# EXCEL
# VOLLAUTOMATISCH

Microsoft bietet ein praktisches Werkzeug, um wiederholende Aufgaben in Excel zu automatisieren: die Makros. **Erstellen und nutzen Sie eigene Excel-Makros – sogar ganz ohne Programmierkenntnisse.**

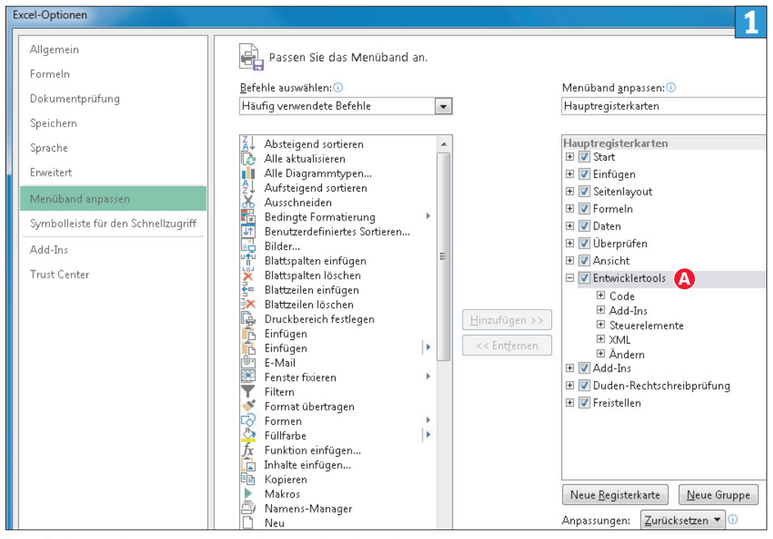
Alle Makros unter **www.pctipp.ch** **WEBCODE 70123**

**VON JANIS BERNEKER**

Die Tabellenkalkulation Excel gehört zu den meistverwendeten PC-Programmen überhaupt. Egal, ob Sie Kundenlisten pflegen, die Buchhaltung erledigen oder Diagramme erstellen wollen – mit Excel funktionierts. Die Schattenseite des Tabellenallrounders: Oft ist mühsame Handarbeit nötig, um die Daten in die gewünschte Form zu bringen. Oder geht es doch einfacher? Ja, mithilfe von Makros. Diese lösen Aufgaben, bei denen einfache Formeln an ihre Grenzen stossen. Müssen Sie vielleicht leere Zeilen löschen? Doppelte Einträge entfernen? Werte vergleichen und bei Unterschieden anstreichen? Das lässt sich dank Makros alles mit einem einzigen Klick erledigen. Dabei brauchen Sie oft gar keine Programmierkenntnisse.

Gleich nachfolgend lesen Sie, wie Sie Makros mit einer simplen Aufnahmefunktion erstellen. Für anspruchsvollere Aufgaben kommt man um ein wenig Programmieren jedoch nicht herum. Eine Einführung in die Makro-Programmierung gibts im Abschnitt «Makros für Profis» ab S. 40.

**Hinweis:** Alle Makro-Beispiele finden Sie als Download unter www.pctipp.ch mit Webcode **70123** (Info zum PCtipp-Webcode, S. 4).




**Die «Entwicklertools» müssen in Office erst aktiviert werden**

FOTO ISTOCKPHOTO.COM/KEMALBAS

## Was sind Makros?

Bei Makros handelt es sich um Miniprogramme, die in der Sprache VBA programmiert sind. VBA steht für «Visual Basic for Applications» und ist eine vereinfachte Form der verbreiteten Sprache Visual Basic. Während sich dieser Artikel auf Excel beschränkt, funktionieren Makros auch in vielen anderen Office-Anwendungen wie Word oder PowerPoint – und das sogar übergreifend.

