einkassiert.

### 4.1.3 Zahlenformate übertragen

Aufsetzend auf der gerade vorgestellten Aufgabe soll die Formatierung des verwen-deten Bereichs in Spalte B der Tabelle auf den gleich großen Bereich von Spalte D übertragen werden. Das Makro zur Lösung dieser Aufgabenstellung sehen Sie in Lis-ting 4.3.

```
Sub ZahlenformateÜbertragen()
    Dim rngQuelle As Range
    Dim rngZiel As Range
    Dim lngZeileMax As Long

    With Tabelle2

        lngZeileMax = .Cells(.Rows.Count, 1).End(xlUp).Row

        Set rngQuelle = .Range("B2:B" & lngZeileMax)
        Set rngZiel = .Range("D2:D" & lngZeileMax)

        rngZiel.NumberFormat = rngQuelle.NumberFormat

    End With

End Sub
```

#### **Listing 4.3** Eingestelltes Zahlenformat aus einem Bereich auf einen anderen übertragen

Deklarieren Sie zu Beginn des Makros zwei Objektvariablen vom Typ Range, und geben Sie über die Anweisung Set bekannt, wo diese Bereiche in der Tabelle liegen. Danach übertragen Sie mithilfe der Eigenschaft NumberFormat die eingestellte Formatierung des Bereichs rngQuelle in den Bereich rngZiel.

### 4.1.4 Zellen mit Nullen auffüllen

Microsoft Excel wird oft als Schnittstelle zwischen mehreren Systemen verwendet. Dazu kann es einmal notwendig sein, Zellen mit führenden Nullen aufzufüllen, damit die Daten weiterverarbeitet werden können.

Bei der folgenden Aufgabenstellung liegen in Spalte A von TABELLE3 einige Zahlen-werte vor, die auf die Zeichenlänge 5 gebracht werden müssen. Sind zu wenige Ziffern erfasst, muss das Makro aus Listing 4.4 diese von links mit sichtbaren Nullen auf-füllen.

```
Sub AuffüllenVonNullen()
    Dim rngZelle As Range

    With Tabelle3

        For Each rngZelle In .Range("A2:A10")

            rngZelle.Offset(0, 1).NumberFormat = "@"

            If Len(rngZelle.Value) < 5 Then
                rngZelle.Offset(0, 1).Value = _
                    Right("00000" & rngZelle.Value, 5)
            Else
                rngZelle.Offset(0, 1).Value = rngZelle.Value
            End If

        Next rngZelle

        .Range("B:B").HorizontalAlignment = xlRight

    End With

End Sub
```

#### **Listing 4.4** Eine Zahlenreihe mit führenden Nullen auffüllen

In einer Schleife des Typs `For Each ... Next` werden alle Zellen im angegebenen Bereich nacheinander verarbeitet. Innerhalb der Schleife wird für den um eine Spalte nach rechts versetzten Bereich das Textzahlenformat über die Eigenschaft `NumberFormat` festgelegt.

Die Verschiebung erfolgt über die Eigenschaft `Offset`. Diese Eigenschaft hat zwei Argumente:

- Im ersten Argument geben Sie die gewünschte Zeilenverschiebung an. Im vorliegenden Beispiel jedoch erfolgt hier keine Verschiebung, daher geben wir die Zeilenverschiebung mit 0 an.
- Im zweiten Argument wird die Spaltenverschiebung angegeben. Der Wert 1 bedeutet, dass die Spalte um eine Spalte weiter nach rechts verschoben wird.

In einer `If`-Bedingung fragen Sie über die Funktion `Len` die Anzahl der erfassten Zeichen ab. Ist die ermittelte Anzahl kleiner als 5 Zeichen, dann muss aufgefüllt werden.

Dazu setzen Sie die Funktion Right ein und verknüpfen den Inhalt mit der Funktion. Stimmt die Länge, dann übertragen Sie den Inhalt 1 : 1 in die Nebenzelle.

Über die Eigenschaft HorizontalAlignment richten Sie den Text rechtsbündig in der Zelle aus, indem Sie dieser Eigenschaft die Konstante xlRight zuweisen.

	A	B	C	D	E	F	G	H
1	Nummer	Nummer aufgefüllt						
2	1	00001						
3	13	00013						
4	154	00154						
5	2456	02456						
6	3245	03245						
7	23	00023						
8	4	00004						
9	17	00017						
10	12345	12345						
11								
12								

Abbildung 4.3 Die zu kurzen Werte wurden um vorangestellte Nullen komplettiert.

#### 4.1.5 Einheitliches Datumsformat einstellen

Es ist wie verhext – alle machen, was sie wollen. In Abbildung 4.4 sehen Sie, dass TABELLE4 Datumsangaben in den unterschiedlichsten Formen enthält. Mehrere Datumseingaben werden gar nicht erst von Excel erkannt. Wer soll so eine Liste noch auswerten können? Hilft nichts, da muss ein Makro drüber und die Datumsformate vereinheitlichen. Dabei sollen die korrekten Datumsformate in Spalte C geschrieben werden.

Starten Sie das Makro aus Listing 4.5, um die Datumsformate zu vereinheitlichen.

```
Sub EinheitlichesDatumsformatEinstellen()
    Dim lngZeile As Long
    Dim lngZeileMax As Long

    With Tabelle4
        .Range("C1").Value = "Datum korrigiert"
        .Columns(3).NumberFormat = "DD.MM.YYYY"

        lngZeileMax = .Cells(.Rows.Count, 1).End(xlUp).Row
    End With
End Sub
```

```
For lngZeile = 2 To lngZeileMax

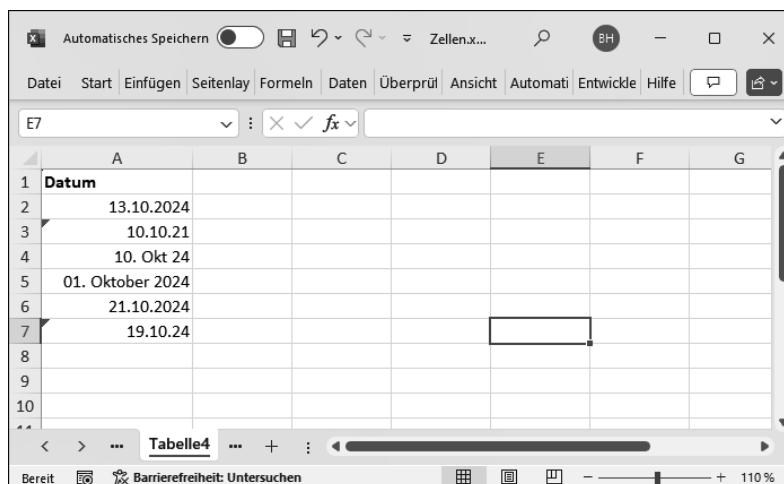
    If IsDate(.Cells(lngZeile, 1).Value) = True Then
        .Cells(lngZeile, 3).Value = CDate(.Cells(lngZeile, 1).Value)
    Else
        .Cells(lngZeile, 1).Interior.ColorIndex = 3
    End If

    Next lngZeile

End With

End Sub
```

**Listing 4.5** Datumsformate vereinheitlichen und in von Excel interpretierbare Datumsangaben wandeln



	A	B	C	D	E	F	G
1	Datum						
2	13.10.2024						
3	10.10.21						
4	10. Okt 24						
5	01. Oktober 2024						
6	21.10.2024						
7	19.10.24						
8							
9							
10							

**Abbildung 4.4** Die Datumsangaben müssen vereinheitlicht und vor allem von Excel erkannt werden.

Die eigentliche Umwandlung und Vereinheitlichung findet direkt in der `For ... Next`-Schleife statt. Dort prüfen Sie sicherheitshalber über die Funktion `IsDate`, ob Excel überhaupt eine Chance hat, ein datumsähnliches Format zu erkennen. Es kann nur dann die Umwandlungsfunktion `CDate` zum Einsatz kommen, wenn sichergestellt ist, dass es sich um ein brauchbares Datum handelt. Sollte in einer Zelle ein ungültiges Datum (z. B. »30.02.15«) stehen, dann wird der `Else`-Zweig der Bedingung durchlaufen. In diesem Fall wird die entsprechende Zelle rot angemalt.

Abbildung 4.5 Alle Datumsangaben wurden konvertiert und einheitlich formatiert.

