

Theorie Excel - VBA (Visual Basic Application)

Inhaltsverzeichnis

1	Einstieg.....	2
1.1	Symbolleiste Visual Basic	2
1.2	Editor	3
1.3	Befehle	4
2	Vorgehensweise beim Programmieren	4
2.1	Betroffene Dateien festhalten	4
2.2	Ablauf festhalten	4
2.3	Makro aufzeichnen.....	4
2.4	Makro dokumentieren	4
2.5	Makro anpassen.....	4
2.6	Makro dokumentieren	5
2.7	Code testen.....	5
3	Code.....	5
3.1	Codeaufbau.....	5
3.2	Code anpassen	5
3.2.1	Datei öffnen -> statisch.....	5
3.2.2	Datei öffnen -> dynamisch	5
3.2.3	Datei speichern -> statisch	6
3.2.4	Datei speichern -> dynamisch	6
3.2.5	Feld markieren -> statisch	6
3.2.6	Feld markieren -> dynamisch	6
3.2.7	Markierungen verschieben	7
3.2.8	Rechnen und Resultat in Messagebox anzeigen	7
3.2.9	Rechnen und Resultat in Arbeitsmappe schreiben	7
3.2.10	Do Until - Schleifen erstellen	8
3.3	Arbeiten mit Formularen.....	9
4	Begriff Konvention	10
5	Tipps.....	10
5.1.1	Formular automatisch öffnen	10
5.1.2	Fehlerfrei programmieren	10
5.1.3	Format von Befehlen suchen.....	10
6	Mustercode.....	11
6.1	Beispiel.....	11
6.2	Flussdiagramm für Mustercode	13
7	Literaturverweis	14

Theorie Excel - VBA (Visual Basic Application)

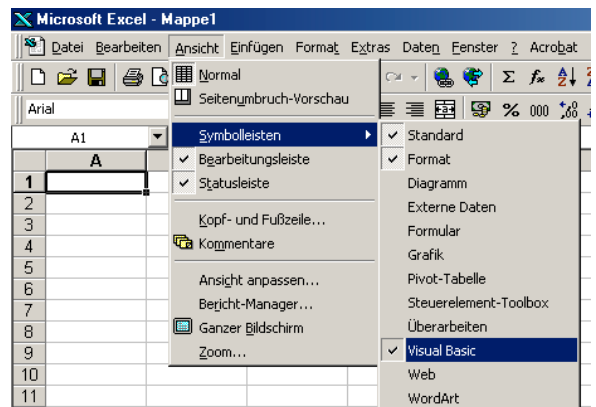
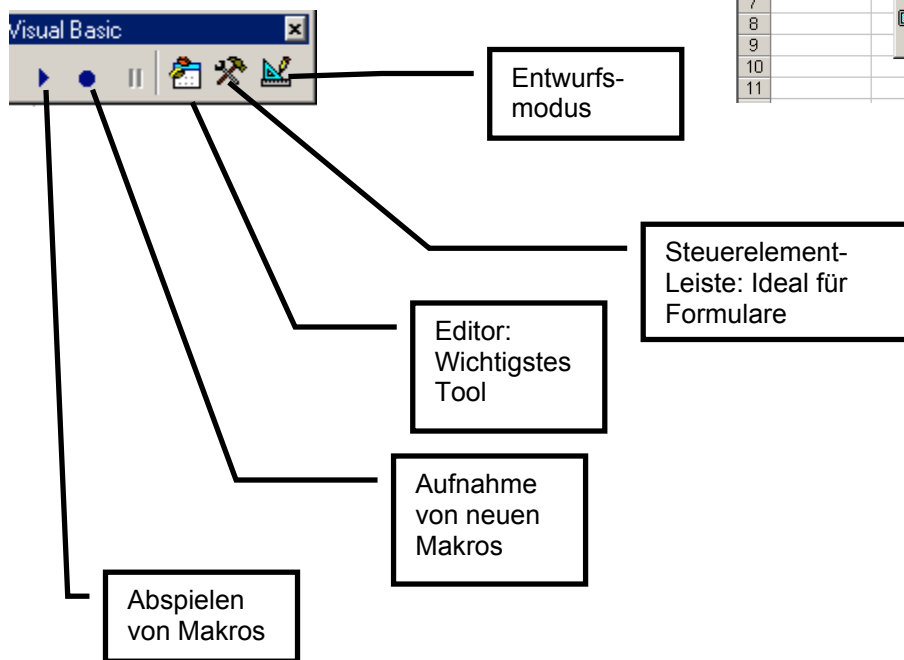
1 Einstieg