## Makros ohne programmieren

Makros müssen nicht unbedingt von Hand programmiert werden. Denn Microsoft bietet ein Aufnahmewerkzeug, das Ihre Aktionen aufzeichnet und daraus ein Makro erstellt. Dazu muss zuerst der Reiter Entwicklertools aktiviert werden. In Excel 2010 und 2013 öffnen Sie das Menü Datei/Optionen. Gehen Sie danach zur Option Menüband anpassen und setzen Sie das Häkchen beim Eintrag «Entwicklertools», **Bild 1 A**. In Excel 2007 klicken Sie auf den Office-Knopf oben links und greifen zu den Excel-Optionen. Anschliessend finden Sie unter Häufig verwendet den Eintrag «Entwicklerregisterkarte in der Multifunktionsleiste anzeigen». In beiden Fällen wird in Excel ein neuer Reiter namens Entwicklertools eingeblendet, **Bild 2 A**.

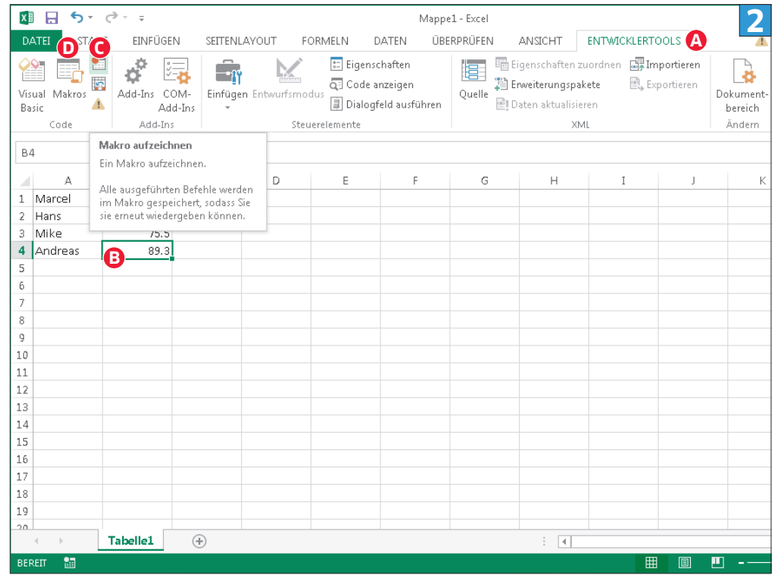
In unserem Beispiel erstellen wir ein einfaches Makro, um eine Zahl in einem Excel-Feld umzuformatieren. Als Erstes markieren Sie das gewünschte Feld **B**. Tippen Sie eine Zahl ein. Klicken Sie auf das Aufnahmesymbol **C**. Es öffnet sich ein neues Fenster, **Bild 3**.

Hier legen Sie einen Namen für das Makro fest und bestimmen falls gewünscht eine Tastenkombination, mit der sich das Makro aufrufen lässt. Sie können dabei auch zusätzlich die *Shift*-Taste drücken, falls Sie drei Tasten benutzen möchten. Oder Sie bestimmen gar kein Tastenkürzel und starten das Makro via Excel-Menü (dazu später mehr). Drücken Sie OK; die Aufnahme läuft.