#### 4.1.6 Unerwünschte führende und nachgestellte Leerzeichen entfernen

Etwas lästig – um nicht zu sagen, etwas hinterlistig – sind auch führende oder nachgestellte Leerzeichen in Zellen. Diese unnötigen Leerzeichen manuell zu entfernen ist eine Bestrafung, erst recht dann, wenn Sie diese Arbeit mit mehreren tausend Zeilen durchführen dürfen. Auch hier wird ein Makro eingesetzt, um alle Geschwindigkeitsrekorde zu brechen. Sehen Sie sich aber zuerst die Ausgangssituation aus Abbildung 4.6 an.

Mitarbeiter Schulz müsste eigentlich einen Gesamtumsatz von 3.850 € haben. Statt dessen liefert uns die Tabellenfunktion SUMMEWENN ein Ergebnis von 1.500 €. Das Problem hierbei sind vorangestellte und nachgestellte Leerzeichen im Datenbestand. Dafür kann die Tabellenfunktion nichts! Für die Funktion handelt es sich um verschiedene Mitarbeiter.

Sie können jetzt aber auch nicht hergehen und die Zellen einzeln kontrollieren, indem Sie die Zelle markieren, die Taste **F2** drücken und oben in der Bearbeitungsleiste schauen, ob da vorangestellte oder nachgestellte leere Zeichen beim Namen stehen. Das ist ein Ding der Unmöglichkeit. Auch wenn dieser Effekt nur in ein paar Zellen von Tausenden auftritt, es bleibt doch ein mulmiges Gefühl, weil das Ergebnis daraus einfach nicht passt. Daher können Sie zur Sicherheit das Makro aus Listing 4.6 über den Datenbestand laufen lassen. Danach können Sie sicher sein, dass alle »Störfriede« entfernt wurden.

```
Sub LeerzeichenEntfernen()
    'unerwünschte Leerzeichen am linken/rechten Rand entfernen
    'Trim entfernt Leerzeichen am rechten und linken Rand
```

```

'LTrim entfernt Leerzeichen am linken Rand
'RTrim entfernt Leerzeichen am rechten Rand
Dim lngZeile As Long
Dim lngZeileMax As Long

With Tabelle5
    lngZeileMax = .UsedRange.Rows.Count

    For lngZeile = 2 To lngZeileMax

        .Range("A" & lngZeile).Value = Trim(.Range("A" & lngZeile).Value)

    Next lngZeile

End With

End Sub

```

**Listing 4.6** Unerwünschte Leerzeichen aus Zellen automatisch entfernen

	A	B	C	D	E	F	G
1	Mitarbeiter	Umsatz		Mitarbeiter	Umsatz gesamt		
2	Schulz	1.500 €		Schulz	1.500 €		
3	Schulz	1.200 €					
4	Schulz	500 €					
5	Schmitt	1.850 €					
6	Meier	2.500 €					
7	Meier	750 €					
8	Schulz	650 €					
9							
10							

**Abbildung 4.6** Dass hier etwas nicht stimmt, ist offensichtlich.

Mithilfe der Funktion `Trim` entfernen Sie führende und nachgestellte Leerzeichen aus Zellen. Diese Funktion wird in der Schleife eingesetzt. Dabei werden die alten Zelleninhalte mit den »getrimmten« Zellenwerten überschrieben.

	A	B	C	D	E	F	G
1	Mitarbeiter	Umsatz		Mitarbeiter	Umsatz gesamt		
2	Schulz	1.500 €		Schulz	3.850 €		
3	Schulz	1.200 €					
4	Schulz	500 €					
5	Schmitt	1.850 €					
6	Meier	2.500 €					
7	Meier	750 €					
8	Schulz	650 €					
9							
10							

Abbildung 4.7 Alle Leerzeichen sind raus – Excel rechnet richtig.

#### 4.1.7 Korrektur nach fehlerhaftem Import von Daten

Vielleicht kennen Sie dieses nicht gerade selten auftretende Phänomen? Sie importieren Daten aus einem Fremdsystem nach Excel und stellen nachher mehr oder weniger zufällig fest, dass Excel manche Zahlen wohl oder übel als Text interpretiert. Dieses »Fehlverhalten« von Excel ist nicht unbedingt gleich auf den ersten Blick erkennbar. Erst wenn Sie testweise einige Zellen markieren und in der Statusleiste die Summe mit den markierten Zahlen vergleichen, sehen Sie, dass etwas nicht stimmt. Schauen Sie sich dazu einmal Abbildung 4.8 an.

	A	B	C	D	E	F	G
1	Werte						
2		13,56					
3		1567,99					
4		34					
5		215,18					
6		1150,80					
7		12,35					
8		99,99					
9							
10							

Abbildung 4.8 Nanu? Da wird ja gar nichts mehr berechnet!

Ja, genau. Excel erkennt diese importierten Daten nicht. Was nun? Schreiben Sie ein Makro wie das in Listing 4.7, das Excel zwingt, die Zahlen richtig zu erkennen.

```
Sub ZahlenwerteRichtigErkennen()
    Dim lngZeile As Long
    Dim lngZeileMax As Long

    With Tabelle7
        .Range("C1").Value = "korrigierte Werte"
        lngZeileMax = .Range("A" & .Rows.Count).End(xlUp).Row

        For lngZeile = 2 To lngZeileMax

            If IsNumeric(.Range("A" & lngZeile).Value) = True Then
                .Range("C" & lngZeile).Value = _
                    .Range("A" & lngZeile).Value * 1
            Else
                .Range("C" & lngZeile).Value = 0
            End If

        Next lngZeile
    End With
End Sub
```

#### Listing 4.7 Zahlenwerte in Excel nachberechnen

Deklarieren Sie zwei Variablen vom Typ Long. Danach legen Sie über die Anweisung With fest, auf welcher Tabelle die Verarbeitung erfolgen soll. Nun ermitteln Sie, wie viele Zeilen in Spalte A der Tabelle belegt sind. Dazu arbeiten Sie mit der Eigenschaft End, über die Sie ausgehend von der letzten möglichen Zelle einer Tabelle zur letzten belegten Zelle nach oben gehen und über die Eigenschaft Row die dazugehörige Zeilennummer auslesen. Sie haben jetzt die Zeile ermittelt, bis zu der die Verarbeitung durchgeführt werden soll. In einer Schleife arbeiten Sie sich also beginnend von der zweiten Zeile bis zur letzten belegten Zeile zeilenweise durch die Tabelle. Innerhalb der Schleife schreiben Sie das Ergebnis der Konvertierung in Spalte C. Dabei prüfen Sie zunächst über die Funktion IsNumeric, ob der jeweilige Zelleninhalt aus Spalte A numerisch ist. Wenn nicht, dann schreiben Sie in Spalte H den Wert 0. Im anderen Fall zwingen Sie Excel sicherheitshalber, die ermittelte Zahl wirklich als Zahl zu erkennen, indem Sie sie mit dem Wert 1 multiplizieren. Dadurch führt Excel eine Neuberechnung der Zelle durch, und der Wert wird richtig erkannt.

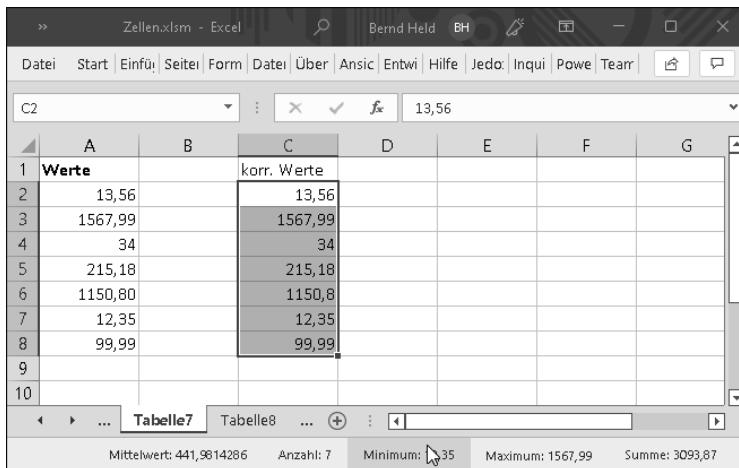


Abbildung 4.9 Jetzt erkennt Excel die Zahlenwerte.

