eload24

# Excel-Programmierung mit VBA

Fachverlag für **Computerwissen**

Klicken, Lesen, Weitermachen. So einfach geht das.

| | |
|---|---|
| Rubrik | **Excel** |
| Thema | **VBA** |
| Umfang | **41 Seiten** |
| eBook | **00700** |
| Autor | **Martin Althaus** |

Auch Nicht-Programmierer können die wichtigsten Grundzüge von VBA schnell erlernen und nach kurzer Zeit eigene VBA-Makros schreiben. Diese Kenntnisse möchte Ihnen dieses eBook vermitteln.

# Excel-Programmierung mit VBA

Fachverlag für **Computerwissen**

Der Fachverlag für Computerwissen ist ein Verlagsbereich der Verlag für die Deutsche Wirtschaft AG.

**eload24 AG**
Blegistrasse 7
CH-6340 Baar

info@eload24.com
www.eload24.com



## Inhalt

# Einstieg: Excel-Programmierung mit VBA

Durch die Excel-Programmierung mit VBA können Sie hohe Effektivität erreichen. Über die integrierte Programmiersprache VBA können Sie viele Aufgaben automatisieren, für die Sie manuell sehr viel Zeit benötigen würden.

Auch Nicht-Programmierer können die wichtigsten Grundzüge von VBA schnell erlernen und nach kurzer Zeit eigene VBA-Makros schreiben. Diese Kenntnisse möchte Ihnen dieses eBook vermitteln. Viel Erfolg bei Ihrer Programmierung!


Alle Programmbeispiele und Praxislösungen können Sie unabhängig von Ihrer Excel-Version einsetzen. Sie können sowohl Excel 97, Excel 2000, Excel XP, Excel 2003 als auch Excel 2007 verwenden. Von Ihrer Windows-Version ist der Einsatz ebenfalls unabhängig: Windows wird ab Version 98 unterstützt.

## Nutzen Sie Fertiglösungen für Ihre Excel-Praxis

Sie können die Tipps und Tricks dieses eBooks direkt in passenden Beispieldateien nachvollziehen und ausprobieren. Wenn Sie die Formeln und Funktionen direkt in Ihren Kalkulationen nutzen möchten, ist dieser Weg besonders praktisch. Dann kopieren Sie die benötigten Formeln einfach direkt in Ihre eigene Kalkulation.

Um die Beispieldateien zu diesem eBook herunterzuladen, klicken Sie auf den folgenden Downloadlink: www.eload24.com/downloads/00700_Dateien.zip. Entpacken Sie die Zipdateien, indem Sie sie doppelt anklicken, und legen Sie einen Pfad fest, in dem die Arbeitsmappen gespeichert werden sollen.

- Öffnen Sie andere Arbeitsmappen direkt aus Ihren VBA-Programmen und setzen Sie dazu ein universelles Unterprogramm ein – die Beispielmakros in der Arbeitsmappe *OeffnenVBA.xls* zeigen, wie das geht.
- Durchlaufen Sie in Ihren Makros die Zellen eines markierten Bereichs einzeln Schritt für Schritt, um Berechnungen durchzuführen oder Formatierungen vorzunehmen – das Praxisbeispiel *Matrix-VBA.xls* zeigt, wie das funktioniert.

# VBA: Die Grundlagen

Mit VBA haben Sie alle Verfahren und Techniken an der Hand, um Ihrem Excel eine eigene Oberfläche zu geben – neue Kommandos, neue Befehle, eigene Dialogfenster und vieles mehr.

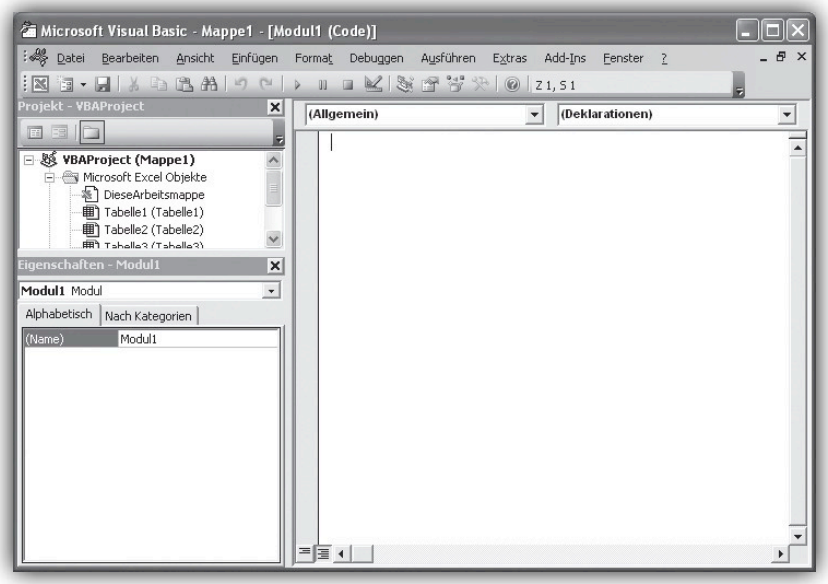
Ihr Vorteil: Im Gegensatz zur Programmierung eigener Lösungen mit einer traditionellen Programmiersprache können Sie beim Einsatz von VBA alle in Excel verfügbaren Funktionen direkt in Ihren Programmen nutzen. Sie müssen also beispielsweise nicht erst aufwändige Berechnungsroutinen selbst programmieren, sondern können direkt auf die eingebundenen Routinen von Excel zugreifen.

Das spart Ihnen Entwicklungsaufwand – und damit Geld.

Ein weiterer Vorteil: Während sich eigenständige Programmiersprachen nur von Programmierern und EDV-Profis bedienen lassen, kann VBA auch von Anwendern eingesetzt werden, die über keine Programmiererfahrung verfügen.

## Was ist ein VBA-Programm?

VBA-Programme setzen sich aus einzelnen Anweisungen zusammen. Die Basis für Ihre VBA-Programme bildet der Einsatz von Objekten, Methoden, Eigenschaften und Anweisungen. Diese Bestandteile Ihres Quelltextes binden Sie in die Programme, Unterprogramme und Funktionen ein. Die einzelnen Komponenten Ihrer VBA-Programme werden in einem Quelltext zusammengefasst. Den geben Sie wie einen Text direkt in ein besonderes VBA-Fenster von Excel ein.




Die wichtigsten Schlüsselwörter für das Formulieren eines Programms sind *Sub* und *End Sub*. Die Sub-Anweisung deklariert den Start eines VBA-Programms, mit dem End-Sub-Befehl definieren Sie das Ende eines Unterprogramms.

Hinter dem Schlüsselwort *Sub* platzieren Sie den Namen, den Sie dem Unterprogramm geben möchten. Dieser Name muss mit einem Buchstaben beginnen und darf keine Leerzeichen enthalten. Der Unterstrich ist als Trennzeichen erlaubt.

## Groß- und Kleinschreibung wird von Excel nicht beachtet

Sie müssen beim Schreiben von VBA-Quelltext die Groß- und Kleinschreibung nicht beachten. Wenn Sie einen Programmtext eintippen, erkennt Excel das automatisch, sobald Sie die Eingabetaste drücken. Dann überprüft Excel sofort die Syntax Ihres Programms.

Sehen Sie sich die folgende Zeile an:

```
sub meinprogramm
```

Nach dem Eintippen stellt Excel die Zeile automatisch so dar:

```
Sub meinprogramm()
```

Excel hat das Schlüsselwort *Sub* erkannt und den Text in Quellcode übersetzt. Hinter dem Unterprogrammnamen platziert Excel automatisch ein Klammerpaar – damit es als Unterprogramm kenntlich wird. Sie erkennen das auch daran, dass Excel das Schlüsselwort *Sub* in anderer Farbe darstellt.

## VBA-Schlüsselwörter

Dieses Prinzip gilt für alle VBA-Zeilen: Blaue Wörter sind Schlüsselwörter von Excel. Diese Wörter dürfen Sie auf keinen Fall als Namen für Ihre Unterprogramme oder Variablen verwenden.

Wenn Sie eine Sub-Anweisung in einem VBA-Quelltext eingerichtet haben, fügt Excel den Namen des Unterprogramms automatisch in die interne Makroliste ein. Sie können das Programm dann über die Funktion *Makro* aus dem Menü *Extras* ausführen.

## Objekte, Methoden und Eigenschaften einsetzen

Mit einem Objekt werden alle Daten und Parameter eines Elements zusammengefasst. Beispielsweise gibt es ein Objekt *Cell*, mit dem Excel Zellen verwaltet. Die Parameter eines Objekts nennt man Eigenschaften. Diese Eigenschaften können Sie in Ihren VBA-Programmen auslesen oder ändern.

So verfügt das Objekt *Cell* beispielsweise über Eigenschaften wie *Value* (der aktuelle numerische Inhalt der Zelle), *Row* (die Zeilennummer der Zelle), *Text* (der Inhalt der Zelle, wie Excel ihn auf dem Bildschirm darstellt) und viele mehr.

## Excel-Objekt-Hierarchie verwenden

Excel-Objekte sind in Hierarchien angeordnet. Es gibt Tausende von Zellen, die auf un-

terschiedlichen Arbeitsblättern und in unterschiedlichen Arbeitsmappen zu finden sind. Das Objekt *Cell* ist daher ein Unterobjekt von *Sheet* (also Tabellenblatt), *Sheet* wiederum ist ein Unterobjekt von *Workbook* (also Arbeitsmappe).

Zu vielen Objekten stellt Excel Anweisungen zur Verfügung, die auf diese Objekte abgestimmt sind. Diese Anweisungen können Sie nur in Kombination mit einem bestimmten Objekt einer bestimmten Objektart aufrufen. Man nennt diese Anweisungen *Methoden*. Eine Methode ist ein Unterprogramm, das an ein bestimmtes Objekt geknüpft ist.