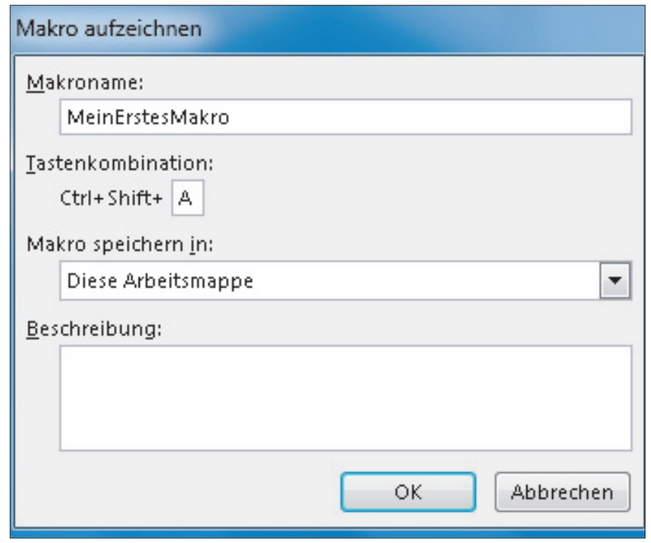
Klicken Sie jetzt auf den Reiter Start und auf das Symbol F für «fett». Wählen Sie danach im Bereich «Zahl» die Geldnote, um die Zahl in Franken umzuwandeln. Im Reiter Entwicklertools finden Sie nun anstelle des Aufnahmeknopfs einen Stopp-Knopf, um die Aufzeichnung zu beenden. Haben Sie die gewünschten Aktionen durchgeführt, klicken Sie darauf.

Markieren Sie jetzt ein zweites Excel-Feld, das ebenfalls eine Zahl beinhaltet. Klicken Sie in den Entwicklertools auf Makros, **Bild 2 D**. Nun wird Ihr Makro angezeigt, das Sie per Klick auf Ausführen starten. Das markierte Feld wird automatisch wie aufgezeichnet verarbeitet. Klicken Sie auf Bearbeiten, gelangen Sie zum Makrocode, **Bild 4**.

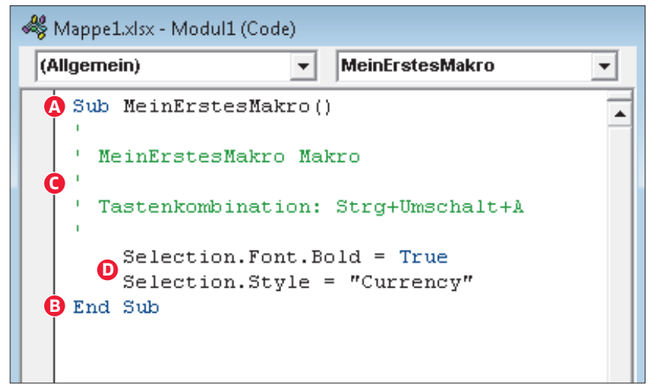
Die Zeilen *Sub MeinErstesMakro()* **A** und *End Sub* **B** stellen immer den Anfang und das Ende eines Makros dar. Die Zeilen mit einem Hochkomma am Anfang sind Kommentare **C**, die nur zur Dokumentation da sind und keine Auswirkung auf das Makro haben. Diese werden grün dargestellt, um sie vom restlichen Code zu unterscheiden. In unserem Beispiel sind lediglich zwei Zeilen relevant **D**. Die erste setzt bei der Auswahl (*selection*) die Schrift (*font*) auf fett (*bold*). Die zweite ändert den Stil (*style*) der Auswahl auf das Format «Währung» (*currency*). ➜