#### 4.1.8 Die Position des Minuszeichens umstellen

Manche Warenwirtschaftssysteme haben die Angewohnheit, negative Zahlen mit einem Minuszeichen am rechten Rand auszustatten. Microsoft Excel erwartet die Minuszeichen jedoch auf der linken Seite und kann mit rechtslastigen Minuszeichen leider nichts anfangen. Haben Sie einmal versucht, solche Zellen manuell zu bereinigen? Diese Nackenschmerzen auslösende Arbeit muss nun wirklich nicht sein – wie schnell verrutschen Sie einmal um eine Stelle oder löschen aus Versehen ein oder auch mehrere Zeichen!

Sehen Sie sich zunächst einmal die Ausgangssituation in Abbildung 4.10 etwas genauer an.

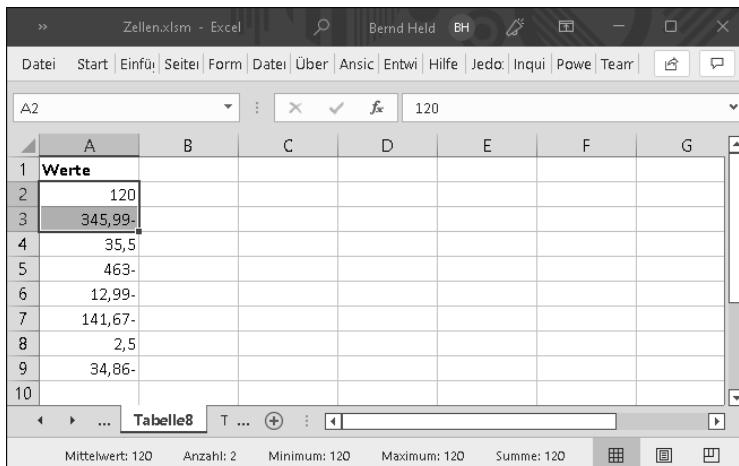


Abbildung 4.10 Das Ergebnis in der Statusleiste stimmt ja in 100 Jahren nicht!

Was genau soll denn das gewünschte Makro tun? Wie drücken Sie in der VBA-Sprache aus, was Sie tun möchten? Es reicht nicht zu sagen: »Bitte entferne auf der rechten Seite das Minuszeichen, und hänge es stattdessen am linken Rand wieder an.« Einen solchen Befehl gibt es in VBA nicht. Sie müssen versuchen, diese Aufgabe schrittweise anzugehen und zu lösen:

1. Prüfe, ob am rechten Rand einer Zelle ein Minuszeichen steht.
2. Wenn ja, dann übertrage vom linken Rand aus gesehen alle Zeichen bis auf das letzte (Minuszeichen).
3. Versorge die Zelle noch mit einem führenden Minuszeichen.

Arbeiten Sie diese einzelnen Schritte nacheinander ab, wie in Listing 4.8 gezeigt.

```
Sub MinuszeichenUmstellen()
    Dim lngZeile As Long
    Dim lngZeileMax As Long

    With Tabelle8
        .Range("C1").Value = "korrigierte Werte"
        lngZeileMax = .Range("A" & .Rows.Count).End(xlUp).Row

        For lngZeile = 2 To lngZeileMax

            If Right(.Range("A" & lngZeile).Value, 1) = "-" Then
                .Range("C" & lngZeile).Value = _
                    Replace(.Range("A" & lngZeile).Value, "-", "") * -1
            Else
                .Range("C" & lngZeile).Value = _
                    .Range("A" & lngZeile).Value
            End If

        Next lngZeile
    End With
End Sub
```

#### **Listing 4.8** Die Position des Minuszeichens tauschen

Mit der Funktion Right prüfen Sie eine bestimmte Anzahl von Zeichen einer Zelle von rechts gesehen. Indem Sie dieser Funktion den Wert 1 im zweiten Argument übergeben, werfen Sie einen Blick auf das eine Zeichen ganz rechts in der jeweiligen Zelle. Wenn dieses Zeichen dem Minuszeichen entspricht, dann können Sie die Funktion Replace auch hier einsetzen, um das Zeichen am rechten Rand zu eliminieren. Danach

multiplizieren Sie das Ergebnis mit dem Wert -1. Damit wird das Minuszeichen an der gewünschten Stelle automatisch gesetzt.

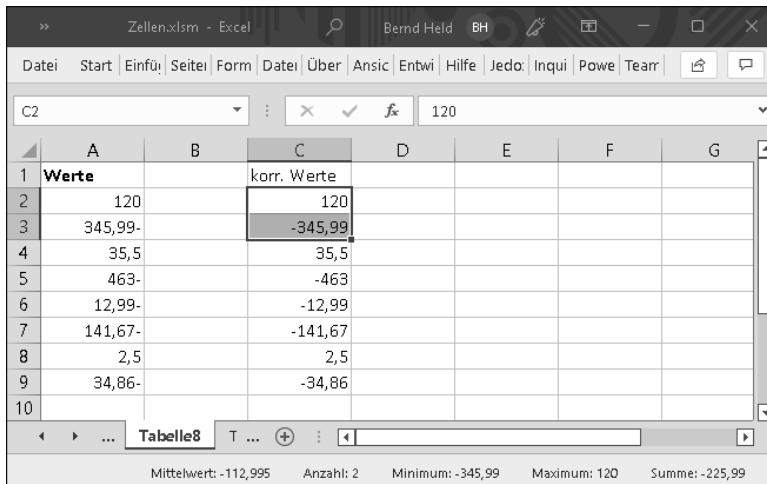


Abbildung 4.11 Excel kann jetzt alle Zellen in Spalte C berechnen.

#### 4.1.9 Daten umschlüsseln

Um Daten weiterverarbeiten zu können, müssen sie hin und wieder auch umgeschlüsselt werden. Sehen Sie sich dazu die Ausgangssituation in Abbildung 4.12 an.

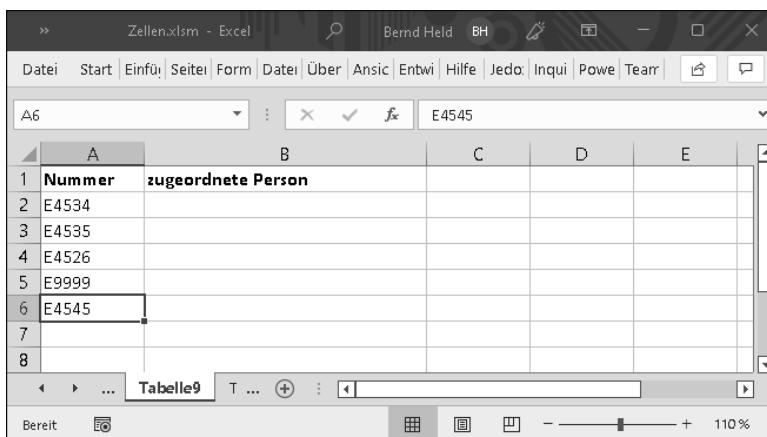


Abbildung 4.12 Welche Nummern werden von welcher Person verarbeitet?

Beantworten Sie diese Frage, indem Sie das Makro aus Listing 4.9 starten.

```
Sub UmschlüsselungVornehmen()
    Dim lngZeile As Long
    Dim lngZeileMax As Long
```

With Tabelle9

```
lngZeileMax = .Range("A" & .Rows.Count).End(xlUp).Row

For lngZeile = 2 To lngZeileMax

    Select Case Trim(UCase(.Cells(lngZeile, 1).Value))

        Case "E4534", "E9999"
            .Cells(lngZeile, 2).Value = "Christian"
        Case "E4535"
            .Cells(lngZeile, 2).Value = "Irina"
        Case "E4536"
            .Cells(lngZeile, 2).Value = "Erik"
        Case Else
            .Cells(lngZeile, 2).Value = "N.N"
    End Select

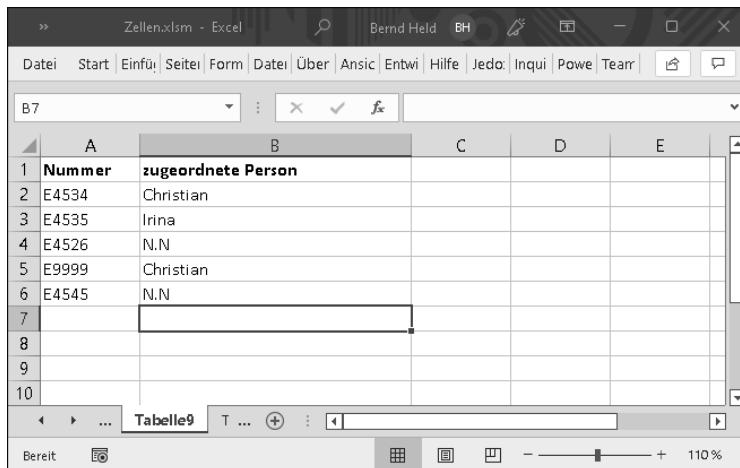
    Next lngZeile

End With
```

End Sub

#### Listing 4.9 Der jeweiligen Nummer den passenden Mitarbeiter zuordnen

Im Makro aus Listing 4.9 sorgen Sie im Innern der For ... Next-Schleife dafür, dass gegebenenfalls vorgestellte und nachgestellte Leerzeichen vorab berücksichtigt werden.