1.1 Symbolleiste Visual Basic

Damit mit Visual Basic (VBA) gearbeitet werden kann, schaltet man am Besten die Symbolleiste Visual Basic ein.

Ansicht -> Symbolleiste -> Visual Basic

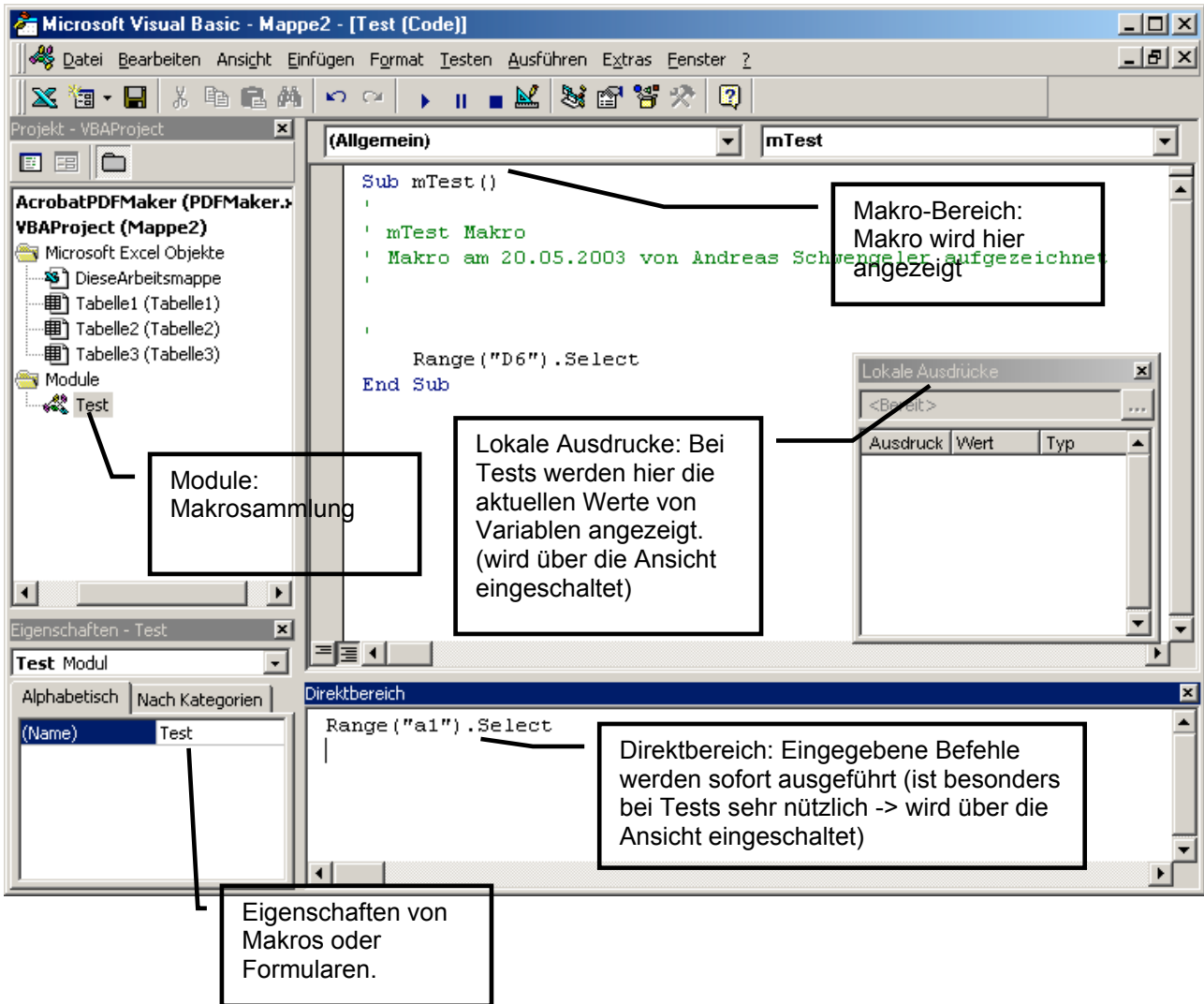
Die folgende Symbolleiste wird dann angezeigt:



Theorie Excel - VBA (Visual Basic Application)

1.2 Editor

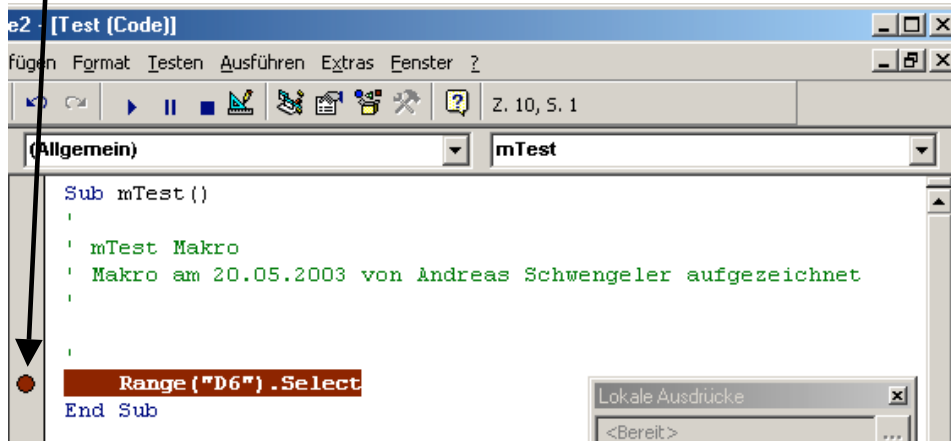
Für Test empfiehlt es sich den Editor verkleinert geöffnet zu halten, so kann man im Hintergrund beobachten, was gerade passiert. Gleichzeitig werden im Vordergrund die Programmzeilen angezeigt.



Theorie Excel - VBA (Visual Basic Application)

1.3 Befehle

- F5 führt Makro sofort aus (bzw. bis zum nächsten Haltepunkt)
- F8 führt einen Schritt im Makro aus (-> für Tests geeignet)
- Haltepunkt klicken Sie im linken grauen Rand auf die gewünschte Zeile und schon wird ein manueller Haltepunkt eingefügt. D.h. das Makro wird beim Ausführen an dieser Stelle gestoppt.



2 Vorgehensweise beim Programmieren

2.1 Betroffene Dateien festhalten

Halten Sie alle betroffenen Dateien mit dem ganzen Pfad fest. Es erleichtert Ihnen anschliessend das Programmieren. Zusätzlich muss man sich so bereits zu Beginn Gedanken über das Ausmass des Projektes machen.

2.2 Ablauf festhalten

Schreiben Sie einen Ablauf (z.B. in Form eines Flussdiagrammes) der Aktionen, welche Sie geplant haben.

2.3 Makro aufzeichnen

Der Aufzeichnungsmodus ist sehr geeignet, um ein "Rohgerüst" zu erhalten. Man zeichnet z.B. einen Kopiervorgang auf (auch wenn man später 10 verschiedenen ausführen will) und übernimmt die vom System übertragenen Angaben. So kann man bereits einige Flüchtigkeitsfehler verhindern.



2.4 Makro dokumentieren

Beim Aufzeichnungsmodus ist das Problem, dass die einzelnen Schritte nicht dokumentiert sind. Jeder selber erstellte Code erscheint Ihnen in den ersten zwei Wochen, jedoch ist er für Sie auch in einem Jahr noch logisch? - Darum empfehle ich Ihnen sich die Zeit für eine kurze Dokumentation zu nehmen. Idealerweise fügen Sie in den entsprechenden Linie die passenden Kommentare ein. Diese beginnen mit einem ' Hochkomma, im Editor werden die Kommentare zusätzlich noch grün angezeigt.