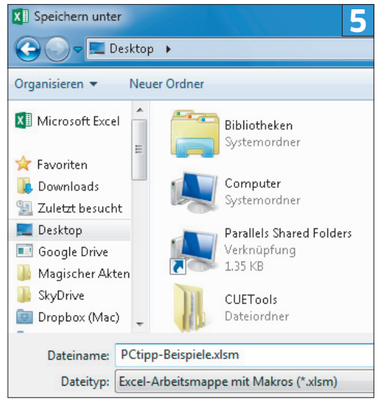

**Alle Makro-Funktionen befinden sich im Reiter ENTWICKLERTOOLS**




**Die Makros können auch mittels Tastenkombination aufgerufen werden**




**So ist ein einfaches Makro aufgebaut**




**Excel-Dateien mit Makros müssen im Format XLSM gespeichert werden**

### MAKROS SPEICHERN

Um die Excel-Datei samt Makro zu speichern, ist ein Kniff nötig. Denn aus Sicherheitsgründen dürfen herkömmliche Excel-Dateien im XLSX-Format keine Makros enthalten. Stattdessen müssen diese im Format XLSM abgelegt werden. Dieses unterscheidet sich ansonsten nicht vom normalen Excel-Format. Klicken Sie auf Datei/Speichern unter oder Office-Knopf/Speichern unter und wählen Sie als Dateityp Excel-Arbeitsmappe mit Makros (*.xlsm), **Bild 5**. Beim Öffnen solcher Dateien erscheint übrigens je nach Einstellungen eine Warnung, ob Makros aktiviert werden sollen. Das können Sie bei vertrauenswürdigen Quellen bedenkenlos bejahen.

Makros können nicht nur über das Makro-Fenster oder Tastenkombinationen gestartet werden. Wenn Sie ein Makro oft brauchen, lohnt es sich, dafür eine eigene Schaltfläche in der Tabelle anzulegen. So haben Sie es mit einem Mausklick griffbereit. Das geht ganz leicht. Klicken Sie auf Entwicklertools/Einfügen und wählen Sie Schaltfläche (Formularsteuerelement). Jetzt können Sie mit gedrückter Maustaste einen Knopf zeichnen, das gewünschte Makro auswählen und die Beschriftung ändern. Klicken Sie auf den Knopf, wird das Makro sofort ausgeführt. Mit einem Rechtsklick auf die Schaltfläche ändern Sie Ihre Einstellungen.

**Tipp:** Selbstverständlich können Sie bei einer Makro-Aufzeichnung auch mehrere Felder bearbeiten. Gehen Sie dazu wie beschrieben vor, markieren Sie ein weiteres Feld und führen Sie wiederum die gewünschten Aktionen aus. Dabei lohnt es sich meistens, die Funktion Relative Verweise verwenden zu aktivieren. Der Vorteil: Markieren Sie zum Beispiel beim Aufzeichnen das Feld A2 neben dem anfangs ausgewählten A1, merkt Excel das und nimmt bei der Ausführung auch automatisch immer das daneben liegende Feld – egal, auf welchem Feld Sie das Makro später anwenden. Andernfalls berücksichtigt das Makro immer fix das Feld A2.

## Makros für Profis

Fortgeschrittene Anwender können den Makro-Code auch direkt programmieren. So lassen sich selbst komplizierte Aufgaben automatisieren. Im Folgenden werden wir einige komplexere Makros analysieren. Die Codes dazu erhalten Sie mit Webcode **70123** auf www.pctipp.ch. Diese dürfen Sie beliebig kopieren und bearbeiten.

Im Internet finden Sie ausserdem Tausende Makros und Anleitungen, die bei der Programmierung und beim Finden von Befehlen helfen; suchen Sie einfach mithilfe von Google nach dem gewünschten Makro. Auch die beschriebene Makro-Aufnahmefunktion ist häufig ein praktisches Werkzeug, um Befehle und Eigenschaften zuerst einfach durchzuspielen und danach im erstellten Code den Ablauf und die Namen der einzelnen Befehle zu analysieren.

Für Einsteiger, die sich tiefer mit Makros beschäftigen wollen, empfiehlt sich ausserdem das Buch «Einstieg in VBA mit Excel: Für Microsoft Excel 2002 bis 2013» von Thomas Theis. Es kostet knapp 30 Franken, die ebenfalls erhältliche E-Book-Version gibts für rund 20 Franken.

### MAKRO ERSTELLEN UND SPEICHERN

Um ein eigenes Makro anzulegen, gehen Sie zu Entwicklertools/Makros, geben bei «Makroname» einen Namen ein und greifen zur Option Erstellen. Danach kann es bereits mit dem Programmieren losgehen. Das Speichern selbst programmierter Makros geht genau gleich wie unter «Makros speichern» auf dieser Seite beschrieben.