	A	B	C	D	E
1	Nummer	zugeordnete Person			
2	E4534	Christian			
3	E4535	Irina			
4	E4526	N.N			
5	E9999	Christian			
6	E4545	N.N			
7					
8					
9					
10					

Abbildung 4.13 Die Zuordnung passt und kann weiter angepasst werden.

Dazu verwenden Sie die Funktion `Trim`, die Sie bereits kennengelernt haben. Des Weiteren sorgen Sie mit der Funktion `UCase` dafür, dass Excel nicht zwischen Groß- und Kleinschreibung unterscheidet.

In einer `Select Case`-Anweisung ordnen Sie die einzelnen Nummern den dazugehörigen Mitarbeitern zu.

#### 4.1.10 Einen eindeutigen Schlüssel aus mehreren Spalten basteln

Wenn Sie beispielsweise gern mit der Tabellenfunktion `SVERWEIS` arbeiten, dann brauchen Sie einen eindeutigen Schlüssel, um die gewünschten Daten zuverlässig zu finden. Gibt es zu einem Schlüssel mehrere Datensätze, dann findet `SVERWEIS` immer nur den ersten Satz. Daher bilden Sie einen Schlüssel über mehrere Spalten. Schauen Sie sich zunächst einmal den Sachverhalt in Abbildung 4.14 an.

	A	B	C	D	E
1	Kostenstelle	Konto	Schlüssel		
2	5135	1000		Spalten Verknüpfen V1	Spalten Verknüpfen V2
3	5135	1001			
4	5135	1002			
5	8790	1003			
6	8790	1004			
7	8790	1005			
8	8790	1006			
9	8790	1007			
10	8790	1008			
11					

Abbildung 4.14 Die beiden Spalten A und B sollen verknüpft und in Spalte C ausgegeben werden.

Für diese Aufgabe soll aber keine Schleife verwendet werden. Gerade bei großen Datenmengen würde eine Schleife unter Umständen zu viel Zeit benötigen. Es soll aber auch keine Formel in Spalte C stehen, über die die beiden Spalten A und B verknüpft werden, da Formeln die Berechnung von Excel etwas lähmten. Was nun? Schauen Sie sich dazu das Makro aus Listing 4.10 an.

```
Sub SpaltenVerknuepfenFuerEindeutigenSchluesselV1()
```

```
With Tabelle10
```

```
With .Range("C2:C" & .UsedRange.Rows.Count)
```

```

        .Value = Evaluate(.Offset(0, -2).Address & "&" _
        & .Offset(0, -1).Address)
    End With

End With

End Sub

```

#### Listing 4.10 Pfeilschnell zwei Spalten miteinander verknüpfen – Variante 1

Mithilfe der Methode Evaluate simulieren Sie quasi eine Formel, die aber eben nicht als Formel in eine Zelle geschrieben, sondern als Festwert in der Zelle abgelegt wird. Da Sie vorher auf den kompletten Zielbereich referenzieren, wird die »Formel« als Festwert in allen Zellen eingefügt und entsprechend angepasst.

	A	B	C	D	E
1	Kostenstelle	Konto	Schlüssel		
2	5135	1000	51351000		
3	5135	1001	51351001		
4	5135	1002	51351002		
5	8790	1003	87901003		
6	8790	1004	87901004		
7	8790	1005	87901005		
8	8790	1006	87901006		
9	8790	1007	87901007		
10	8790	1008	87901008		
11					
12					

 The status bar at the bottom shows: Mittelwert: 75717671 Anzahl: 9 Minimum: 51351000 Maximum: 87901008 Summe: 681459036
 

Abbildung 4.15 Schneller geht's nicht – der eindeutige Schlüssel wurde generiert.

Alternativ können Sie die gleiche Aufgabe auch über eine andere Variante lösen, wie in Listing 4.11 gezeigt.

```
Sub SpaltenVerknuepfenFuerEindeutigenSchlueselV2()
```

```

With Tabelle10

    With .Range("C2:C" & .UsedRange.Rows.Count)
        .NumberFormat = "0"
        .FormulaR1C1 = "=RC[-2]&RC[-1]"
        .Copy
        .PasteSpecial Paste:=xlPasteValues
    End With

```

```
Application.CutCopyMode = False  
  
End With  
  
End Sub
```

**Listing 4.11** Pfeilschnell zwei Spalten miteinander verknüpfen – Variante 2

Auch dieser Ansatz zum Verknüpfen mehrerer Spalten kommt ohne Schleife aus. Dabei wird die Eigenschaft `FormulaR1C1` genutzt und eine Formel geschrieben. Dabei steht `R` für *row* (Zeile) und `C` für *column* (Spalte). Der Bezug für die Formel, wenn Sie von Spalte `C` beginnen, ist `-2`, um Spalte `A` zu adressieren, und `-1` zeigt auf Spalte `B`. Einen Zeilenversatz gibt es dabei nicht, da der Schlüssel in der gleichen Zeile gebildet werden muss. Zunächst wird also die Formel in den verwendeten Bereich von Spalte `C` geschrieben, dann wird dieser Bereich kopiert und über die Methode `PasteSpecial` über Einsatz der Konstanten `xlPasteValues` als Festwert eingefügt. Über die Eigenschaft `CutCopyMode`, der Sie den Wert `False` zuweisen, sorgen Sie dafür, dass der Laufrahmen um den Bereich, der beim Kopieren entsteht, wieder zurückgesetzt wird.

## 4.2 Zellen, Rahmen und Schriften formatieren

Neben den bereits vorgestellten Zahlenformaten gibt es die normalen Formate, die für Zellen, Schriften und Rahmen eingesetzt werden können. Einige davon habe ich im Buch schon behandelt. An dieser Stelle erfolgt nun eine Vertiefung.

**Hinweis**

Sie finden alle folgenden Makros in der Datei `Zellen.xlsxm` im Modul `mdl_Formatieren`.

### 4.2.1 Schriftart ermitteln

In Windows stehen Ihnen Hunderte verschiedener Schriftarten zur Verfügung. Möchten Sie prüfen, welche Schriftart beispielsweise in der momentan aktiven Zelle verwendet wird, dann starten Sie das Makro aus Listing 4.12.

```
Sub SchriftartErmitteln()  
  
    MsgBox ActiveCell.Font.Name  
  
End Sub
```

**Listing 4.12** Schriftart ermitteln

# Kapitel 6

## Tabellen und Diagramme programmieren

*Das Objekt »Worksheet« symbolisiert das Tabellenblatt. Tabellenblätter lassen sich individuell modifizieren. Sie können Tabellenblätter einfügen, umbenennen, löschen, drucken, kopieren, verschieben und vieles mehr. Über das Objekt »ChartObject« erstellen Sie Diagramme, die Sie in Tabellen einbetten.*

In diesem Kapitel erfahren Sie anhand ausgesuchter Beispiele aus der täglichen Praxis mehr über den Einsatz von Eigenschaften und Methoden des Objekts `Worksheet`. Auch die Themen Pivot-Tabellen und Diagramme werde ich in diesem Kapitel behandeln.

### Kapitelbegleitende Beispiele zum Download

Sie finden alle Beispiele in der Datei *Tabellen.xlsxm* aus dem Downloadpaket, das Sie von [www.rheinwerk-verlag.de/6028](http://www.rheinwerk-verlag.de/6028) herunterladen können.



### 6.1 Tabellen einfügen

#### Hinweis

Die Datei *Tabellen.xlsxm* enthält im Modul `mdl_Allgemein` alle folgenden Makros.