2.5 Makro anpassen

Passen Sie jetzt Ihr aufgezeichnetes Makro entsprechend an. D.h. machen Sie aus einem statischen Dateiname ein dynamischer, indem Sie eine Variable mit dem Dateiname versehen und diese zu Beginn des Makros definieren.

Oder wiederholen Sie einige Schritte mit Hilfe von Schleifen.

Theorie Excel - VBA (Visual Basic Application)

2.6 Makro dokumentieren

Auch hier gilt: Ihre Schleifen und Variablen sind im Moment logisch, jedoch wie sieht das in einem Jahr aus. Darum gilt: Je besser Sie dokumentiert sind, je einfacher sind Ihre späteren Anpassungen.

2.7 Code testen

Benutzen Sie dafür die Funktion mit F8. (Einzelschritte)

3 Code

3.1 Codeaufbau

Jeder Makrocode beginnt mit SUB MAKRONAME() und endet mit END SUB.

```
Sub Makroname()  
  
'Code in VBA  
  
End Sub
```

3.2 Code anpassen

3.2.1 Datei öffnen -> statisch

```
Sub mDatei_öffnen()  
  
' öffnet Datei  
Workbooks.Open FileName:= D:\Daten\Uebungen.xls"  
End Sub
```

3.2.2 Datei öffnen -> dynamisch

```
Sub mDatei_öffnen()  
  
' Aktiviert das Sheet (Blatt) "Daten"  
Sheets("Daten").activate  
  
' Sucht Dateiname in der Zelle A5 und schreibt diese in die Variable vDateiname  
vDateiname = Range("A5")  
  
' Sucht den Pfad in der Zelle A6 und schreibt diese in die Variable vPfad  
vPfad = Range("A6")  
  
' öffnet Datei  
Workbooks.Open FileName:= vPfad & vDateiname  
  
End Sub
```

Theorie Excel - VBA (Visual Basic Application)

3.2.3 Datei speichern -> statisch

```
Sub mSpeichern()  
  
' Speichert die aktive Datei  
ActiveWorkbook.SaveAs FileName:="D:\Daten\Uebungen1.xls", FileFormat _  
    :=xlNormal, Password:="", WriteResPassword:="", ReadOnlyRecommended:= _  
    False, CreateBackup:=False  
  
End Sub
```

3.2.4 Datei speichern -> dynamisch

```
Sub mSpeichern()  
  
' Aktiviert das Sheet (Blatt) "Daten"  
Sheets("Daten").activate  
  
' Sucht Dateiname in der Zelle A5 und schreibt diese in die Variable vDateiname  
vDateiname = Range("A5")  
  
' Sucht den Pfad in der Zelle A6 und schreibt diese in die Variable vPfad  
vPfad = Range("A6")  
  
' Speichert die aktive Datei  
ActiveWorkbook.SaveAs FileName:=vPfad & vDatei, FileFormat _  
    :=xlNormal, Password:="", WriteResPassword:="", ReadOnlyRecommended:= _  
    False, CreateBackup:=False  
  
End Sub
```

3.2.5 Feld markieren -> statisch

```
Sub mFeld_markieren_statisch()  
  
' Zellen B5 bis C5 werden markiert  
Range("B5:C5").Select  
  
End Sub
```

3.2.6 Feld markieren -> dynamisch

```
Sub mFeld_markieren_dynamisch()  
  
' Die Zelle auf Zeile 5 in der zweiten Spalte (B5) wird markiert  
Cells(5, 2).Select  
  
End Sub
```

Theorie Excel - VBA (Visual Basic Application)

3.2.7 Markierungen verschieben

```
Sub mFeld_markieren_offset()  
  
' Von der aktiven Zelle geht es 5 Zeilen hinunter und eine Spalte nach rechts.  
ActiveCell.Offset(5, 1).Select  
  
End Sub
```