Beispielsweise gibt es die Methode *Activate*, mit der ein Objekt aktiviert werden kann. Dabei kann es sich um ein Anwendungsobjekt, ein Tabellenblattobjekt oder eine Reihe anderer Objekte handeln.

### Wie Sie die richtige Schreibweise einhalten

Wenn Sie beispielsweise dafür sorgen möchten, dass eine bestimmte Seite in einer bestimmten Arbeitsmappe aktiviert wird, setzen Sie das unten folgende Kommando ein. Sie trennen den Objektnamen und die Methode durch einen Punkt.

Ein Beispiel für den Aufruf einer Methode in Kombination mit einem Objekt ist dieser Befehl:

```
Workbooks(„dreh.xls“).
Sheets(„Eingabe“).Activate
```

Die Anweisung sorgt dafür, dass das Tabellenblatt *Eingabe* aus der Arbeitsmappe *DREH.XLS* aktiviert wird. Die Methode *Workbooks* gibt dazu ein Objekt vom Typ *Workbook* – also eine Arbeitsmappe – zurück. Zu diesem

Objekt wird über die Methode *Sheets* ein Unterobjekt vom Typ *Sheet* – Tabellenblatt – ausgewählt, das mit der Methode *Activate* eingeschaltet wird.

### Zugriff auf das aktuelle Objekt erhalten

Grundsätzlich gilt: Falls nicht anders vereinbart, bezieht Excel Ihre Anweisungen auf das aktuelle Objekt.

Wenn beim Ausführen der obigen Programmzeile beispielsweise die Arbeitsmappe *DREH.XLS* schon aktiv ist, können Sie sich die Auswahl des entsprechenden Objekts über die Sheets-Methode sparen. Dann heißt die Anweisung schlicht:

```
Sheets("Eingabe").Activate
```

## Arbeitsmappe aus einem VBA-Programm heraus öffnen

*VBA-Anwendung: Arbeitsmappen öffnen.*

Es ist ganz einfach, das Öffnen einer anderen Arbeitsmappe aus einem VBA-Programm heraus zu erledigen. Wenn Sie den Pfad der Arbeitsmappe kennen, reicht ein einziger VBA-Befehl aus, um die Datei zu öffnen. Aber Vorsicht: Standardmäßig werden die Auto-Open-Makros in einer solchen Arbeitsmappe nicht ausgeführt. Diese führt Excel nur aus, wenn Sie eine Arbeitsmappe manuell öffnen.

### Das Kommando *Open*

Daher müssen Sie Excel separat anweisen, die Auto-Open-Makros auszuführen. Hier ist der Quelltext, den Sie verwenden können, um Arbeitsmappen aus Ihren VBA-Programmen heraus zu öffnen:

```
Private Sub MappeOeffnen(Wname, Pfad
As String)
Workbooks.Open FileName:=Pfad + "\"
+ Wname
Workbooks(Wname).RunAutoMacros
xlAutoOpen
End Sub
```

Mit diesem Makro-Unterprogramm ist es für Ihre VBA-Programme ganz einfach, eine Arbeitsmappe von Diskette, Festplatte, CD-ROM oder einem anderen Datenträger zu öffnen.

### Mappe öffnen inklusive Startmakros

Rufen Sie das Unterprogramm *MappeOeffnen* auf und übergeben Sie dem Unterprogramm zwei Parameter: Den Namen der Datei (mit der Endung „.xls“) sowie einen kompletten Pfad (ohne abschließenden Backslash „\“). Das folgende Beispiel zeigt, wie der Aufruf aussehen muss, wenn Sie die Arbeitsmappe *MEINE.XLS* aus dem Pfad *D:\ARBEIT\PROJEKTE* aufrufen möchten:

```
MappeOeffnen("MEINE.XLS",
"D:\ARBEIT\PROJEKTE")
```

Natürlich können Sie mit der Funktion auch Mappen in Ihrem Netzwerk öffnen. Sorgen Sie in diesem Fall dafür, dass dem Pfad die Netzwerkangabe vorangestellt wird – also mit einem Doppel-Backslash „\\“ vor dem Namen der Ressource. Sie können also über das ganz oben definierte Unterprogramm *MappeOeffnen* mit nur einem Kommando in Ihren VBA-Lösungen andere Arbeitsmappen öffnen.


Denken Sie daran, den Programm-Code für das Makro *MappeOeffnen* (von *Sub* bis *End Sub*) in Ihre Lösung zu kopieren, damit Sie das Kommando aufrufen können.

### Mappen speichern

Wenn Sie an geöffneten Arbeitsmappen Veränderungen vornehmen, können Sie die Mappen speichern, indem Sie die Save-Methode einsetzen. Um die zuvor geöffnete Arbeitsmappe *MEINE.XLS* zu schließen, verwenden Sie die folgenden Kommandos:

```
Sub MappeSpeichern
Workbooks("Meine.xls").Save
End Sub
```

Auf ähnliche Weise schließen Sie in Ihren Programmen Arbeitsmappen, auf die Sie nicht mehr zugreifen möchten. Das passende Kommando lautet *Close*. Zum Schließen der Mappe *MEINE.XLS* setzen Sie die folgenden Kommandos ein:

```
Sub MappeSchliessen
Workbooks("Meine.xls").Close
End Sub
```

## Schritt für Schritt: Variablen einsetzen

*Variablen – Platzhalter für Ihre Daten.*

Variablen sind ein wichtiger Bestandteil aller Programme. Das ist auch in VBA nicht anders. Unter einer Variablen versteht man einen Speicherplatz für eine Zahl, einen Text oder andere Daten. Sie können sich eine Variable wie eine Schublade vorstellen, auf der ein Name steht.

Um beispielsweise eine Zahl vom Anwender einzulesen und das Zehnfache der Zahl auf dem Bildschirm auszugeben, liest ein VBA-Programm die Zahl von der Tastatur ein und speichert sie in einer Variablen. Im nächsten Schritt wird in einer weiteren Variablen das Zehnfache der Zahl gebildet und gespeichert. Um die Zahl schließlich auf dem Bildschirm

auszugeben, verwendet das Kommando wiederum den Variablennamen.

### Variablen bringen Flexibilität

Der Vorteil beim Einsatz von Variablen liegt in der hohen Flexibilität. Der Befehl zur Datenausgabe auf dem Bildschirm gibt immer den Inhalt der Variablen auf dem Bildschirm aus, egal welcher Wert in der Speicherstelle der Variablen zu finden ist.

Bei jeder einzelnen Zelle eines Arbeitsblatts handelt es sich im Prinzip um eine Variable. Der Variablenname entspricht der Adresse einer Zelle in Ihrem Arbeitsblatt.

Angenommen, Sie benötigen eine Variable zur Speicherung einer Zahl. Die Variable möchten Sie gern *Zahl* nennen. Die Variable soll als Inhalt den Wert 100 erhalten. In Ihrem Programm können Sie die Variable mit dem folgenden Kommando definieren: *Zahl=100*.

### Bei Variablen ist nicht nur Englisch erlaubt

Die Benennung von Variablen unterliegt einigen Beschränkungen. Im Gegensatz zu den Befehlsnamen und den Namen von Objekten, Methoden und Eigenschaften können Variablennamen aber deutsche, englische oder sonstige Begriffe sein. Der Name einer Variablen ist für Excel nichts weiter als eine symbolische Bezeichnung. Achten Sie aber auf Folgendes, wenn Sie Variablen festlegen:

- **Regeln für die Festlegung von Variablen**
  Das erste Zeichen muss ein Buchstabe oder der Unterstrich „_" sein; als weitere Zeichen können Buchstaben, Ziffern, Punkte und Unterstriche folgen.

- **Leerzeichen sind verboten**
  Leerzeichen können nicht verwendet werden. Setzen Sie stattdessen den Unterstrich oder Punkt ein, um einzelne Wörter Ihrer Variablennamen zu trennen.
- **Groß- und Kleinschreibung wird nicht beachtet**
  Buchstaben dürfen in Groß- oder in Kleinschreibung verwendet werden. Excel unterscheidet nicht zwischen den beiden Schreibweisen. Die Variable mit der Bezeichnung *EinWert* ist daher identisch mit der Variablen *einwert*.
- **Maximale Variablenlängen beachten**
  Ein Variablenname darf maximal 255 Zeichen lang sein. Es ist sinnvoll, ausführliche Namen für Variablen zu verwenden. Sie sollten die Namen so wählen, dass ihre Aufgabe im Programm hinreichend bezeichnet wird.

## So verwenden Sie die richtigen Datentypen

Wie Sie durch den Umgang mit Excel-Tabellen wissen, ist das Programm in der Lage, mit den verschiedensten Formen von Eingaben zu arbeiten. Sie können Text, Zahlen, Bezüge auf andere Zellen, Formeln oder Kombinationen aus allem in Zellen eintragen.

## VBA liefert eine ganze Palette von Datentypen

Man bezeichnet die unterschiedlichen Arten der Daten, die ein Programm verarbeiten kann, als Datentypen. Auch in VBA-Programmen unterscheidet Excel eine Reihe verschiedener Datentypen.

Die folgende Tabelle fasst sie zusammen:

| Datentyp | Bedeutung |
|---|---|
| *Integer* | Ganzzahl |
| *Long* | Lange Ganzzahl |
| *Single* | Einfach genaue Fließkommazahl |
| *Double* | Doppelt genaue Fließkommazahl |
| *Currency* | Zahlen im Währungsformat |
| *String* | Besteht aus Buchstaben und Ziffern und kann nicht in Berechnungen verwendet werden |
| *Boolean* | Kann nur „wahr" oder „falsch" annehmen |
| *Date* | Variable zum Speichern von Datum und Zeit |
| *Object* | Zeiger auf Objekte |
| *Variant* | Universeller Datentyp |

*Die Datentypen von Excel-VBA.*

Standardmäßig legt Excel Variablen vom Typ *Variant* an, wenn Sie in einer Prozedur einen Variablennamen anlegen. Alle Variablen, die Sie nicht gesondert deklarieren, erhalten den Datentyp *Variant*.