Standardmäßig bietet Excel Ihnen bei der Erstellung einer neuen Arbeitsmappe drei Tabellenblätter an. Wenn Sie weitere hinzufügen möchten, setzen Sie die Methode `Add` ein wie in Listing 6.1. Das neu eingefügte Tabellenblatt wird immer vor dem aktiven Tabellenblatt der Arbeitsmappe eingefügt.

```
Sub TabelleEinfügen()
```

```
    Worksheets.Add
```

```
End Sub
```

**Listing 6.1** Neues Tabellenblatt einfügen

Möchten Sie ein Tabellenblatt an einer bestimmten Position einfügen, starten Sie das Makro aus Listing 6.2.

```
Sub TabelleAnPositionEinfügen()
```

```
    Worksheets.Add Before:=ThisWorkbook.Worksheets(1)
```

```
End Sub
```

**Listing 6.2** Neues Tabellenblatt als erstes Blatt in eine Mappe einfügen

In Listing 6.2 wurde die neue Tabelle zu Beginn der Arbeitsmappe, also als erste Tabelle, eingefügt. Das bisherige Tabellenblatt mit dem Index 1 wird dann eine Position nach rechts geschoben. Möchten Sie die neue Tabelle ganz am Ende einfügen, also ganz rechts, setzen Sie das Makro aus Listing 6.3 ein.

```
Sub TabelleAmEndeEinfügen()
```

```
    Worksheets.Add After:=Worksheets(Worksheets.Count)
```

```
End Sub
```

**Listing 6.3** Neues Tabellenblatt am Ende der Arbeitsmappe einfügen

Um die gewünschte Einfügeposition des neuen Tabellenblattes zu ermitteln, müssen Sie zuerst herausfinden, wie viele Tabellenblätter bereits in der Arbeitsmappe enthalten sind. Dabei hilft Ihnen die Eigenschaft `Count`. Sie liefert die Anzahl der Tabellenblätter. Danach brauchen Sie nur noch den Parameter `After` anzugeben, und das neue Tabellenblatt wird als letztes Tabellenblatt in die Arbeitsmappe eingefügt.

## 6.2 Tabellenblätter benennen

Excel vergibt beim Einfügen von Tabellennamen selbstständig Namen, die sich aus dem Ausdruck `TABELLE` und einer fortlaufenden Zahl zusammensetzen. Wenn Sie andere Namen verwenden möchten, können Sie dies jederzeit tun.

### 6.2.1 Eine neue Mappe erstellen, zwölf Monatstabellen anlegen und benennen

Bei der nächsten Aufgabe (siehe Listing 6.4) soll eine neue Mappe mit zwölf Tabellen erstellt werden. Diese Tabellen sollen nach den Monatsnamen benannt werden.

```
Sub MappeMit12MonatenAnlegen()
    Dim intAnz As Integer
    Dim wkbMappe As Workbook
    Dim wksBlatt As Worksheet

    intAnz = Application.SheetsInNewWorkbook
    Application.SheetsInNewWorkbook = 12
    Set wkbMappe = Workbooks.Add
    Application.SheetsInNewWorkbook = intAnz

    For Each wksBlatt In wkbMappe.Worksheets
        wksBlatt.Name = MonthName(wksBlatt.Index)
    Next wksBlatt

    End Sub
```

#### **Listing 6.4** Eine neue Mappe mit zwölf Tabellen anlegen und diese nach Monatsnamen benennen

Ermitteln Sie zunächst über die Eigenschaft SheetsInNewWorkbook, welche Applikationseinstellung bezüglich der Tabellenanzahl festgelegt ist, wenn eine neue Mappe angelegt wird. Speichern Sie diesen Wert in der Variablen intAnz zwischen. Jetzt ändern Sie den Wert dieser Eigenschaft in zwölf Tabellen. Durch die Methode Add, die auf das Auflistungsobjekt Workbooks angewendet wird, wird nun eine neue Arbeitsmappe mit zwölf Tabellen erstellt. Stellen Sie jetzt am besten gleich wieder die vorher eingestellte Anzahl der angebotenen Tabellen ein. Dazu weisen Sie der Eigenschaft SheetsInNewWorkbook den Inhalt der Variablen intAnz zu.

Setzen Sie danach eine Schleife des Typs For Each ... Next ein, in der Sie alle zwölf Tabellen nacheinander verarbeiten. In der Schleife benennen Sie die Tabellen. Dazu setzen Sie die Funktion MonthName ein. Diese Funktion benötigt einen Wert zwischen 1 und 12, um den gewünschten Monat direkt aus der Windows-Systemsteuerung in der dort eingestellten Landessprache zu holen. Diesen Wert leiten Sie über die Eigenschaft Index der jeweiligen Tabelle ab.

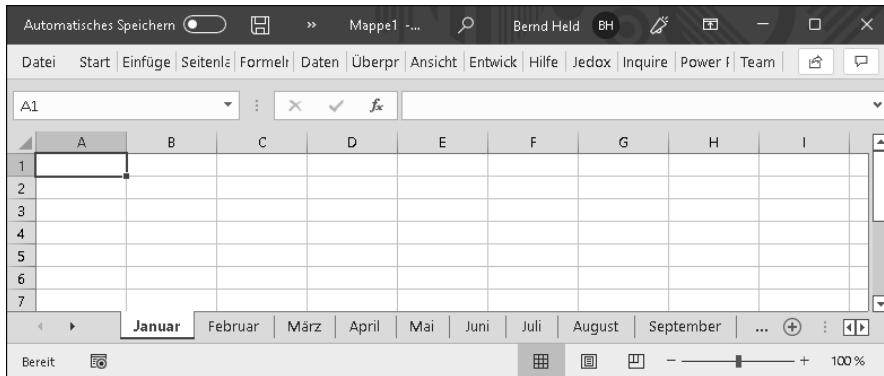


Abbildung 6.1 Die Monatstabellen wurden automatisch angelegt.

### 6.2.2 Eine neue Mappe mit den nächsten 14 Tagen anlegen

Im Beispiel aus Listing 6.5 wird eine neue Arbeitsmappe zunächst mit nur einer Tabelle angelegt. Danach werden über eine Schleife 14 weitere Tabellen hinzugefügt und mit einem fortlaufenden Datum versehen.

```
Sub TabellenMitDatumEinfügen()
    Dim inTabz As Integer
    Dim intAnz As Integer
    Dim wkbMappe As Workbook
    Dim wksBlatt As Worksheet

    intAnz = Application.SheetsInNewWorkbook
    Application.SheetsInNewWorkbook = 1
    Set wkbMappe = Workbooks.Add
    Application.SheetsInNewWorkbook = intAnz

    For inTabz = 1 To 14
        wkbMappe.Worksheets.Add after:=wkbMappe.Worksheets(Worksheets.Count)
        wkbMappe.Worksheets(Worksheets.Count).Name = Date + inTabz
    Next inTabz

End Sub
```

Listing 6.5 Eine neue Mappe für die nächsten 14 Tage wird angelegt.

Setzen Sie die Eigenschaft SheetsInNewWorkbook auf den Wert 1, und fügen Sie anschließend in einer For ... Next-Schleife weitere 14 Tabellen über den Einsatz der Methode

Add hinzu. Innerhalb der Schleife benennen Sie Tabellen beginnend vom aktuellen Tagesdatum. Mit jedem Schleifendurchlauf wird der Schleifenzähler, der gleichzeitig der Tageszähler ist, um den Wert 1 inkrementiert.

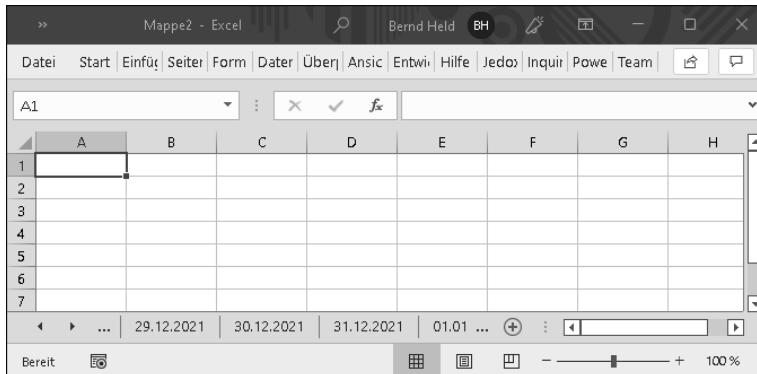


Abbildung 6.2 Eine neue Mappe für die nächsten 14 Tage steht bereit.