3.2.8 Rechnen und Resultat in Messagebox anzeigen

```
Sub mRechnen_und_anzeigen()  
  
' legt Resultat in der Variable vResultat ab. (12)  
vResultat = 5 + 7  
  
' zeigt eine Infobox mit dem Resultat an.  
MsgBox (vResultat)  
  
End Sub
```

3.2.9 Rechnen und Resultat in Arbeitsmappe schreiben

```
Sub mRechnen_und_schreiben()  
  
' legt Variable vMultiplikator fest auf 3  
vMultiplikator = 3  
  
' legt Resultat in der Variable vResultat ab. (12)  
vResultat = 5 + 7  
  
' zählt zu den 12 vier dazu und legt es wieder in der Variable vResultat ab (16)  
vResultat = vResultat + 4  
  
' Multipliziert die Variable vResultat (16) mit der Variable vMultiplikator (3), das ergibt 48  
vResultat = vResultat * vMultiplikator  
  
' schreibt den Wert von vResultat (48) in die Zelle B1  
Range("B1") = vResultat  
  
End Sub
```

Theorie Excel - VBA (Visual Basic Application)

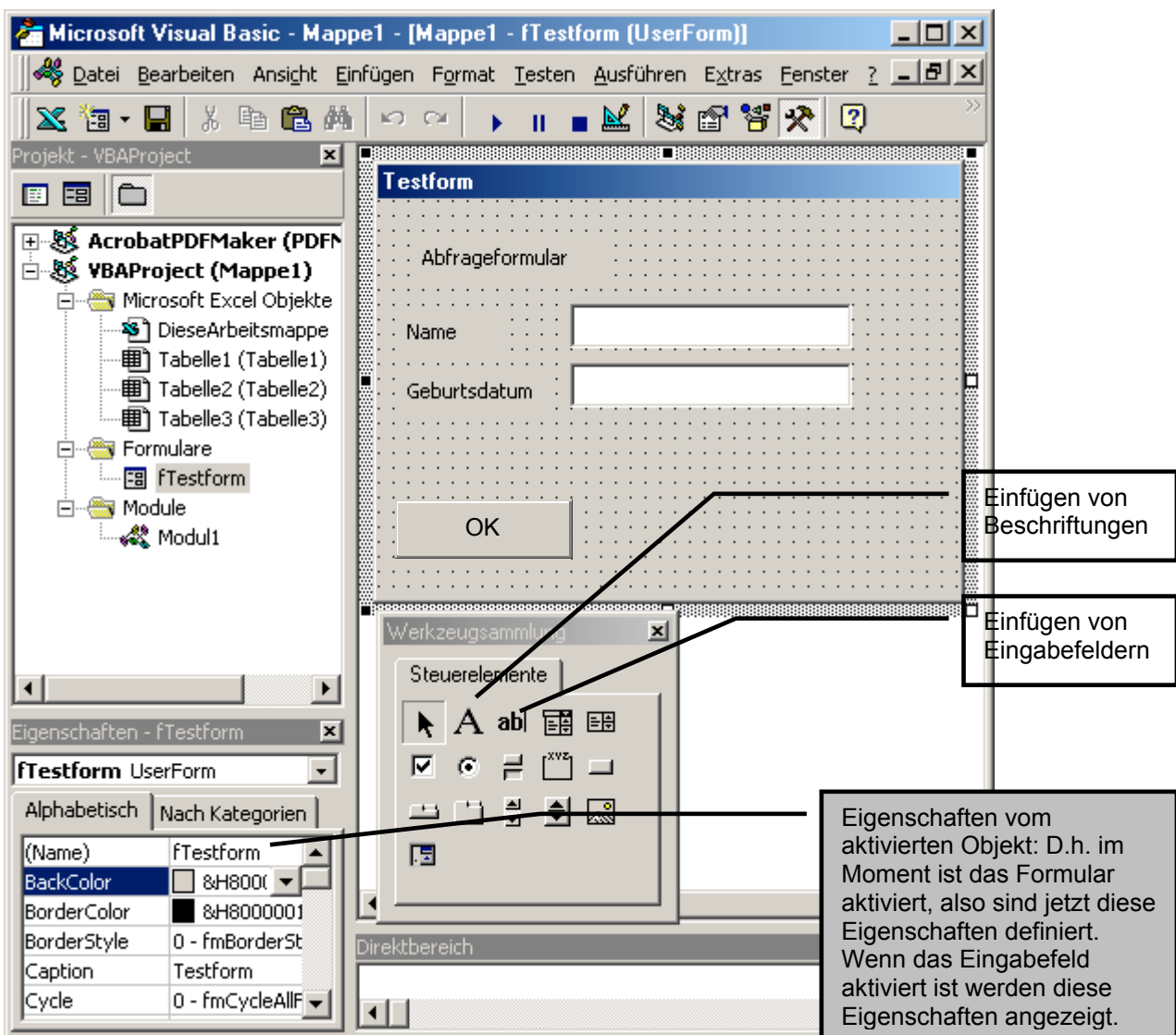
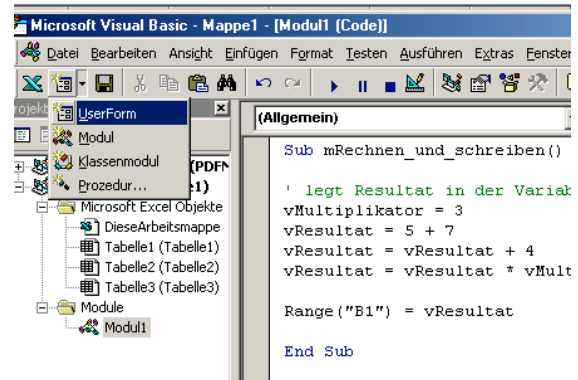
3.2.10 Do Until - Schleifen erstellen

```
Sub mDo_Until_Schlaufen()  
  
' Wert x wird auf eins gesetzt  
x = 0  
  
' Schleife wird ausgeführt bis der die Variable x 5 ist  
Do Until x = 5  
  
    ' Variable x wird um eines erhöht  
    x = x + 1  
    ' Variable x wird angezeigt in einem Infenster  
    MsgBox (x)  
  
' Die Schleife wird wiederholt  
Loop  
  
End Sub
```