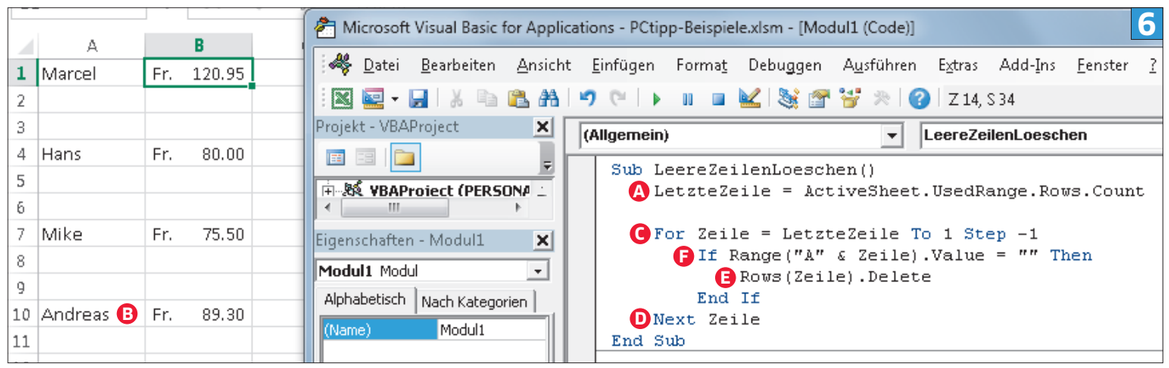
### LEERE ZEILEN LÖSCHEN

Ein häufiges Problem in Excel sind leere Zeilen, die meist in mühseliger Handarbeit entfernt werden müssen. Bei grossen Tabellen nimmt diese eigentlich einfache Aufgabe viel Zeit in Anspruch. Abhilfe schafft hier unser Makro LeereZeilenLoeschen. Es ist vielleicht nicht die eleganteste Lösung, zeigt aber sehr gut, wie man ein Makro Zeile für Zeile in einer Excel-Datei ablaufen lässt.

Bevor wir mit dem Erklären des Makrocodes loslegen, möchten wir das Konzept der Variablen erklären. Dieses ist eine wichtige Grundlage zum Erstellen von Makros. Bei Variablen handelt es sich um Gefässe, in die zum Beispiel Zahlen oder auch Zeichenketten wie Wörter gespeichert werden. Mit einem Code wie *BeispielVariable = 15* würde der Wert 15 in die Variable *BeispielVariable* abgelegt und kann später jederzeit wieder mit dem Namen *BeispielVariable* aufgerufen werden. Variablen dürfen beliebig benannt werden, aber keine Leerzeichen oder Sonderzeichen umfassen.