### 6.2.3 Tabelle einfügen und gleichzeitig benennen

Selbstverständlich können Sie das Einfügen von neuen Tabellenblättern und deren Benennung auch in einem Aufwasch erledigen, wie Listing 6.6 unter Beweis stellt.

```
Sub TabelleEinfügenUndBenennen()
    Worksheets.Add.Name = "Tabelle10"
End Sub
```

Listing 6.6 Neue Tabelle einfügen und benennen in einem Schritt

Allerdings müssen Sie sich sicher sein, dass der Name nicht schon in Verwendung ist, da es sonst zu einem Laufzeitfehler kommt.

## 6.3 Tabellen markieren

Um eine einzige Tabelle zu markieren, können Sie den Befehl `Worksheets("Tabelle2").Select` anwenden. Sollen es ein paar Tabellen mehr sein, dann wenden Sie das Makro aus Listing 6.7 an.

```
Sub MehrereTabellenMarkieren()  
  
    Sheets(Array("Tabelle1", "Tabelle2")).Select  
  
End Sub
```

**Listing 6.7 Mehrere Tabellen markieren**

Mithilfe der Funktion `Array` bilden Sie ein Datenfeld, in das Sie die Namen der Tabellen aufnehmen, die Sie markieren möchten.

Soll diese Lösung ein wenig dynamischer sein, dann markieren Sie in der nächsten Aufgabe einmal alle Tabellen einer Arbeitsmappe mit Ausnahme der ersten Tabelle. Wie das geht, entnehmen Sie dem Makro aus Listing 6.8.

```
Sub MehrereTabellenMarkierenÜberArray()  
    Dim lngZ As Long  
    Dim intTab As Integer  
    Dim Vardat() As Long  
  
    intTab = ThisWorkbook.Worksheets.Count  
  
    ReDim Vardat(1 To intTab - 1)  
  
    For lngZ = 2 To intTab  
        Vardat(lngZ - 1) = lngZ  
    Next lngZ  
  
    ThisWorkbook.Worksheets(Vardat).Select  
  
End Sub
```

**Listing 6.8 Mehrere Tabellen markieren (nur nicht die erste)**

Ermitteln Sie im ersten Schritt die Gesamtzahl der Tabellen, die sich in der aktiven Arbeitsmappe befinden, und speichern Sie diese Information in der Variablen `intTab`. Danach definieren Sie mit der Anweisung `ReDim` ein Datenfeld in der Größe der Anzahl der Tabellen in der Arbeitsmappe. Von dieser ermittelten Größe subtrahieren Sie den Wert 1, da Sie die erste Tabelle nicht markieren möchten. In einer Schleife füllen Sie dann das Datenfeld. Am Ende der Schleife stehen die Namen aller Tabellen im Datenfeld. Markieren Sie anschließend alle im Datenfeld stehenden Tabellen mit der Methode `Select`.

## 6.4 Tabellenblätter gruppieren

In Excel haben Sie die Möglichkeit, Ihre Arbeit an einem Tabellenblatt automatisch auch für andere Tabellenblätter gültig zu machen. Dazu gruppieren Sie die einzelnen Tabellenblätter. Manuell klappt das, indem Sie die **Strg**-Taste gedrückt halten und die einzelnen Tabellenregister mit der linken Maustaste anklicken. Das Ergebnis dieser Aktion erreichen Sie selbstverständlich auch mit VBA. Im Folgenden erfahren Sie, wie Sie das machen.

### 6.4.1 Mehrere Tabellen gruppieren

Die Funktion **Array** ermöglicht es Ihnen, eine durch Kommas getrennte Liste von Werten (hier Tabellennamen) anzugeben (siehe Listing 6.9). Auch hier ist wieder die **On Error-Anweisung** wichtig, um eine Fehlermeldung zu vermeiden, falls eines der Tabellenblätter nicht vorhanden ist.

```
Sub MehrereTabellenblätterMarkieren()

    On Error Resume Next
    Sheets(Array("Tabelle2", "Tabelle3", "Tabelle5")).Select

End Sub
```

**Listing 6.9** Mehrere Tabellenblätter gruppieren

### 6.4.2 Alle Tabellen gruppieren

Möchten Sie alle Tabellenblätter einer Arbeitsmappe gruppieren, können Sie die Tabellenblätter in ein Array einlesen und anschließend gruppieren. Dazu wenden Sie das Makro aus Listing 6.10 an.

```
Sub AlleTabellenMarkieren()
    Dim lngZ As Long
    Dim lngTab As Long
    Dim TabArray() As Long

    lngTab = ThisWorkbook.Worksheets.Count

    ReDim TabArray(1 To lngTab)
    On Error Resume Next
```

```
For lngZ = 1 To lngTab
    TabArray(lngZ) = 1
Next lngZ

ThisWorkbook.Worksheets(TabArray).Select

End Sub
```

**Listing 6.10** Alle Tabellenblätter einer Arbeitsmappe gruppieren

Ermitteln Sie mit der Eigenschaft `Count` die Anzahl der Tabellenblätter, die in der Arbeitsmappe enthalten sind. Mit der Anweisung `ReDim` reservieren Sie Speicherplatz für die dynamische Datenfeldvariable `TabArray`. Danach füllen Sie das Array mithilfe einer `For ... Next`-Schleife. Im Anschluss daran werden alle Tabellenblätter der Arbeitsmappe gruppiert.

#### 6.4.3 Gruppierte Tabellen übertragen

Im nächsten Beispiel in Listing 6.11 werden alle gruppierten Tabellen in eine neue Arbeitsmappe eingefügt.

```
Sub GruppierteTabellenInNeueMappeTransferieren()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWindow.SelectedSheets
        wksBlatt.Copy
    Next wksBlatt

End Sub
```

**Listing 6.11** Gruppierte Tabellen in neue Arbeitsmappe überführen

Setzen Sie die Eigenschaft `SelectedSheets` ein, um alle markierten Tabellenblätter zu ermitteln. Kopieren Sie all diese Tabellen mithilfe der Methode `Copy`.

#### 6.4.4 Gruppierte Tabellen ermitteln

Möchten Sie herausfinden, welche Tabellen in Ihrer Arbeitsmappe gruppiert sind, dann starten Sie das folgende Makro aus Listing 6.12.

```
Sub GruppierteBlätterErmitteln()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Windows(1).SelectedSheets
```

```
    MsgBox wksBlatt.Name
    Next wksBlatt

End Sub
```

#### **Listing 6.12** Gruppierte Tabellen ermitteln

Mithilfe der Eigenschaft `Name` ermitteln Sie die Namen der gruppierten Tabellen. Gruppierte Tabellen fragen Sie über die Eigenschaft `SelectedSheets` ab.

## 6.5 Tabellenblätter löschen

Wie Sie Tabellenblätter einfügen, wissen Sie jetzt. Aber wie löschen Sie Tabellenblätter? Dafür setzen Sie die Methode `Delete` ein.

### 6.5.1 Eine Tabelle löschen

Beim Beispiel aus Listing 6.13 wird `TABELLE1` in der Arbeitsmappe gelöscht. Hierbei müssen Sie zwischen dem »normalen« Tabellennamen und dem Codenamen der Tabelle unterscheiden.

```
Sub TabellenblattLöschen()
    On Error GoTo fehler
    Sheets("Tabelle1").Delete

    'oder eben über den Codenamen
    'Tabelle1.Delete

    Exit Sub

fehler:
    MsgBox "Es gibt keine Tabelle1 zum Löschen"
End Sub
```

#### **Listing 6.13** Ein benanntes Tabellenblatt löschen

Zu Beginn sorgt die Anweisung `On Error` dafür, dass im Fehlerfall sofort zur Textmarke `fehler` gesprungen wird. Ein Fehler kann z. B. auftreten, wenn die Tabelle gar nicht in der Arbeitsmappe enthalten ist. Danach wird versucht, die Tabelle `TABELLE1` über den Einsatz der Methode `Delete` zu löschen. Sollte der Vorgang erfolgreich sein, wird die nächste Zeile abgearbeitet, wenn nicht, wird zur Textmarke `fehler` gesprungen. Die Anweisung `Exit Sub` sorgt dafür, dass nach dem erfolgreichen Löschen des Tabellen-

blattes das Makro sofort beendet, also die Textmarke fehler nicht mehr abgearbeitet wird. Die Textmarke fehler leitet die Fehlerbehandlung ein. Sie wird nur ausgeführt, wenn z. B. versucht wurde, eine Tabelle zu löschen, die es gar nicht mehr gibt. Als Fehlerreaktion wird eine einfache Meldung auf dem Bildschirm ausgegeben.