Theorie Excel - VBA (Visual Basic Application)

3.3 Arbeiten mit Formularen

Sie können im Editor unter dem zweiten Symbol links ein sogenanntes UserForm einfügen.



Theorie Excel - VBA (Visual Basic Application)

Um dem Formular ein Makro zu hinterlegen können einzelne Felder oder Knöpfe angeklickt werden und anschliessend das gewünschte Makro hinterlegt werden. Das kann dann z.B. so aussehen.

```
Private Sub kOK_Click()  
  
' Wenn der Knopf OK (fOK) geklickt wird, schreibt Excel den Wert vom Feld tName ins Feld A1  
Range("A1") = tName  
  
' Wenn der Knopf OK (fOK) geklickt wird, schreibt Excel den Wert vom Feld tGeburtsdatum ins Feld B1  
Range("B1") = tGeburtsdatum  
  
End Sub
```

4 Begriff Konvention

Es sehr empfehlenswert immer die selbe Namensgebungen für gleiche Objekte zu verwenden. Darum empfehle ich folgende Konvention einzuhalten:

Vorzeichen	Objekt	Beispiel
f	Formular	fEingabemaske
k	Knopf	kAbbrechen
m	Makros	mDaten_kopieren()
t	Textfeld	tNachname
v	Variablen	vResultat

5 Tipps

5.1.1 Formular automatisch öffnen

Um ein Formular sofort beim öffnen einer Datei zu aktivieren kann folgendes Makro hinterlegt werden:

```
Private Sub Workbook_Open()  
  
fTestform.Show  
  
End Sub
```

5.1.2 Fehlerfrei programmieren

Schreiben Sie beim Programmieren alles klein. Denn wenn das System eine Variable oder ein Befehl kennt wird diese nach der Eingabetaste automatisch angepasst. Z.B:

```
fTestform.show -> wird automatisch umgewandelt in -> fTestform.Show  
oder  
vresultat = range("C1") -> wird automatisch umgewandelt in -> vResultat = Range("C1")
```

5.1.3 Format von Befehlen suchen

Theorie Excel - VBA (Visual Basic Application)

Schreiben Sie im Editor z.B. offset und markieren Sie das Wort. Anschliessend drücken Sie F1 und schon erhalten Sie die Beschreibung und Beispiele für diesen Befehl.