### Objektvariablen verwenden

Besonders leistungsfähige Programme, die noch dazu komfortabel zu programmieren sind, können Sie gestalten, wenn Sie mit Objektvariablen arbeiten.

Eine Objektvariable ist eine Variable, mit der Sie nicht den Inhalt eines Objekts ansprechen, sondern das Objekt selbst. Sie können dann auf den Namen der Objektvariablen alle Methoden und Eigenschaften anwenden, die Excel für das passende Objekt vorsieht.

## Das Set-Kommando

Objektvariablen legen Sie über den Set-Befehl an. Hinter dieses Kommando setzen Sie den Namen der Objektvariablen, dann ein Gleichheitszeichen und anschließend das Objekt, mit dem Sie die Variable gleichsetzen möchten.

Wenn Sie beispielsweise die Zelle A1 in der aktiven Tabelle der aktuellen Arbeitsmappe als Objektvariable *MeineZelle* ansprechen möchten, verwenden Sie die folgenden VBA-Befehle:

```
Dim MeineZelle As Range
Set MeineZelle=ActiveWorkbook.
ActiveSheet.Range("A1")
```

Anschließend können Sie mit der Variablen *MeineZelle* immer das komplette Objekt ansprechen, das Sie zuvor mit dem Set-Kommando definiert haben. Um beispielsweise den Ausdruck „=A2+A3" als Formel in die Zelle einzutragen, setzen Sie das folgende Kommando ein:

```
MeineZelle.Formula="=A2+A3"
```

Besonders praktisch werden Objektvariablen dann, wenn Sie damit auf Objekte zugreifen, die Sie erst mit Ihrem VBA-Programm erzeugen – zum Beispiel eine neue, leere Arbeitsmappe. Das geht so:

```
Dim NeueMappe As Workbook
Set NeueMappe = Workbooks.Add
```

Über den Add-Befehl legt die zweite Programmzeile eine neue, leere Arbeitsmappe an. Das passende Objekt wird über das Set-Kommando direkt der Objektvariablen *NeueMappe* zugewiesen.

Die Objektvariable *NeueMappe* können Sie nun einsetzen, um direkt auf die neue Ar-

beitsmappe zuzugreifen. Wenn Sie also beispielsweise in Zelle A1 des ersten Tabellenblatts der neuen Mappe den Wert 100 eintragen möchten, verwenden Sie das folgende Kommando:

```
NeueMappe.Worksheets(1).Range("A1").
Formula=100
```

## Arbeitsgänge mit VBA automatisieren

*Praxislösung: Makros aufzeichnen.*

Bei Ihrer Arbeit mit Excel wird es immer wieder vorkommen, dass Sie bestimmte Arbeitsgänge häufiger durchführen müssen. Das kann zum Beispiel das Einrichten eines bestimmten Seitenlayouts mit anschließendem Druck oder auch das Eingeben beziehungsweise Verschieben bestimmter Wertekombinationen und Funktionen sein.

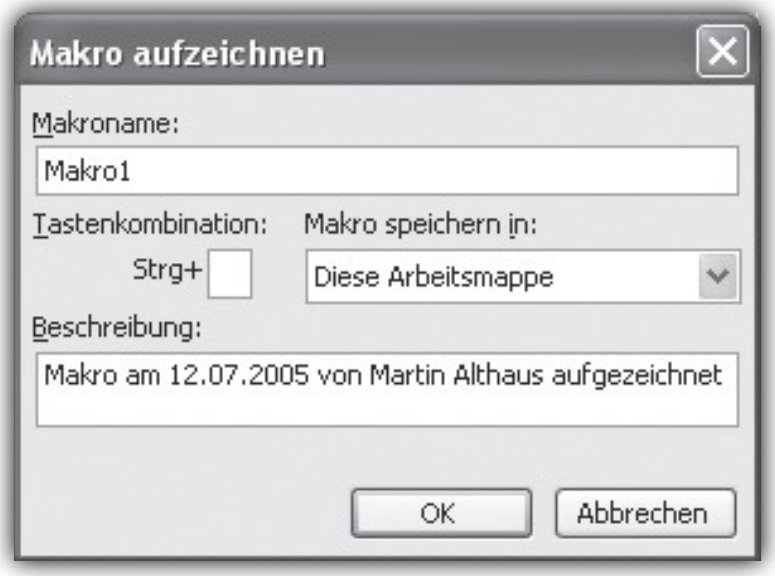
Damit Sie für solche häufig wiederkehrenden Arbeitsgänge keine unnötige Zeit verschwenden, können Sie sie über ein aufgezeichnetes Makro automatisieren und anschließend schnell und bequem auf Wunsch wiederholen, ohne alle erforderlichen Arbeitsschritte manuell durchführen zu müssen.

## Makros mit Tastenkombinationen und Symbolen verknüpfen

Zum schnellen Zugriff auf die aufgezeichneten Arbeitsgänge können Sie die dazugehörigen Makros mit Symbolen verknüpfen, den Makros eine Tastenkombination zuweisen oder die Makros mit einem Menübefehl verbinden.

Das Praktische am sogenannten Makro-Rekorder: Sie können auch ohne Programmierkenntnisse Ihre eigenen VBA-Makros erstellen. Ein weiterer positiver Nebeneffekt ist, dass Sie beim Überarbeiten Ihrer aufgezeichneten Makros einen Einblick in die Art und Weise der Makroprogrammierung erhalten.

Wenn Sie genau wissen, wie der Arbeitsgang aussieht, den Sie aufzeichnen möchten, können Sie die Makroaufzeichnung beginnen, indem Sie die Funktion *Extras* | *Makro* | *Aufzeichnen* aktivieren. Excel öffnet daraufhin das Dialogfenster *Neues Makro aufzeichnen*.




In Excel 95 finden Sie die Funktion *Aufzeichnen* im Untermenü *Extras* | *Makro aufzeichnen*. Das Dialogfenster sieht etwas anders aus. Indem Sie aber die Schaltfläche *Optionen* betätigen, erhalten Sie die Einstellungsmöglichkeiten, die Ihnen auch ab Excel 97 geboten werden.

### Speicherort festlegen

Bevor Sie mit der Aufzeichnung beginnen, müssen Sie einen Namen für das aufzuzeichnende Makro sowie einige Optionen festlegen. Für unser Beispiel klicken Sie das Feld *Makroname* an und geben den Namen *Schwarzweißdruck* ein. In der Auswahlliste *Makro speichern in* (ab Excel 97) beziehungsweise der Optionsgruppe *Speichern in* (Excel 95) legen Sie fest, wo das aufgezeichnete Makro gespeichert werden soll.

Ihnen stehen die folgenden drei Optionen zur Verfügung:

- *Persönliche Makro-Arbeitsmappe*
- *Diese Arbeitsmappe*
- *Neue Arbeitsmappe*

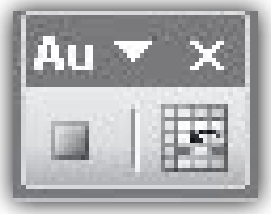
Für Makros, die Sie bei Ihrer Arbeit mit Excel immer zur Verfügung haben möchten, sollten Sie die Option *Persönliche Makro-Arbeitsmappe* wählen. Egal mit welcher Mappe Sie dann arbeiten, Ihnen steht das Makro immer zur Verfügung.

Manche Makros sind nur für eine bestimmte Arbeitsmappe interessant. Um Ihre persönliche Makro-Arbeitsmappe nicht mit solchen Makros zu belasten, sollten Sie diese Makros in der entsprechenden Mappe speichern. Dazu verwenden Sie die Option *Diese Arbeitsmappe*. Das Makro wird dann in der während der Aufzeichnung aktiven Mappe gespeichert.

### Tastenkombination festlegen

Zum schnelleren Zugriff auf Ihre als Makro aufgezeichneten Arbeitsgänge haben Sie die Möglichkeit, vor dem Aufzeichnen eine Tastenkombination festzulegen, mit der Sie das Makro anschließend schnell starten können. Sie können dabei eine beliebige Taste defi-

nieren, über die Sie das Makro in Verbindung mit der Taste [Strg] starten können. Geben Sie die gewünschte Taste in das Eingabefeld *Strg-* bzw. *Strg+* (Excel 95) ein. Mit der dritten Option *Neue Arbeitsmappe* erreichen Sie, dass nach dem Aufzeichnen des Makros eine neue Mappe angelegt wird, in die das Makro eingebunden wird.



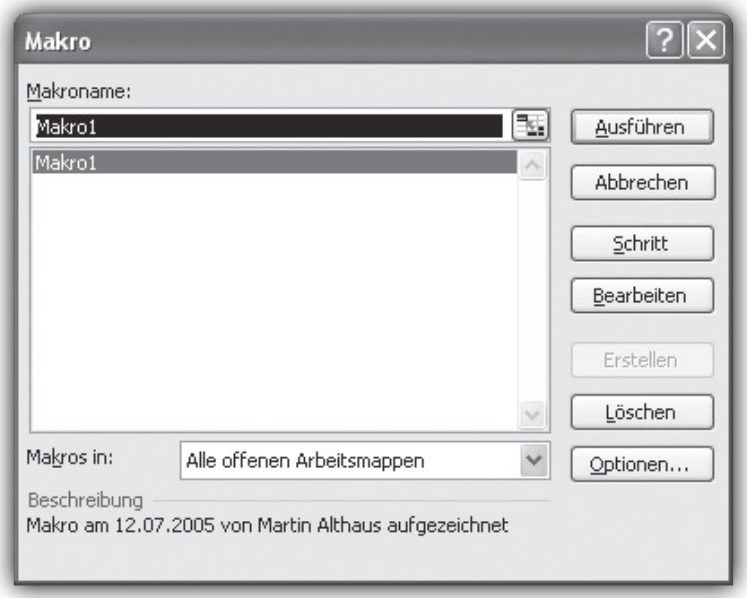
Nachdem Sie alle gewünschten Optionen festgelegt haben, betätigen Sie die Schaltfläche *OK*, um mit der Aufzeichnung der gewünschten Arbeitsschritte zu beginnen. Das Dialogfenster wird daraufhin wieder geschlossen, und eine kleine Symbolleiste mit einem Symbol wird eingeblendet.