Diese On Error-Geschichte sollten Sie aber auch nicht übertreiben. Oft werden mir Quellcodes zur Begutachtung vorgelegt, bei denen es davon nur so wimmelt. Einen möglichen Fehler mehrfach zu ignorieren ist keine gute Reaktion auf einen Fehler. Besser wäre eine Funktion, die immer vorher prüft, ob sich eine Tabelle in der Arbeitsmappe befindet. In diesem Fall bräuchten Sie keine Fehlerbehandlung mehr. In meinen Softwareprojekten habe ich immer eine solche Funktion zur Verfügung (siehe Listing 6.14).

```
Function TabDa(strBlatt As String) As Boolean
Dim wksBlatt As Worksheet

For Each wksBlatt In ThisWorkbook.Worksheets

    If wksBlatt.Name = strBlatt Then
        TabDa = True
        Exit For
    End If

    Next wksBlatt

End Function
```

**Listing 6.14** Funktion, die prüft, ob eine bestimmte Tabelle in der Mappe vorhanden ist

Immer, wenn Sie eine Tabelle löschen oder auf eine Tabelle zugreifen müssen, rufen Sie vorher die Funktion TabDa auf und übergeben ihr den Namen der entsprechenden Tabelle. In der Funktion selbst wird eine Schleife des Typs For Each ... Next durchlaufen, die den Namen der an die Funktion übergebenen Tabelle mit dem jeweiligen Namen der Tabelle vergleicht, den Sie durch die Schleife ansprechen. Wird die gesuchte Tabelle in der Arbeitsmappe gefunden, dann setzen Sie den Rückgabewert der Funktion auf True und verlassen vorzeitig die Schleife, indem Sie die Anweisung Exit For einsetzen. Damit springen Sie direkt im Anschluss auch aus der Funktion und landen wieder im aufrufenden Makro aus Listing 6.15.

```
Sub TabelleBenennenMitVorherigerPrüfung()

    If TabDa("Tabelle10") = False Then
        Worksheets.Add.Name = "Tabelle10"
```

```
Else
    MsgBox "Die Tabelle10 ist bereits in der Mappe!"
End If

End Sub
```

**Listing 6.15** Das Makro ruft eine Funktion auf und verarbeitet ihre Rückmeldung.

### 6.5.2 Bestimmte Tabellen aus einer Mappe entfernen

Im Beispiel aus Listing 6.16 werden alle Tabellen aus der Arbeitsmappe entfernt, die im Tabellennamen das Kürzel »neu« aufweisen.

```
Sub BestimmteTabellenEntfernen()
    Dim wksBlatt As Worksheet

    Application.DisplayAlerts = False

    For Each wksBlatt In ActiveWorkbook.Worksheets

        If wksBlatt.Name Like "*neu*" Then
            wksBlatt.Delete
        End If

    Next wksBlatt

    Application.DisplayAlerts = True

End Sub
```

**Listing 6.16** Alle Tabellen mit einer bestimmten Zeichenfolge im Namen werden gelöscht.

Mithilfe der Eigenschaft `DisplayAlerts` schalten Sie die Excel-Warnmeldungen temporär aus, indem Sie dieser Eigenschaft den Wert `False` zuweisen. Dies ist wichtig, da Sie sonst die Löschung einer jeden Tabelle bestätigen müssten. In einer Schleife der Form `For Each ... Next` arbeiten Sie alle Tabellen der Arbeitsmappe nacheinander ab. Im Innern der Schleife prüfen Sie mit dem Operator `Like`, ob im Tabellennamen der Begriff »neu« vorkommt. Dabei unterscheidet dieser Operator zwischen Groß- und Kleinschreibung. Wird eine Tabelle gefunden, deren Name die Zeichenfolge »neu« enthält, dann wird die Methode `Delete` verwendet, um die Tabelle zu löschen.

### Hinweis

Möchten Sie nicht, dass Excel zwischen Groß- und Kleinschreibung unterscheidet, dann tauschen Sie die Bedingung in Listing 6.16 durch die folgende Bedingung aus:

```
If UCase(wksBlatt.Name) Like "*NEU*" Then
```

### 6.5.3 Tabellen mit gefärbten Registerlaschen entfernen

Zur Identifizierung zu löschernder Tabellen könnten Sie auch die Farbe der Tabellenreiter heranziehen. Das Makro aus Listing 6.17 entfernt alle Tabellen ohne Rückfrage aus der Arbeitsmappe, die mit rotem Tabellenreiter formatiert wurden.

```
Sub FarbTabellenEntfernen()
    Dim wksBlatt As Worksheet

    Application.DisplayAlerts = False

    For Each wksBlatt In ActiveWorkbook.Worksheets

        If wksBlatt.Tab.ColorIndex = 3 Then

            wksBlatt.Delete

        End If

    Next wksBlatt

    Application.DisplayAlerts = True

End Sub
```

**Listing 6.17** Alle Tabellen mit einem roten Tabellenreiter werden entfernt.

Über das Objekt Tab sprechen Sie den Tabellenreiter einer Tabelle an. Mithilfe der Eigenschaft ColorIndex lesen Sie die Farbe des Tabellenreiters aus. Ist der Tabellenreiter rot eingefärbt, dann gibt die Eigenschaft die Farbnummer 3 zurück. In diesem Fall wenden Sie die Methode Delete an, um die Tabelle zu löschen.

### Hinweis

Möchten Sie nicht nur rote, sondern alle farbigen Tabellen aus der Mappe entfernen, dann passen Sie das Löschkriterium wie folgt an:

```
If wksBlatt.Tab.ColorIndex > 0 Then
```

#### 6.5.4 Leere Tabellen aus Arbeitsmappen entfernen

Bei der nächsten Lösung sehen Sie im Makro aus Listing 6.18, wie Sie leere Tabellen aus einer Arbeitsmappe entfernen.

```
Sub LeereTabellenAusMappeEntfernen()
    Dim Blatt As Worksheet
    Application.DisplayAlerts = False
    For Each Blatt In ActiveWorkbook.Worksheets
        If Application.WorksheetFunction.CountA(Blatt.Cells) = 0 Then
            Blatt.Delete
        End If
        Application.DisplayAlerts = True
    Next Blatt

End Sub
```

#### Listing 6.18 Leere Tabellen aus der Arbeitsmappe entfernen

In einer Schleife des Typs For Each ... Next arbeiten Sie sich Tabelle für Tabelle durch die aktive Arbeitsmappe hindurch. Innerhalb der Schleife prüfen Sie mithilfe der Tabellenfunktion ANZAHL2 (engl. CountA), ob die Anzahl der Zellen, die einen Eintrag enthalten, null ist. Ist dies der Fall, dann sind in der Tabelle keine Daten vorhanden, und Sie wenden die Methode Delete an, um die leere Tabelle zu löschen.

## 6.6 Tabellenblätter ein- und ausblenden

Wenn Sie bestimmte Tabellenblätter nicht mit einem Passwort schützen, jedoch trotzdem einen gewissen Schutz Ihrer Daten erreichen möchten, können Sie Tabellenblätter auch ausblenden. Das Ein- und Ausblenden von Tabellenblättern erreichen Sie mit der Eigenschaft `Visible`, wie Listing 6.19 zeigt.

```
Sub TabelleAusblenden()
    On Error Resume Next
    Worksheets("Tabelle1").Visible = False
    ' oder
    Tabelle1.Visible = xlSheetHidden
End Sub
```

#### Listing 6.19 Tabellenblatt ausblenden

Nachdem Sie das Makro TabelleAusblenden ausgeführt haben, wird die Tabelle in der Arbeitsmappe nicht mehr angezeigt.

Anwenderinnen und Anwender der Excel-Versionen 2007 bis 2024 sowie Microsoft 365 klicken mit der rechten Maustaste auf einen beliebigen Tabellenreiter und wählen den Befehl EINBLENDEN aus dem Kontextmenü aus.