6 Mustercode

6.1 Beispiel

```
Sub mDatenKopieren()  
,  
' mDatenKopieren Makro  
' Makro am 11.04.2003 von AS aufgezeichnet  
,  
,  
  
    ' erstellt neuDatei -> Kommt von Einstiegseite  
    Stammdatei = Sheets("einstieg").Range("b17")  
    Pfad = Sheets("einstieg").Range("b21")  
  
    daten = Range("f1")  
  
    ' öffnet die Monatsdatei  
    Workbooks.Open FileName:= Pfad & daten, UpdateLinks:=3  
  
    Sheets("daten").Select  
  
    ' Fügt in Monatsdatei eine Zeile ein  
    Rows("4:4").Select  
    Selection.Insert Shift:=xlDown  
  
    ' Geht zur Stammdatei zurück und holte die Daten  
    Windows(Stammdatei).Activate  
  
    Zeilenzähler = 2  
  
    Do While Cells(Zeilenzähler, 1) <> ""  
        Zeilenzähler = Zeilenzähler + 1  
  
        If Cells(Zeilenzähler, 2) <> "" Then  
  
            Range(Cells(Zeilenzähler, 1), Cells(Zeilenzähler, 5)).Select  
            Selection.Copy  
  
            ' Geht zur Datendatei und fügt die Daten in Zeile 4 ein.  
            Windows(daten).Activate  
            Range("A4").Select  
            Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= False, Transpose:=False  
  
            ' fügt wieder eine leere Zeile ein  
            Rows("4:4").Select  
            Selection.Insert Shift:=xlDown  
  
            ' Geht zur Stammdatei zurück und holte die Daten  
            Windows(Stammdatei).Activate  
  
        End If  
    End Do  
End Sub
```

Theorie Excel - VBA (Visual Basic Application)