Führen Sie nun die gewünschten Arbeitsschritte aus, die Sie aufzeichnen möchten.

Anschließend beenden Sie die Aufzeichnung über das Symbol *Aufzeichnung beenden* in der neu eingeblendeten Symbolleiste oder Sie nutzen die Funktion *Extras* | *Makro* | *Aufzeichnung beenden*. In Excel ab Version 98 heißt die Funktion zum Beenden *Extras* | *Makro aufzeichnen* | *Aufzeichnung beenden*.

### Makro starten

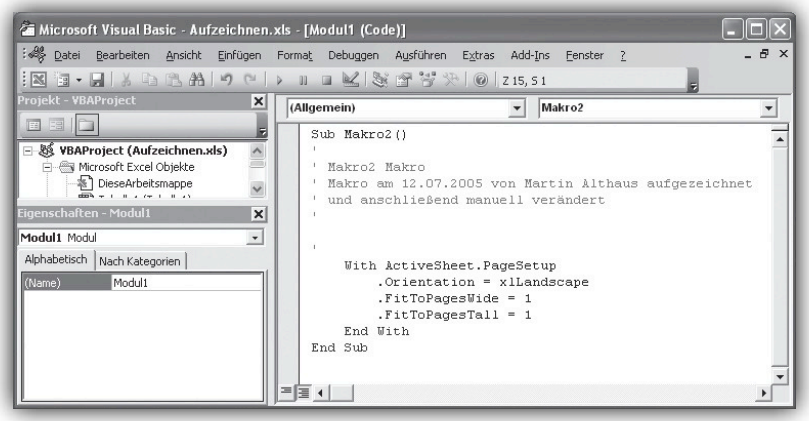
Falls Sie ein aufgezeichnetes Makro starten möchten, können Sie entweder die Tastenkombination benutzen (falls Sie vor der Aufzeichnung eine festgelegt haben) oder Sie verwenden den Menübefehl *Extras* | *Makro* | *Makros*. Excel-95-Anwender finden den Befehl *Makros* direkt im Menü *Extras*. Im erscheinenden Dialogfenster *Makro* wählen Sie das Makro in der Auswahlliste *Makroname* aus und betätigen die Schaltfläche *Ausführen*, um es zu starten.




## Aufgezeichnete Makros ansehen

Die aufgezeichneten Tätigkeiten werden von Excel automatisch in VBA-Befehle umgesetzt und in dem entsprechenden Makro (also einem VBA-Unterprogramm) gespeichert. Wenn Sie sich anschauen möchten, wie Excel Ihre Tätigkeiten in VBA-Befehle umgesetzt hat, rufen Sie das Kommando *Extras* | *Makro* | *Makros* auf und wählen das gewünschte Makro in der Liste *Makroname* aus. Klicken Sie anschließend auf die Schaltfläche *Bearbeiten*.

Jetzt zeigt Ihnen Excel das Makro im VBA-Editor an. In dem Quelltext können Sie nun Veränderung vornehmen und die VBA-Befehle Ihren Wünsche anpassen. Wenn Sie diesen Vorgang beendet haben, wählen Sie die Funktion *Speichern und zurück zu Microsoft Excel* aus dem *Datei*-Menü. Excel schließt dann den VBA-Editor und führt Sie zurück in die gewohnte Excel-Oberfläche.



# Zellen und Zellinhalte mit VBA verarbeiten

*Schritt für Schritt: Zellen ansprechen.*

Eine der wichtigsten Aufgaben von VBA-Programmen besteht darin, die Inhalte von Zellen einer Arbeitsmappe zu verarbeiten oder auf ihrer Grundlage Berechnungen vorzunehmen. Das kann bedeuten, bestehende Daten zu konvertieren, zu verteilen oder damit neue Berechnungen durchzuführen.

Eigentlich ist es ganz einfach: Um eine Zelle einer Tabelle in einem Makro zu adressieren, müssen Sie Excel drei Dinge mitteilen:

1. Welche Arbeitsmappe Sie ansprechen möchten,
2. welche Tabelle in der Arbeitsmappe Sie adressieren,
3. auf welche Zelle Sie zugreifen möchten.

## Kombinieren Sie Objekte und Methoden

Für Schritt 1 setzen Sie die Methode *Workbooks* ein. Sie liefert als Ergebnis ein Objekt vom Typ *Workbook*. Auf dieses Objekt wenden Sie die Methode *Sheets* an. Als Ergebnis erhalten Sie ein Objekt vom Typ *Sheet*. Das kombinieren Sie mit einer Methode, die Ihnen als Ergebnis eine Zelle liefert. Besonders geeignet: die Range-Methode. Das Ganze erledigen Sie mit einem einzigen Befehl, den Sie überall in Ihren Programmen einsetzen können:

```
Workbooks("kalk.xls").
Sheets("Tabelle2").Range("A2")
```

Setzen Sie anstelle von *KALK.XLS* einfach den Namen der geöffneten Datei ein, auf die Sie zugreifen möchten. Anstelle des Textes *Tabelle2* verwenden Sie den Namen der Tabelle, die Sie ansprechen möchten. Damit ist der Name gemeint, den Sie auf dem Blatt-

register finden. Schließlich setzen Sie statt der Zellbezeichnung *A2* die Adresse der Zelle ein, die Sie in Ihrem Makro verarbeiten möchten.

### Automatisch die aktuelle Zelle verwenden

Übrigens: Sie können sich oft einige der Angaben sparen. Wenn das Makro automatisch die Zelle verwenden soll, die sich in der gerade aktiven Arbeitsmappe befindet, lassen Sie die Workbooks-Methode einfach weg und schreiben nur:

```
Sheets("Tabelle2").Range("A2")
```

Ähnliches gilt für die Angabe der Tabelle. Falls das Makro immer die Zelle verarbeiten soll, die gerade auf dem Bildschirm sichtbar ist (also die im aktiven Tabellenblatt), schreiben Sie einfach nur:

```
Range("A2")
```

Dieses Kommando adressiert immer Zelle A2 im aktuell aktiven Tabellenblatt der aktuell aktiven Arbeitsmappe. Aktiv ist immer die Arbeitsmappe, die Sie gerade auf dem Bildschirm sehen und bearbeiten können.

### Welche Inhaltseigenschaften brauchen Sie?

Die Zelladresse allein reicht noch nicht ganz aus, wenn Sie auf die Inhalte einer Zelle zugreifen möchten. Inhalt ist für Excel nämlich nicht das, was Sie darunter verstehen. Excel unterteilt den Inhalt einer Zelle sehr genau. Vereinfacht gesagt: Je nachdem, ob in der Zelle ein Text, eine Formel oder ein Wert steht, ist das unterschiedlich gespeichert. Das ist auch sinnvoll so: In Ihren VBA-Makros können Sie also beispielsweise sowohl auf die Ergebnisse einer Formel in einer Zelle zugreifen als auch auf den entsprechenden Text der Formel.

### Zugriff auf die aktuelle Markierung

Wenn Sie in Excel Zellen verarbeiten, können Sie über eine Markierung mehrere Zellen gleichzeitig zur Bearbeitung auswählen. Das ist ein Verfahren, das Sie auch in Ihren Makros einsetzen sollten, wenn Sie komfortable Lösungen formatieren möchten.

Der Schlüssel zum Zugriff auf die aktuelle Markierung ist die Methode *Selection* – sie liefert Ihnen ein Objekt vom Typ *Range*, auf das Sie zugreifen können wie auf andere Bereiche. Der Ausdruck

```
Range("A1").Formula
```

liefert Ihnen den aktuellen Formeltext der Zelle A1. Wenn Sie aber die Eigenschaft *Formula* durch *Value* ersetzen, dann enthalten Sie den Wert der Zelle.

Neben diesen beiden Methoden können Sie noch eine Reihe anderer Eigenschaften einsetzen, um auf die Inhalte Ihrer Zellen zuzugreifen.

Die Tabelle auf der nächsten Seite fasst die Eigenschaften zusammen, mit denen Sie auf die Inhalte der Zellen zugreifen können.

| Methode | Bedeutung |
|---|---|
| *Formula* | Formel einer Zelle |
| *Text* | Text einer Zelle |
| *Value* | Wert einer Zelle |
| *Z1S1Formula* | Formel einer Zelle im Z1S1-Format |
| *Numberformat* | Zahlenformat einer Zelle |
| *Font.Name* | Schriftart der Zelle |
| *Font.Style* | Schriftstil der Zelle |
| *Font.Size* | Schriftgröße der Zelle |
| *Font.Underline* | Wahr für durchgestrichenen Text |
| *Font.Superscript* | Wahr für hochgestellten Text |
| *Font.Subscript* | Wahr für tiefgestellten Text |
| *Font.Bold* | Wahr für fett gedruckten Text |
| *Font.Shadow* | Wahr für Text mit Schatten |
| *Font.Underline* | Art der Textunterstreichung |
| *Font.Colorindex* | Farbcode des Textes |
| *HorizontalAlignment* | Art der horizontalen Ausrichtung |
| *VerticalAlignment* | Art der vertikalen Ausrichtung |
| *WrapText* | Wahr für Zeilenumbruch |
| *Interior.Colorindex* | Farbcode für Zellfarbe |
| *Interior.Pattern* | Art des Musters |

## Makro mit Symbol verknüpfen

*Praxislösung: Symbole für Makros.*

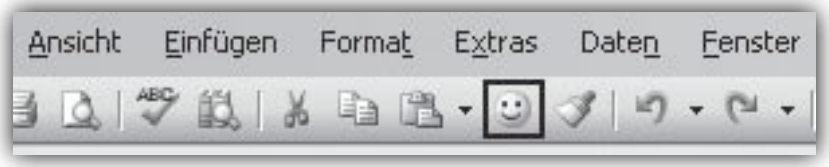
Anstatt einem aufgezeichneten Makro eine Tastenkombination zuzuweisen, können Sie Makros auch mit neuen Symbolen in einer beliebigen Symbolleiste verknüpfen. Dadurch sparen Sie Zeit beim Start des Makros.