Das Einblenden eines ausgeblendeten Tabellenblattes funktioniert in VBA wie in Listing 6.20.

```
Sub TabelleEinblenden()

    On Error Resume Next

    Sheets("Tabelle1").Visible = True

    ' oder

    Tabelle1.Visible = xlSheetVisible

End Sub
```

**Listing 6.20** Tabellenblatt wieder einblenden

### 6.6.1 Tabellenblätter sicher ausblenden

Wenn Sie verhindern möchten, dass die Anwenderin Ihre ausgeblendeten Tabellenblätter über die Benutzeroberfläche wieder einblendet, dann verwenden Sie bei der Eigenschaft Visible die Konstante xlSheetVeryHidden.

```
Sub TabelleSicherAusblenden()

    On Error Resume Next
    Tabelle1.Visible = xlSheetVeryHidden

End Sub
```

**Listing 6.21** Tabelle ausblenden (sichere Methode)

In diesem Fall können Sie Ihre ausgeblendete Tabelle nur mit einem Makro wieder verfügbar machen. Dazu setzen Sie das Makro aus Listing 6.21 ein.

### 6.6.2 Tabellen je nach Status ein- oder ausblenden

In einer Arbeitsmappe sollen alle eingeblendeten Tabellenblätter ausgeblendet bzw. alle ausgeblendeten Tabellenblätter eingeblendet werden. Das Makro zur Umsetzung dieser Aufgabe sehen Sie in Listing 6.22.

```
Sub TabellenJeNachStatusEinAusblenden()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Worksheets

        Select Case wksBlatt.Visible
            Case xlSheetHidden: Blatt.Visible = xlSheetVisible
            Case xlSheetVisible: Blatt.Visible = xlSheetHidden
        End Select

    Next wksBlatt

End Sub
```

#### **Listing 6.22** Tabellenblätter je nach Status ein- oder ausblenden

In einer Schleife der Art For Each ... Next überprüfen Sie mithilfe einer Select Case-Anweisung, wie der Status der Eigenschaft `Visible` für das jeweilige Tabellenblatt ist. Je nach Status wird der Eigenschaft dann entweder die Konstante `xlSheetVisible` bzw. `xlSheetHidden` zugewiesen.

#### **Achtung**

Achten Sie darauf, dass Sie die Anweisung `On Error` in das Makro integrieren. In einer Arbeitsmappe muss immer wenigstens eine Tabelle sichtbar bleiben. Versucht nun das Makro, das letzte Tabellenblatt auszublenden, kommt es zum Fehlerfall, den Sie aber mit dieser Anweisung abfangen.

### 6.6.3 Alle Tabellenblätter anzeigen

Ausgeblendete Tabellenblätter werden oft vergessen. Diese versteckten Tabellenblätter schlummern dann jahrelang in Arbeitsmappen. Eines Tages erfahren Sie mehr durch Zufall, dass es in der Arbeitsmappe versteckte Tabellenblätter gibt.

Schreiben Sie daher ein Makro wie das in Listing 6.23, das in der aktiven Arbeitsmappe alle Tabellenblätter wieder sichtbar macht.

```
Sub VersteckteBlätterEinblenden()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Worksheets
        wksBlatt.Visible = True
    Next wksBlatt

End Sub
```

**Listing 6.23** Alle Tabellenblätter einblenden

In einer For Each ... Next-Schleife setzen Sie die Eigenschaft `Visible` aller Tabellenblätter auf den Wert `True`.

#### 6.6.4 Alle Tabellen außer der aktiven Tabelle ausblenden

Wenn Sie möchten, können Sie alle Tabellenblätter einer Arbeitsmappe mit Ausnahme des aktiven Tabellenblattes ausblenden, indem Sie das Makro aus Listing 6.24 starten.

```
Sub NurAktivesBlattSichtbar()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Worksheets

        If wksBlatt.Name <> ActiveSheet.Name Then
            wksBlatt.Visible = xlSheetHidden
        End If

    Next wksBlatt

End Sub
```

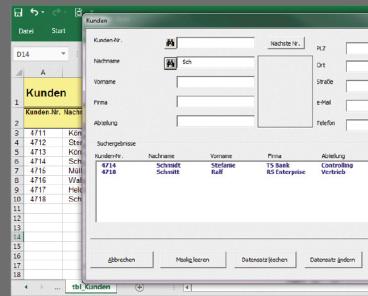
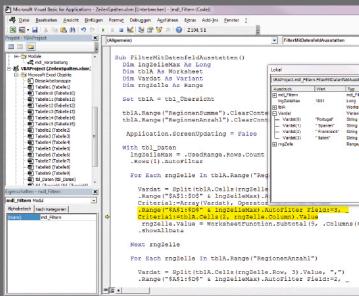
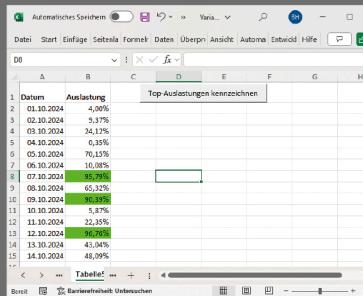
**Listing 6.24** Alle Tabellen außer der aktiven Tabelle werden ausgeblendet.

Definieren Sie zuerst eine Objektvariable vom Typ `Worksheet`. Danach greifen Sie in einer Schleife des Typs `For Each ... Next` auf das Auflistungsobjekt `Worksheets` zu, das alle Tabellenblätter der aktiven Arbeitsmappe enthält. Innerhalb der Schleife vergleichen Sie den Namen des aktiven Tabellenblattes mit dem jeweiligen Tabellenblatt aus dem Auflistungsobjekt. Mit der Eigenschaft `Visible`, die Sie auf den Wert `False` oder die Konstante `xlSheetHidden` setzen, blenden Sie alle Tabellenblätter aus der Arbeitsmappe mit Ausnahme des aktiven Tabellenblattes aus.

## Geprüfte VBA-Lösungen für die tägliche Praxis

Dieses Standardwerk zeigt Ihnen zuverlässig, wie Sie wiederkehrende Aufgaben und Anwendungen in Excel programmieren – praxisnah und gut verständlich. Neben einer Einführung in Excel-VBA bietet Ihnen dieses Buch eine umfassende Sammlung an VBA-Programmen für alle Aufgaben. Geeignet für Excel 2016 bis 2024 sowie Microsoft 365.

»Über 650 Makros  
zum sofortigen  
Einsatz«



Aufgaben mit VBA vereinfachen

Eigene Makros schreiben

UserForms entwickeln

## Einstegen und Excel automatisieren

Ob Sie Tabellen einrichten, Diagramme erstellen, Pivot-Tabellen aktualisieren oder nur Zeilen löschen wollen: Hier erfahren Sie, wie Sie all dies mit VBA erledigen. Die Aufgaben sind übersichtlich aufgelistet, sodass Sie schnell das Gewünschte finden.

## Anwendungen entwickeln

Funktionen, Ereignisse, Dialoge, UserForms, Schaltflächen, Listen, Kontextmenüs und Ribbons: Sie erfahren alles über die Programmierung von Anwendungen für Excel.

## Schnell, sicher, praktisch: geprüfte Makros

Hier finden Sie jederzeit die passende Lösung für Ihre beruflichen Excel-Aufgaben. Über 650 sorgfältig getestete Makros werden umfassend beschrieben und stehen Ihnen sofort für verschiedenste Automatisierungsaufgaben zur Verfügung.



Alle im Buch beschriebenen Makros können Sie herunterladen und direkt in Ihre Anwendungen integrieren.



**Bernd Held** ist Spezialist für Excel, VBA-Programmierung, Access und allgemeine Office-Themen. In Schulungen, Büchern, Fachartikeln, Foren, Lernvideos und seinem VBA-Newsletter gibt er sein Wissen weiter. Er wurde von Microsoft mehrfach als MVP für Excel ausgezeichnet.

## Aus dem Inhalt

### Abläufe automatisieren

Zellen, Zeilen, Spalten und Bereiche bearbeiten

Tabellenblätter und Arbeitsmappen programmieren

Eigene Funktionen schreiben

Diagramme und Pivot-Tabellenberichte erstellen

Elegant programmieren: SQL, ADO, Arrays, Dictionaries etc.

Mit Ereignissen arbeiten

VBE-Programmierung

### Anwendungen entwickeln

Dialoge und Steuerelemente  
Kontextmenüs und Ribbons  
Datenzugriff aus Excel

### Kleine Helfer

VBA-Grundlagen, KI nutzen  
Fehlerbehandlung  
Sofort einsetzbare Makros