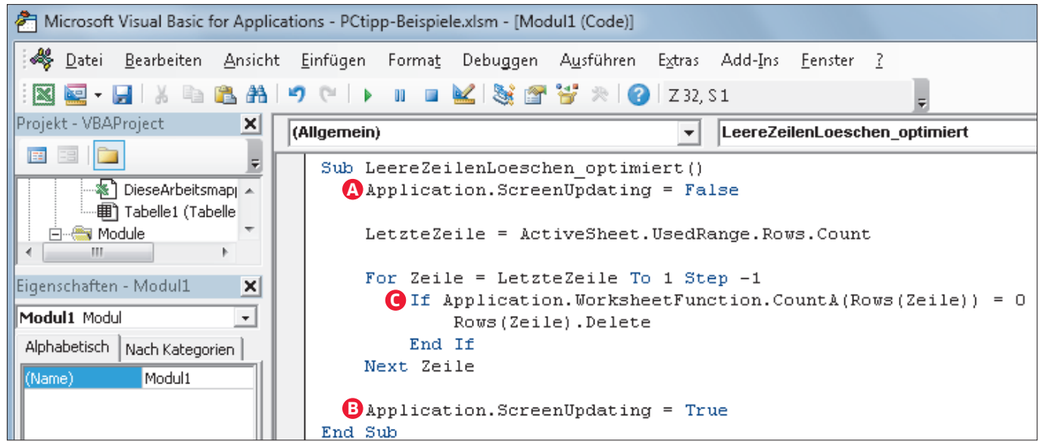
In unserem Beispiel, **Bild 6**, wird mit der Codezeile *ActiveSheet.UsedRange.Rows.Count* **A** zuerst die Anzahl Tabellenzeilen (*Rows.Count*) im aktiven Blatt (*ActiveSheet*) des genutzten Tabellenbereichs (*UsedRange*) ausgelesen und in der Variable *LetzteZeile* gespeichert. In diesem Fall ist es der Wert 10, da unter der Zeile *Andreas* **B** keine Inhalte mehr vorkommen. Sehr wichtig ist nun die Codezeile **C**. Die For-Schleife sorgt dafür, dass das Makro die einzelnen Tabellenzeilen abarbeitet. Dabei startet das Makro mit dem Wert der Variable *LetzteZeile* (also Zeile 10) und geht rückwärts bis *To 1* (zu Zeile 1). Der neue Wert wird jeweils in der Variable *Zeile* abgespeichert. Das Rückwärtszählen wird mit *Step -1* definiert (Step heisst Schritt). Sobald der Code bei *Next Zeile* **D** angelangt ist, zieht das Makro eine Tabellenzeile ab, die For-Schleife beginnt von vorne. Dies wird so oft wiederholt, bis das Makro bei Zeile 1 angelangt ist. Danach verlässt es die For-Schleife.

Ausser den For-Schleifen gehören If-Anweisungen zu den wichtigsten Elementen jeder Programmiersprache. Dabei handelt es sich um Wenn-dann-Anweisungen. In unserem Code soll die aktuell geprüfte Zeile gelöscht werden, wenn ein Feld der Spalte A leer ist. Will man zum Beispiel prüfen, ob das Feld A1 nichts enthält, verwendet man *If Range("A1").Value = "" Then*. Value heisst im Englischen so viel wie Wert und gibt den Wert des Felds A1 aus. Alle darauffolgenden Zeilen bis zum obligatorischen *End If* werden nur ausgeführt, wenn diese Anweisung erfüllt ist. Ist also der Wert einer Zeile leer, geht es weiter zu *Rows(Zeile).Delete* **E**. Das bedeutet, dass die betreffende Zeile gelöscht wird.