Es steht Ihnen frei, in welche Symbolleiste Sie das neue Symbol einbinden. Sie sollten allerdings eine Symbolleiste wählen, die Sie bei Ihrer Arbeit mit Excel normalerweise eingeblendet haben, damit Sie auf Ihre aufgezeichneten Makros auch jederzeit schnellen Zugriff haben.

Zum Verknüpfen eines Makros mit einem neuen Symbol aktivieren Sie die Funktion *Anpassen* aus dem Untermenü *Ansicht* |

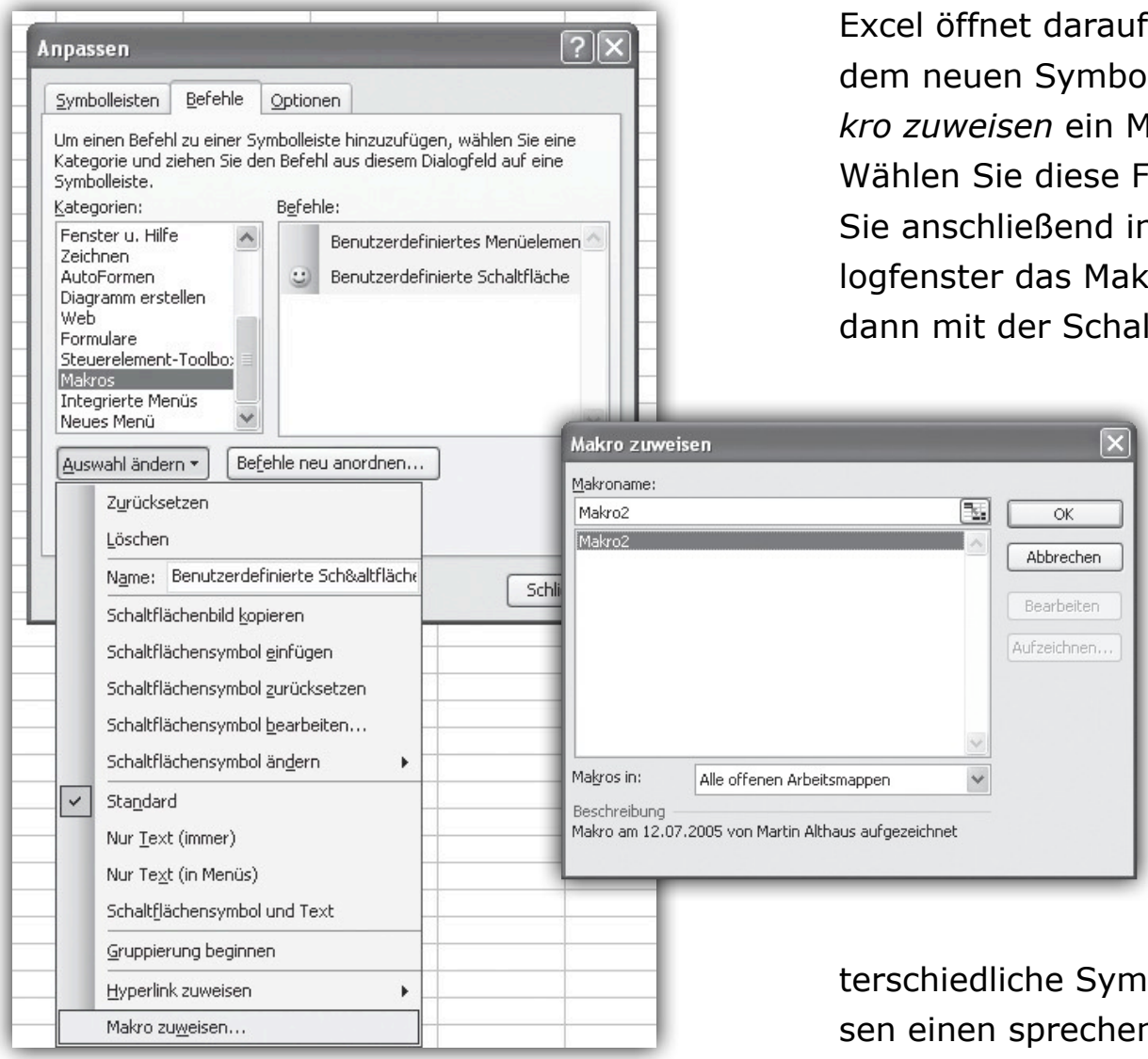
*Symbolleisten* oder dem Kontextmenü einer eingeblendeten Symbolleiste. In dem erscheinenden Dialogfenster *Anpassen* wechseln Sie in das Register *Befehle*. Aus der Auswahlliste *Kategorie* wählen Sie den Eintrag *Makros*. Rechts im Fenster erscheint daraufhin ein „Smiley" (Sonne mit Gesicht).


Diesen Smiley ziehen Sie nun bei gedrückter linker Maustaste an eine beliebige Position einer eingeblendeten Symbolleiste.




Nachdem Sie so das neue Symbol an der gewünschten Position eingebunden haben, müssen Sie es noch mit dem Makro verknüpfen. Dazu betätigen Sie im Fenster *Anpassen* die Schaltfläche *Auswahl ändern*.

Achten Sie aber darauf, dass das neue Symbol vor dem Betätigen dieser Schaltfläche immer noch durch einen schwarzen Kasten markiert ist. Sonst kann es vorkommen, dass Sie entweder ein anderes Symbol mit dem Makro verknüpfen oder die Schaltfläche *Auswahl ändern* nicht betätigen können.

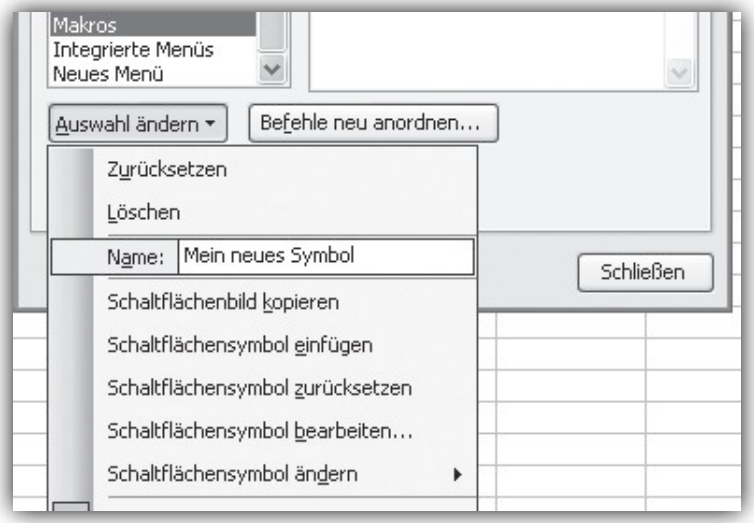



Excel öffnet daraufhin ein Menü, in dem Sie dem neuen Symbol über die Funktion *Makro zuweisen* ein Makro zuweisen können. Wählen Sie diese Funktion aus, aktivieren Sie anschließend in dem erscheinenden Dialogfenster das Makro und bestätigen Sie das dann mit der Schaltfläche *OK*.

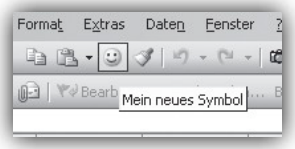
Falls Sie mehrere aufgezeichnete Makros mit Symbolen verknüpfen möchten, können Sie natürlich beliebig viele neue Symbole in Ihre Symbolleisten einbinden. Damit Sie diese Symbole auch voneinander unterscheiden können, sollten Sie unterschiedliche Symbole verwenden und diesen einen sprechenden Namen zuweisen.

## Namen eines Symbols ändern

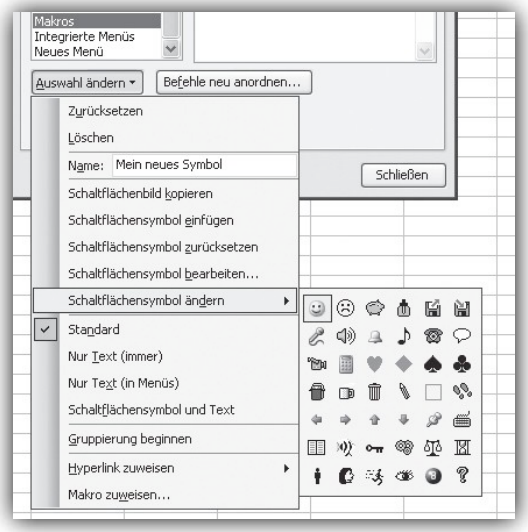
Um den Namen eines Symbols zu ändern, finden Sie im Menü *Auswahl ändern* das Eingabefeld *Name*. Geben Sie dort einen Namen für das Symbol ein.




Dieser Name erscheint dann in der Quickinfo des neuen Symbols. Die Quickinfo blendet Excel ein, wenn Sie den Mauszeiger über ein Symbol bewegen.