Loop

' geht zur Daten-Datei zurück und schliesst sie

Windows(daten).Activate

Rows("4:4").Select

Selection.Delete Shift:=xlUp

ActiveWorkbook.Save

ActiveWindow.Close

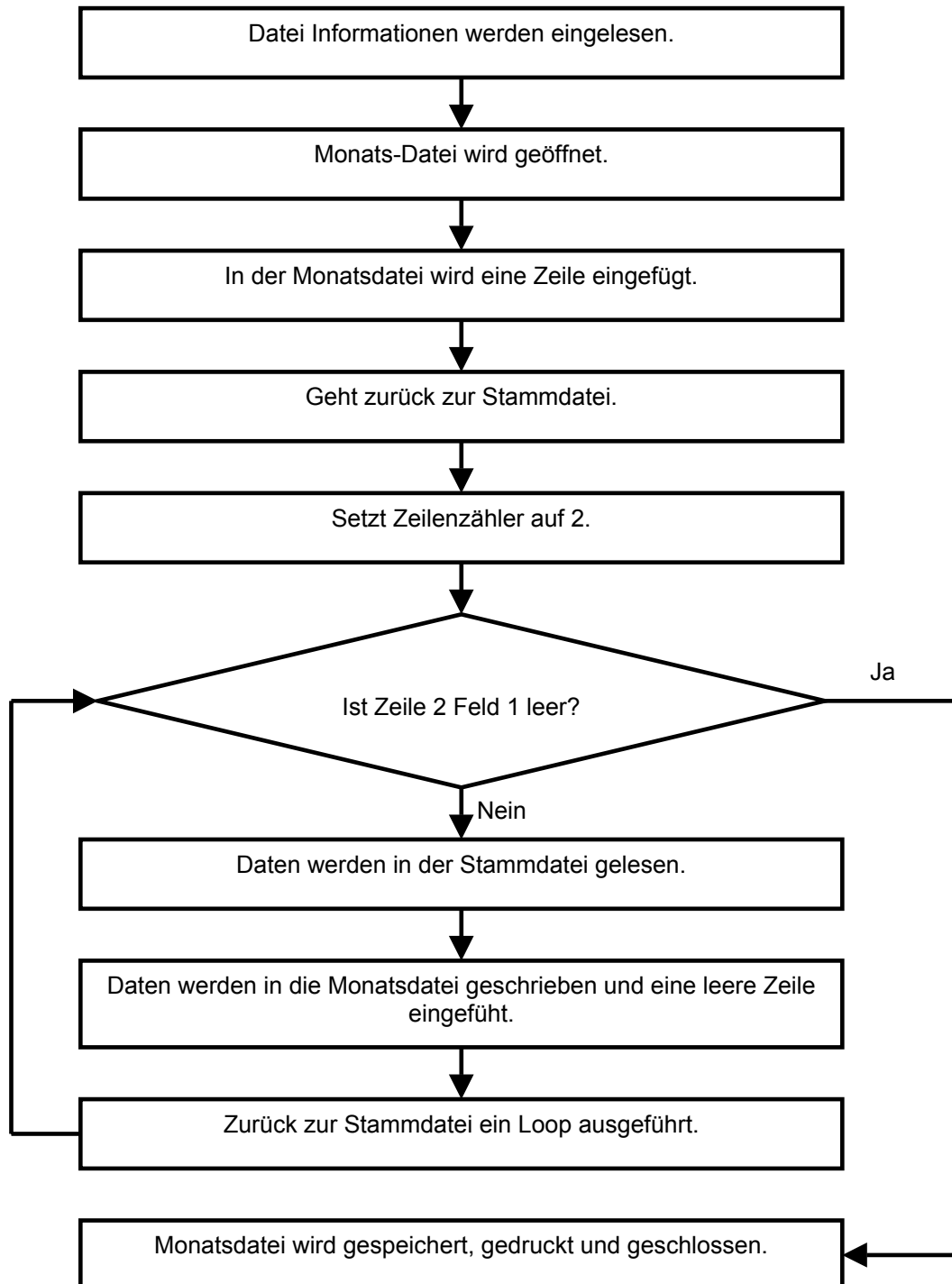
mDrucken

mSchliessenOhneSpeichern

End Sub

Theorie Excel - VBA (Visual Basic Application)

6.2 Flussdiagramm für Mustercode



Theorie Excel - VBA (Visual Basic Application)

7 Literaturverweis

Titel: VBA mit Excel 2000 lernen
Einstieg in die Welt der Makroprogrammierung. Mit CD-ROM
Prudenzi, Patrizia S.
CHF 39,50

deutsch
2002, 510 S., s/w. Abb., 4. A.
Kartoniert/Paperback
ISBN 3-8273-1572-7
[Addison Wesley \(dt. Programm\), München](#)



Für alle, die Excel programmieren möchten, auch wenn sie bisher keine Programmierkenntnisse haben, ist dieses Buch gedacht. Es vermittelt neben grundlegendem Wissen über die Programmiersprache VBA auch weitreichende Kenntnisse über die objektorientierte Programmierung und das Objektmodell von Excel. Alle Themen, die für die Excel-Programmierung relevant sind, werden ausführlich und mit zahlreichen Beispielen aus der Praxis behandelt. Schwerpunkte sind z.B.: die Entwicklungsumgebung, Elemente der Sprache VBA, Objekte, Ereignisse, Makroaufzeichnung und -anpassung, Erstellen von Dialogen, eigene Symbol- und Menüleisten, eigene Funktionen für die Tabellenblätter, Add-Ins sowie die neuen Möglichkeiten von Excel 2000 wie z.B. das Erstellen von Internet-Dokumenten mit VBA.