**Das Excel-Makro rechts entfernt die leeren Zeilen im Tabellenblatt**

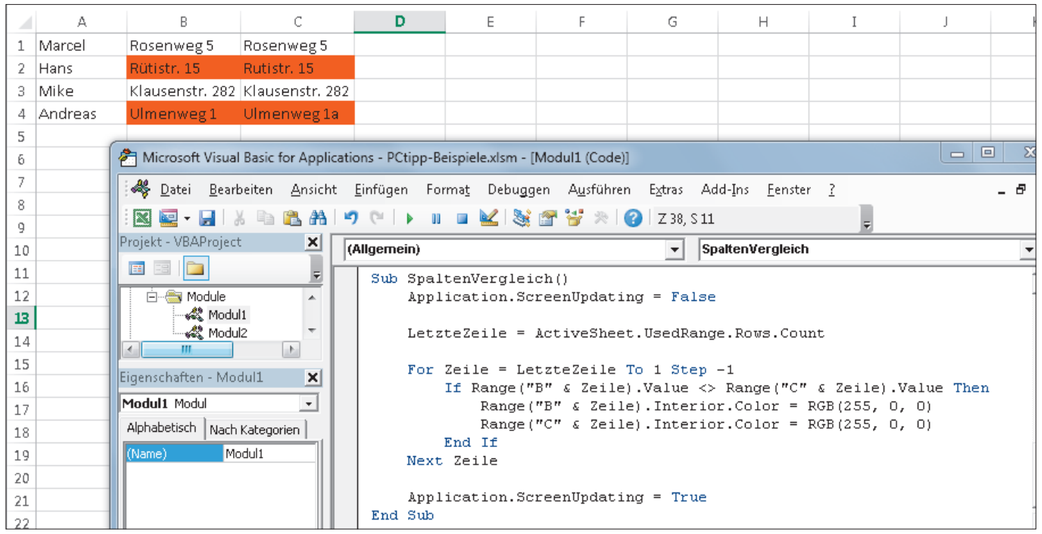





**Mit *Application.ScreenUpdating* werden Excel-Makros rascher abgearbeitet**

In unserem Beispiel soll jedoch nicht immer nur die Zelle A1 geprüft werden, sondern wir wollen die Spalte A der jeweils aktuellen Zeile kontrollieren. Deshalb verwenden wir den Makrocode *If Range("A" & Zeile).Value = "" Then* **F**, wobei das Wort *Zeile* jeweils mit dem aktuellen Variablenwert ersetzt wird. So wird der Befehl in der Zeile im ersten Durchgang zu *If Range("A10").Value = "" Then*, im zweiten zu *If Range("A9").Value = "" Then* und so weiter. Ist eine Zelle in Spalte A tatsächlich leer, wird wie erwähnt der Code *Rows(Zeile).Delete* ausgeführt. Dies ist etwa bei Zeile 9 der Fall, in welcher der ausgeführte Code der Zeile *Rows(9).Delete* entspricht, also gelöscht wird. Anschliessend wird die If-Schleife verlassen und der nächste Durchgang der For-Schleife mit Zeile 8 beginnt.

**Tipp:** Dieser einfache Code lässt sich noch optimieren, **Bild 7**. Die Codezeilen **A** und **B** sorgen beispielsweise für deutlichen Tempozuwachs bei grossen Tabellen. Normalerweise führt Excel ein Makro aus und zeigt dabei live alle Änderungen an. Das führt zu zwei Problemen: Zum einen wird das Bild sehr unruhig und springt wild herum, da das Makro viele Zeilen pro Sekunde bearbeitet. Zum anderen benötigt die Darstellung unnötig viel Rechenleistung. Mit dem Befehl *Application.ScreenUpdating = False* wird das Bild während der Verarbeitung eingefroren. *ScreenUpdating* steht für die Bildschirmaktua- ➜




| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Marcel | Rosenweg 5 | Rosenweg 5 | | | | | | | |
| 2 | Hans | Rütistr. 15 | Rutistr. 15 | | | | | | | |
| 3 | Mike | Klausenstr. 282 | Klausenstr. 282 | | | | | | | |
| 4 | Andreas | Ulmenweg 1 | Ulmenweg 1a | | | | | | | |

**8** Auch Werte, die abweichen, machen Sie mit Excel-Makros ausfindig

lisierung. Diese Eigenschaft wird zu Beginn des Makros auf *False* (falsch) gesetzt und zum Schluss wieder mit *True* (wahr) aktiviert. Die optimierte Codezeile **C** sorgt dafür, dass pro Zeile nicht nur geprüft wird, ob die Spalte A leer ist, sondern ob die ganze Zeile keine Inhalte hat.

## SPALTEN VERGLEICHEN

Das vorgestellte Grundgerüst mit der For-Schleife lässt sich auch nutzen, um Unterschiede zweier Spalten zu vergleichen, **Bild 8**. In unserem Fall wurde gegenüber dem letzten Makro nur die If-Anweisung angepasst. Wir testen nun, ob der Wert der Spalte B dem Wert der Spalte C entspricht: *If Range("B" & Zeile).Value <> Range("C" & Zeile).Value Then* (<> bedeutet ungleich). Falls nicht, wird zuerst das jeweilige Feld der Spalte B mit *Interior.Color = RGB(255,0 0)* eingefärbt und anschliessend das Feld der Spalte C.