Neben dem Namen ist auch ein unterschiedliches Aussehen der Symbole zur Unterscheidung wichtig. Um das Aussehen eines neuen Symbols zu ändern, finden Sie im Menü *Auswahl ändern* den Menüpunkt *Anderes Symbol*.




Nach der Auswahl dieses Punktes klappt Excel ein Register mit 42 Symbolen auf. Klicken Sie dort ein beliebiges Symbol an, um das Aussehen des aktiven neuen Symbols zu ändern.

## Markierte Zellen einzeln verarbeiten

*VBA-Anwendung: Zellbereiche durchlaufen.*

VBA-Programme benötigen Hilfsmittel und Methoden, um auf die einzelnen Elemente einer Matrix zuzugreifen. Die flexibelste Form des Zugriffs erreichen Sie über das Kommando *Cells*. Sie können den Cells-Befehl, durch einen Punkt getrennt, einfach an Ihre Matrixvariable anhängen und dann über die Parameter von *Cells* festlegen, auf welches Element der Matrix Sie zugreifen möchten.

Mit diesen Parametern übergeben Sie dem Cells-Kommando die Informationen, welche Zeile und welche Spalte der Matrix verarbeitet werden sollen. Als ersten Wert übergeben Sie eine Zeilennummer, mit dem zweiten Parameter legen Sie eine Spaltennummer fest.

Dieser Bezug ist immer relativ. Der Befehl *Cells(1,1)* steuert in Kombination mit Ihrer Matrixvariablen immer die linke obere Ecke der Matrix an. Das folgende Makro zeigt, wie Sie eine Matrix im Bereich B1:D5 der aktiven Mappe festlegen und dann die einzelnen Elemente der Matrix der Reihe nach verarbeiten. Im Programmbeispiel wird der Wert 100 in die Zellen der Matrix eingetragen:

```
Sub MatrixMit100Fuellen()
Dim MeineMatrix As Range
Dim Zeile, Spalte As Integer
Set MeineMatrix = Workbooks
("m130.xls").Sheets("Tabelle1").
Range("b1:d5")
For Zeile = 1 To 5
For Spalte = 1 To 3
MeineMatrix.Cells(Zeile,Spalte).
Formula = 100
Next Spalte, Zeile
End Sub
```

Innerhalb der beiden Schleifen im Programmbeispiel findet die Verarbeitung der einzelnen Elemente der Matrix statt. Die eine Schleife (*For Zeile*) durchläuft die Matrix zeilenweise. Innerhalb dieser Schleife ist eine zweite Schleife (*For Spalte*) eingebunden, mit der die Matrix spaltenweise durchlaufen wird.

Auf diese Weise werden Zeile für Zeile alle Elemente der Matrix der Reihe nach verarbeitet. Das Eintragen des Werts 100 nimmt dann die Formula-Anweisung für jede Zelle der Matrix vor.

### Dimensionen einer Matrix bestimmen

Im Programmbeispiel *Matrix Mit100Fuellen* ist die Dimension der Matrix bekannt – weil sie schließlich zuvor über den Set-Befehl direkt angelegt wurde. Daher ist es einfach, für die beiden Schleifen die Endwerte festzustellen. In vielen Fällen ist das anders. Das gilt vor allem dann, wenn Anwender Bereiche in Tabellen mit der Maus markieren und sie diese Matrix in ihren VBA-Programmen weiterverarbeiten möchten.

Um die Anzahl der Zeilen einer Matrix zu ermitteln, setzen Sie die Kombination der Eigenschaften *Rows* und *Count* ein. Den markierten Bereich stellen Sie über die Eigenschaft *Selection* fest. Das folgende Kommando ermittelt die Anzahl der Zeilen einer Matrix, die Sie mit der Matrixvariablen *MeineMatrix* definiert haben:

```
Zeilenanzahl = MeineMatrix.Rows.
Count
```

Für die Berechnung der Spaltenanzahl einer Matrix verwenden Sie die Ausdrücke *Columns* und *Count*. Kombinieren Sie beide Eigenschaften zu einer Abfrage. Der folgende Befehl zeigt, wie es in der Praxis aussieht, wenn

Sie die Anzahl der Spalten einer Matrix bestimmen möchten:

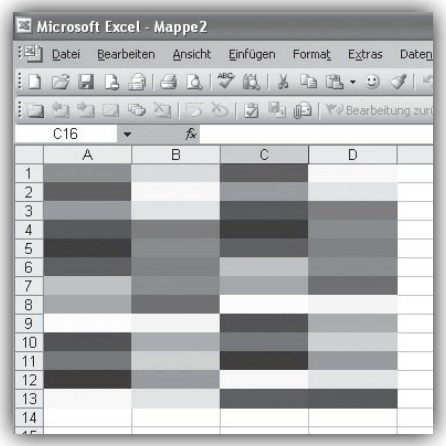
```
Spzahl = MeineMatrix.Columns.Count
```

Das folgende kleine Programm setzt den aktuell markierten Bereich in der Tabelle als Matrix ein und legt neue Farben für die Zellen fest:

```
Sub FarbenAendern()
Set Matrix = Selection
For Z = 1 To Matrix.Rows.Count
For S = 1 To Matrix.Columns.Count
Matrix.Cells(Z, S).Interior.
ColorIndex = Z * 2 + S
Next S, Z
End Sub
```

## Verschachtelte Schleifen

Das Programm setzt die Technik der zwei verschachtelten Schleifen ein, um der Reihe nach die Zellen zu verarbeiten. Für jede Zelle wird dann ein neuer Farbcode berechnet, den das Makro über die Eigenschaft *Interior.Colorindex* als Farbwert für den Hintergrund der Zelle definiert. Die folgende Abbildung zeigt, wie das Ergebnis aussehen kann.




Durch die Technik der verschachtelten Formeln ist es auch problemlos möglich, die Zeilen und Spalten einer Markierung unterschiedlich zu verarbeiten. Wenn jede Zeile der Matrix eine andere Farbe erhalten soll, reicht es aus, den ColorIndex-Befehl im Makroprogramm *FarbenAendern* anzupassen:

```
Matrix.Cells(Z, S).Interior.Color-
Index = Z * 2
```

# Schritt für Schritt: Aufgezeichnete Makros bearbeiten

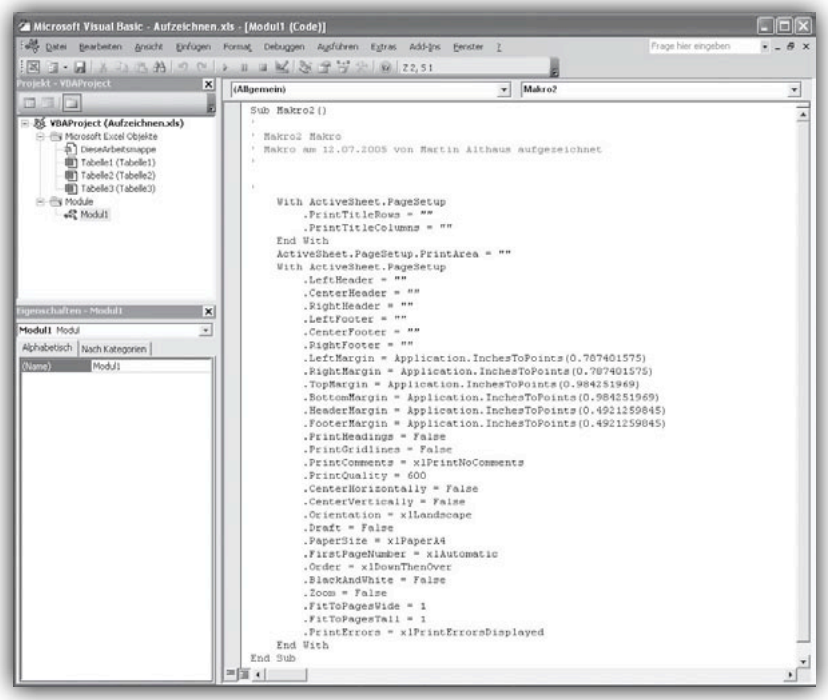
Das Problem beim Aufzeichnen von Makros ist, dass nicht immer nur diejenigen Arbeitsschritte aufgezeichnet werden, die Sie eigentlich in das Makro einbinden möchten. Jeder Klick auf eine Zelle während der Makroaufzeichnung wird auch bei der späteren Anwendung des Makros ausgeführt.

Unerwünschter Nebeneffekt: Das Makro nimmt Einstellungen vor, die Sie eigentlich nicht beabsichtigen. Außerdem wird das Makro dadurch länger und arbeitet langsamer. Damit Ihre aufgezeichneten Makros ohne Störungen funktionieren, können Sie den Makro-Code nachträglich noch überarbeiten und so an Ihre Bedürfnisse anpassen.

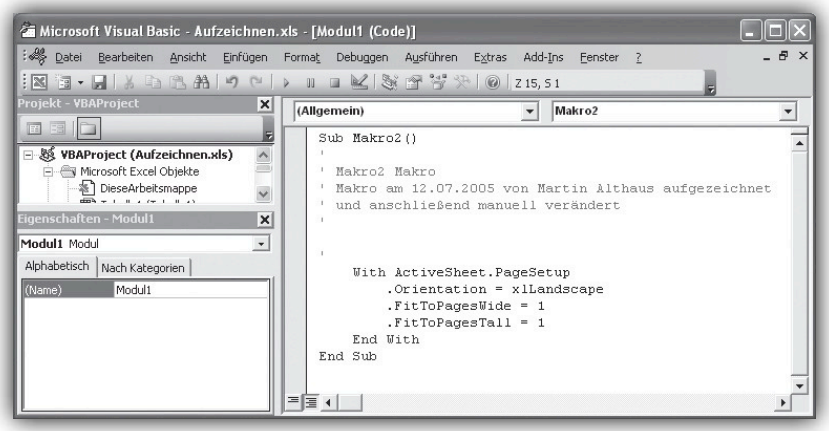
## VBA-Editor einsetzen

Um einen Makro-Code ab Excel 97 zu überarbeiten, aktivieren Sie die Funktion *Extras* | *Makro* | *Makros*, wählen das entsprechende Makro aus und betätigen die Schaltfläche *Bearbeiten*. Excel startet daraufhin automatisch den VBA-Editor in einem eigenen Fenster und blendet den Makro-Code des zuvor gewählten Makros ein.

In der folgenden Abbildung können Sie erkennen, dass das Makro aus vielen Programmbefehlen entsteht. Aufgezeichnet wurde aber nur eine relativ einfache Operation: Im Dialogfenster *Datei* | *Seite einrichten* sind die Druckeinstellungen auf *Querformat* und *Anpassen an eine Seite* festgelegt worden. In den aufgezeichneten Befehlen werden aber alle Einstellungen der Seiteneinrichtung aufgeführt, die zum Zeitpunkt der Makroaufzeichnung aktiv waren.




Diese überflüssigen Kommandos können Sie manuell löschen. Der Makro-Code sieht nach dem Löschen der überflüssigen Befehle so aus:




Das kann dazu führen, dass das Makro nicht so arbeitet, wie Sie sich das vorstellen: Das Makro soll nur die Einstellungen *Querformat* und *Anpassen an eine Seite* einschalten, aber alle anderen Einstellungen unverändert lassen.

Um überflüssige Befehle zu löschen, müssen Sie wissen, welche Kommandos für welche Aufgabe in dem aufgezeichneten Makro zuständig sind. Das ist nicht einfach, mit etwas Englischkenntnissen und Experimentierfreude ist es aber möglich. Beim Beispiel des Makros zum Einschalten der Optionen *Querformat*

und *Anpassen an eine Seite* sind nur drei Kommandos entscheidend, mit denen die Eigenschaften *Orientation*, *FitToPagesWide* und *FitToPagesTall* verändert werden.


Grundsätzlich gilt: Reduzieren Sie schrittweise den Makro-Code um einzelne Kommandos und prüfen Sie anschließend immer, ob das Makro noch so arbeitet, wie Sie sich das vorstellen.

### Fehler in der Schreibweise feststellen

Bevor Sie Ihre veränderten Makros starten, können Sie auch Excel selbst dazu veranlassen, nach Fehlern in der Schreibweise Ihrer Makros zu suchen. Rufen Sie dazu die Funktion *Kompilieren von VBAProject* aus dem Debuggen-Menü des VBA-Editors auf. Diese Funktion findet syntaktische Fehler und zeigt sie an.

## VBA-Tricks

### Kommando abschalten, ohne zu löschen

Wenn Sie mit VBA-Befehlen im VBA-Editor experimentieren, kann es sinnvoll sein, einzelne Programmzeilen abzuschalten, ohne gleich die entsprechende Programmzeile aus dem Quelltext zu entfernen. Dann können Sie die Programmzeile jederzeit wieder reaktivieren, wenn Sie mit der Änderung nicht das gewünschte Ergebnis erzielt haben. Um eine Programmzeile inaktiv zu schalten, können Sie diese Zeile in einen Kommentar umwandeln. Für VBA-Kommandos gilt: Texte hinter einem einfachen Anführungszeichen (das Zeichen auf der Taste [#]) werden als Kommentare gewertet und nicht ausgeführt.

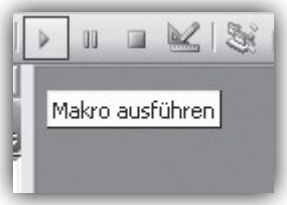
Das Umwandeln ist daher ganz einfach. Setzen Sie ein einfaches Anführungszeichen vor

die entsprechende Befehlszeile. Der VBA-Editor zeigt die Zeile nun in der Farbe Grün an. Das ist das Zeichen dafür, dass der Code als Kommentar interpretiert wird.

```
    With ActiveSheet.PageSetup
        .Orientation = xlLandscape
        .FitToPagesWide = 1
'        .FitToPagesTall = 1
    End With
End Sub
```

## Unterprogramme testen

Wenn Sie Unterprogramme testen möchten, können Sie sich den Umweg über die Funktion *Extras* | *Makro* | *Makros* von Excel sparen. Sie können Unterprogramme auch direkt im VBA-Editor testen. Das machen Sie folgendermaßen:




Bewegen Sie die aktive Eingabemarkierung auf einen Befehl in dem entsprechenden Unterprogramm (also wollten Sie in dem Programm den Quelltext verändern). Klicken Sie dann auf die Schaltfläche *Makro ausführen*. Sie sieht aus wie die Start-Taste bei einem Videorekorder. Die Abbildung zeigt Ihnen die Schaltfläche.

## Makros in allen Mappen verfügbar machen

Grundsätzlich legt Excel aufgezeichnete Makros immer in der Arbeitsmappe ab, in der Sie sie erstellt haben. Bei Bedarf können Sie Ihre Makros aber auch in allen Arbeitsmappen zur Verfügung stellen. Dazu gehen Sie folgendermaßen vor:

Rufen Sie über das Menü *Extras* den Befehl *Makro | Aufzeichnen* auf. In dieser Dialogbox öffnen Sie mit einem Mausklick das Listfeld *Makro speichern in*. Aus der Liste der möglichen Speicherorte wählen Sie die Option *Persönliche Makroarbeitsmappe*. Das bestätigen Sie mit der Schaltfläche *OK*. Dann können Sie mit der Aufzeichnung Ihres Makros beginnen. Wenn diese abgeschlossen ist, können Sie das Makro in allen Arbeitsmappen verwenden.

Info

Beachten Sie, dass Sie den Speicherort eines aufgezeichneten Makros nur vor der Aufzeichnung bestimmen können. Nachträgliche Änderungen am Speicherort erlaubt Excel nicht. Sie können das Makro aber über den VBA-Editor in die persönliche Mappe kopieren.

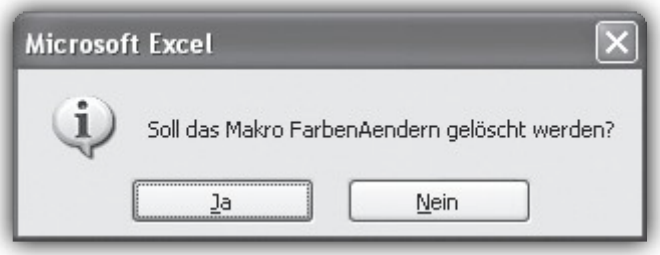
**So löschen Sie ein Makro, das Sie nicht mehr benötigen**

Makros, die Sie in der Vergangenheit aufgezeichnet haben und jetzt nicht mehr benötigen, sollten Sie aus Gründen der Übersichtlichkeit entfernen. Dazu gehen Sie folgendermaßen vor:

Öffnen Sie die Arbeitsmappe, die das zu löschende Makro enthält. Dann rufen Sie im Menü *Extras* den Befehl *Makro* auf. Im verzweigenden Menü aktivieren Sie den Befehl *Makros*.

In dieser Dialogbox wird Ihnen eine Liste der verfügbaren Makros angezeigt. Damit nur noch die Makros angezeigt werden, die in der aktiven Arbeitsmappe verfügbar sind, aktivieren Sie im Listfeld *Makros in* die Option *Diese Arbeitsmappe*. In der Liste der Makros

markieren Sie mit einem Mausklick das zu entfernende Makro. Anschließend klicken Sie auf die Schaltfläche *Löschen*. Die in der folgenden Abbildung gezeigte Sicherheitsabfrage bestätigen Sie mit der Schaltfläche *Ja*, damit ist das Makro entfernt.




## Makros in Excel 95

In Excel 95 gibt es den VBA-Editor noch nicht. Dort bearbeiten Sie Makros direkt in Excel auf einem eigenen VBA-Blatt. Aktivieren Sie zuerst die Funktion *Extras* | *Makro* und wählen Sie dann im erscheinenden Dialogfenster das gewünschte Makro.

Betätigen Sie anschließend die Schaltfläche *Bearbeiten*. Excel wechselt daraufhin zum Makroblatt dieses Makros. Hier können Sie das Makro so bearbeiten, wie Sie es von Ihrem Textverarbeitungsprogramm gewohnt sind.

Allerdings wird die Makrosprache von Excel von Version zu Version ergänzt: Nicht alle aktuellen Makrobefehle stehen Ihnen in Excel 95 zur Verfügung.

### Ältere Makros werden automatisch übersetzt

Wenn Sie Arbeitsmappen, in denen ältere Excel-95-Makros enthalten sind, in einer neueren Excel-Version öffnen, übersetzt Excel die bestehenden Makros automatisch in den neuen Makro-Standard. Zur Veränderung dieser Makros setzen Sie dann den VBA-Editor ein.

## Wechseln zwischen Mappen und Tabellen

Häufig ist es wünschenswert, in einem Makro genau festzulegen, welche Tabelle verarbeitet werden soll. Das können Sie erledigen, indem Sie vor den Programmbefehlen, die einen Zellbereich verarbeiten, ein neues Kommando einfügen, mit dem Sie gezielt eine bestimmte Arbeitsmappe oder Tabelle auswählen.

Dazu fügen Sie einfach vor dem entsprechenden Befehl (zum Beispiel einer Zeile, die mit dem Ausdruck „.Select" aufhört) eine Leerzeile ein. In diese Leerzeile tragen Sie nun das Kommando ein, mit dem Sie eine Arbeitsmappe und Tabelle Ihrer Wahl ansteuern.