Ein solcher Vergleich ist etwa praktisch, wenn zwischen zwei Spalten – in unserem Beispiel Adressen – kleine Unterschiede bestehen, die von Auge nur schwierig zu finden sind. Das Einfärben erleichtert das Suchen der Unterschiede.

## JEDE ZWEITE SPALTE LÖSCHEN

Ein weitverbreitetes Problem sind auch Excel-Tabellen, bei denen jede zweite Zeile gelöscht werden muss. Diese Aufgabe benötigt von Hand sehr viel Zeit, lässt sich aber mit einem Makro einfach umsetzen. Wir verwenden wieder dasselbe Grundgerüst. Geändert werden muss allerdings die For-Schleife, **Bild 9**. Nun wird der Code *For Zeile = 2 To LetzteZeile Step 1* **A** verwendet. Mit der Zahl nach *For Zeile =* wird festgelegt, mit welcher Zeile begonnen wird. In unserem Beispiel beginnen wir bei Zeile 2 und löschen diese ohne weitere Wenn-dann-Überprüfung. Im nächsten Schritt wird die Zeile um eins erhöht (*Step 1*) und wieder gelöscht. Dieser Vorgang wird wiederholt bis zur letzten Zeile mit Inhalt. Achtung: Man könnte auf die Idee kommen, dass man *Step 2* statt *Step 1* verwenden müsste, da jede zweite Zeile gelöscht werden soll. Dies ist allerdings nicht der Fall, da mit dem Löschen alle unteren Zeilen jeweils eine Zeile nach oben rutschen.

Beispiel: Wird im ersten Durchgang Zeile 2 gelöscht, befindet sich die nächste zu löschende Zeile nicht mehr auf Position 4, sondern auf der Position 3. Verwenden Sie *Step 2* anstatt *Step 1*, würde jede dritte Zeile gelöscht. Grundsätzlich gilt also, dass bei *Step* immer 1 abgezogen werden muss. Wollen Sie jede fünfte Zeile in einer Tabelle entfernen, nutzen Sie beispielsweise *Step 4*.

Noch etwas ausgeklügelter ist unser Makro *JedeXteZeileLoeschen*, das Sie ebenfalls in der Tabelle unter Webcode **70123** finden, **Bild 10**. Hier wird der Benutzer beim Ausführen des Makros gefragt, bei welcher Zeile der Löschvorgang starten soll (*StartZeile = InputBox*) **A**. Ausserdem kann der Nutzer bestimmen, ob er jede zweite, dritte, vierte etc. Zeile löschen will **B**. Die Eingaben werden in den Variablen *StartZeile* und *XteZeile* gespeichert. Ausserdem haben wir die For-Schleife so angepasst **C**, dass die Startzeile und die Schritte nicht mehr fix sind, sondern aus den Variablen ausgelesen werden.

```
Sub JedeZweiteZeileLoeschen()
    Application.ScreenUpdating = False

    LetzteZeile = ActiveSheet.UsedRange.Rows.Count

  A For Zeile = 2 To LetzteZeile Step 1
        Rows(Zeile).Delete
    Next Zeile

    Application.ScreenUpdating = True
End Sub
```

**9** Mit einem Klick jede zweite Zeile löschen

```
Sub JedeZweiteXteLoeschen()
  A StartZeile = InputBox("Geben Sie die Zeilenummer an, ab welcher jede x-te Zeile gelöscht weren soll.")
  B XteZeile = InputBox("Geben Sie jede x-te Zeile an, die gelöscht werden soll.")

    Application.ScreenUpdating = False

    LetzteZeile = ActiveSheet.UsedRange.Rows.Count

  C For Zeile = StartZeile To LetzteZeile Step XteZeile - 1
        Rows(Zeile).Delete
    Next Zeile

    Application.ScreenUpdating = True
End Sub
```

**10** Dieses Makro ist äusserst flexibel