### Gewünschte Mappe auswählen

Wenn Sie anstelle der Arbeitsmappe, die das aufgezeichnete Makro bisher verarbeitet (meist ist es die aktuelle Arbeitsmappe), eine andere Mappe auswählen möchten, setzen Sie folgenden Befehl ein:

```
Workbooks("Meine.xls").
Activate
```

Über den Activate-Befehl teilen Sie Excel mit, welche Arbeitsmappe als aktuelle Arbeitsmappe ausgewählt werden soll. Fügen Sie einfach den Namen der gewünschten Arbeitsmappe zwischen den Klammern in Anführungszeichen ein. Achten Sie darauf, dass Sie den kompletten Namen der Mappe inklusive des nachgestellten „.xls" verwenden müssen, damit das Kommando richtig arbeitet.

### Tabelle zur Verarbeitung festlegen

Falls Sie nicht nur die Arbeitsmappe, sondern auch die zu verarbeitende Tabelle spezifizieren möchten, fügen Sie nach der Programmzeile mit dem Workbooks-Befehl eine neue,

leere Programmzeile ein. Hier können Sie dann mit einem anderen Activate-Kommando festlegen, welche Tabelle innerhalb der zuvor gewählten Arbeitsmappe als aktive Tabelle eingestellt werden soll.

### Zurück zur alten Tabelle springen

Wenn Ihr Makro dann in der gewählten Mappe und Tabelle die passenden Arbeitsgänge ausgeführt hat, können Sie anschließend wieder in die Tabelle zurückspringen, die beim Aufruf des Makros aktiv war. Besonders pfiffig: Registrieren Sie diese Tabelle zu Beginn in Ihrem Makro (also gleich nach der Zeile, die mit *Sub* beginnt), indem Sie hier die beiden folgenden Zeilen einfügen:

```
Set AMap = ActiveWorkbook
Set ATab = ActiveSheet
```

Jetzt können Sie am Ende des Makros wieder zurück zur ursprünglichen Tabelle springen, indem Sie vor dem Kommando *End Sub* die folgende Zeile einfügen:

```
AMap.Activate ATab.Activate
```

### Tabelle nach Wahl anzeigen

Falls Sie anstelle der aktiven Tabelle eine andere Tabelle der Arbeitsmappe als aktuelle Tabelle anzeigen möchten, setzen Sie anstelle des Ausdrucks *ATab.Activate* eine Programmzeile ein, mit der Sie die gewünschte Tabelle auswählen. Das folgende Listing zeigt, wie die Befehle aussehen müssen:

```
AMap.
Worksheets("Tabelle1").
Activate
```

Tragen Sie dann den Namen der Tabelle zwischen den Klammern in Anführungszeichen ein. Sie können stattdessen auch eine Zahl verwenden. Excel zählt die Tabellenblätter, mit 1 beginnend, von links nach rechts in

der Registerleiste durch. Beim Einsatz einer Tabellenblattnummer dürfen Sie aber keine Anführungszeichen verwenden!

## Zellbereiche und Markierungen verarbeiten

Besonders praktisch ist der Einsatz von VBA, weil Sie mit einem Programm einen ganzen Bereich von Zellen Schritt für Schritt verarbeiten können. Solche Zellbereiche nennt man „Matrix". Wenn Sie in einer Tabelle einen Bereich mit der Maus markieren, können Sie diese Matrix anschließend in einem VBA-Programm verarbeiten.

Damit Sie in Ihren VBA-Programmen problemlos Matrizen erzeugen und auf sie zugreifen können, müssen Sie passende Variablen im Programm-Code definieren.

Legen Sie dazu über den Dim-Befehl Variablen an, denen Sie den Datentyp *Range* zuordnen. Dann können Sie über einen Set-Befehl den Variablen sofort den entsprechenden Zellbereich Ihrer Matrix zuweisen. Das folgende Listing zeigt ein Beispiel:

```
Sub MatrixVerarbeitung()
Dim MeineMatrix As Range
Set MeineMatrix =
Range("a1:b4")
End Sub
```

Das Programmbeispiel richtet zuerst die Variable *MeineMatrix* ein. Anschließend weist der Set-Befehl dieser Variablen den Zellbereich A1:B4 zu.

Nach diesem Vorgang können Sie über *MeineMatrix* auf diesen Zellbereich zugreifen.

### Matrix über Text festlegen

Sie können an dem Beispiel erkennen, wie einfach das Einrichten einer Matrix in Ihren VBA-Programmen ist. Sie definieren in der Range-Festlegung einfach den gesamten Bereich der Matrix über einen Text. Der Text legt die linke obere und die rechte untere Ecke des Zellbereichs fest und trennt beide Angaben durch einen Doppelpunkt.

## Eine neue, leere Arbeitsmappe anlegen

Als Ergebnis vieler VBA-Programme entsteht eine fertige Tabelle, in der Berechnungen, Formatierungen usw. vordefiniert sind. Wenn Sie eine solche Tabelle nicht auf der Basis einer Mustervorlage erstellen, sondern in VBA benutzergesteuert erzeugen wollen, ist der erste Schritt das Anlegen einer neuen, leeren Arbeitsmappe.

Wichtig dabei: Die Referenz auf diese Arbeitsmappe müssen Sie ebenfalls speichern, damit Sie die Mappe in Ihren VBA-Programmen auch ansprechen können. Hier ist die passende VBA-Funktion, die Ihnen bei dieser Aufgabe hilft:

```
Private Function
LeereMappeAnlegen()
As String
Dim wb As Workbook
Set wb = Workbooks.Add
LeereMappeAnlegen = wb.Name
End Function
```

Die Funktion öffnet eine neue, leere Arbeitsmappe und liefert als Funktionsergebnis den Namen der Arbeitsmappe zurück. Über diesen Namen können Sie dann auf die Arbeitsmappe zugreifen. Damit Sie in Ihren Makros die Funktion problemlos nutzen können, sollten Sie das Ergebnis der Funktion

*LeereMappeAnlegen* in einer Variablen des Typs *String* speichern. So können Sie in Ihren VBA-Routinen sofort auf die neue, leere Arbeitsmappe zugreifen.

### So legen Sie eine leere Mappe an

Ein Beispiel für den Aufruf der Funktion finden Sie im folgenden kleinen Testprogramm. Es ruft die Funktion *LeereMappeAnlegen* auf und gibt den Namen der Mappe in einem Mitteilungsfenster aus:

```
Sub TesteLeereMappe()
Dim MeinWorkbook As String
MeinWorkbook = LeereMappeAnlegen
MsgBox (MeinWorkbook)
End Sub
```

# Sie mögen Excel?
# Dann machen Sie doch mehr daraus.

Mit Excel effizient zu arbeiten, ist oft einfacher gesagt als getan. Viel zu häufig erschwert die kaum noch überschaubare Fülle an Formeln, Funktionen, Makros und Add-Ins die tägliche Arbeit. Genau hier setzt der neue Informationsdienst **Excel aktuell** an.

Excel-Profi Martin Althaus und sein Team stellen jeden Monat die besten Tools für Listen, Formeln, Diagramme sowie exklusive Erweiterungen für Ihr Excel zusammen. Alle Lösungen stehen Ihnen zum Soforteinsatz direkt zur Verfügung. So können Sie Ihre Arbeit effektiv gestalten und sparen garantiert wertvolle Arbeitszeit.

Auch der Inhalt dieses eBooks ist Teil von **Excel aktuell**. Sie sind auf den Geschmack gekommen? Dann fordern Sie jetzt Ihren 30-Tage-Gratis-Test an! Als Dankeschön erhalten Sie die Excel-Flatrate-CD mit Fertiglösungen und Power-Funktionen gratis.

**Hier geht's zum 30-Tage-Gratis-Test.**

EXCEL_aktuell

Die besten Tipps, Tricks und Tools für Ihre erfolgreiche Tabellenkalkulation

# ratschlag24.com

Das neue Ratgeber-Portal ratschlag24.com liefert Ihnen täglich die besten Ratschläge direkt auf Ihren PC.

Viele bekannte Autoren, Fachredakteure und Experten schreiben täglich zu Themen, die Sie wirklich interessieren und für Sie einen echten Nutzen bieten. Zu den Themen zählen Computer, Software, Internet, Gesundheit und Medizin, Finanzen, Ernährung, Lebenshilfe, Lernen und Weiterbildung, Reisen, Verbrauchertipps und viele mehr. Alle diese Ratschläge sind für Sie garantiert kostenlos. Testen Sie jetzt ratschlag24.com – Auf diese Ratschläge möchten Sie nie wieder verzichten.

ratschlag24.com ist ein kostenloser Ratgeber-Dienst der eload24 AG
www.eload24.com

# Viel guter Rat ab 3 Euro monatlich
# Die neuen Flatrate Modelle von eload24

Das ist ein Wort: Sie bekommen **freien Zugang zu allen eBooklets und eBooks** bei eload24. Sie können alles laden, lesen, ausdrucken, ganz wie es Ihnen beliebt. Eine echte Flatrate eben, ohne Wenn und Aber. Sie werden staunen: Unser Programm mit nützlichen eBooklet-Ratgebern ist groß und wird laufend erweitert.

**Der Preisvorteil ist enorm:**

24 Monate Flatrate für nur 72,– € (3,– € monatlich)

12 Monate Flatrate für nur 48,– € (4,– € monatlich)

6 Monate Flatrate für nur 36,– € (6,– € monatlich)

Selbst wenn Sie nur zwei eBooklets der preiswertesten Kategorie im Monat laden, sparen Sie im Vergleich zum Einzelkauf.

Tausende Kunden haben dieses Angebot schon wahrgenommen, profitieren auch Sie dauerhaft. Wenn Sie nach Ablauf der Flatrate weitermachen wollen, dann brauchen Sie nichts zu tun: das Flatrate-Abonnement verlängert sich automatisch. Bis Sie es beenden.

Kaufen Sie jetzt die Flatrate Ihrer Wahl. Und schon einige Augenblicke später stehen Ihnen hunderte toller Ratgeber uneingeschränkt zur Verfügung: Packen Sie mal richtig zu!


**www.eload24.com/flatrate**