

3 Funktionen und Anweisungen

In VBA gibt es Hunderte von Funktionen. Die meisten davon können im kompletten Office-Paket eingesetzt werden, andere sind Excel-spezifisch. Funktionen unterteilt man in verschiedene Gruppen. So gibt es beispielsweise Textfunktionen, Datums- und Zeitfunktionen, mathematische Funktionen und vieles mehr. Ein Teil der Funktionen wird in diesem Kapitel vorgestellt. Die anderen folgen sukzessive in den nächsten Kapiteln.

- Datums- und Zeitfunktionen
- Textfunktionen
- Dateifunktionen und -anweisungen
- Mathematische Funktionen

3.1 Datums- und Zeitfunktionen

Um vorab eine Übersicht über diese Funktionen zu bekommen, werfen Sie einen Blick auf die Tabelle 3.1. Danach erfolgen die Erklärung der Syntax und die Demonstration anhand eines kurzen Beispiels.

Funktion	Kurzbeschreibung
CDate	Wandelt eine Zeichenfolge in einen Datumswert um.
Date	Gibt das aktuelle Systemdatum aus.
DateAdd	Liefert einen Wert zurück, der ein Datum enthält, zu dem ein bestimmtes Zeitintervall addiert wurde.
DateDiff	Liefert einen Wert zurück, der die Anzahl der Zeitintervalle zwischen zwei bestimmten Terminen angibt.
DatePart	Liefert einen Wert zurück, der einen bestimmten Teil eines angegebenen Datums enthält.
DateSerial	Liefert einen Wert zurück, der die angegebene Jahres-, Monats- und Tageszahl enthält.
DateValue	Wandelt eine Zeichenfolge in einen gültigen Datumswert um.
Day	Extrahiert den Tag als ganzzahligen Wert (1-31) aus einem Datumswert.

Tabelle 3.1: Datums- und Zeitfunktionen

Funktion	Kurzbeschreibung
FileDateTime	Liefert das Erstellungsdatum bzw. das letzte Änderungsdatum einer Datei.
FormatDateTime	Gibt einen als Datum oder Uhrzeit formatierten Ausdruck zurück.
Hour	Liefert die Stunde aus einem Datums-/Zeitwert als ganzzahligen Wert (0-23).
Minute	Liefert die Minute aus einem Datums-/Zeitwert als ganzzahligen Wert (0-59).
Month	Extrahiert den Monat aus einem Datumswert als ganzzahligen Wert (1-12).
Monthname	Gibt eine Zeichenfolge zurück, die den festgelegten Monat angibt.
Now	Liefert das Systemdatum inklusive der Uhrzeit.
Second	Liefert die Sekunde aus einem Datums-/Zeitwert als ganzzahligen Wert (0-59).
Time	Gibt die aktuelle Systemzeit aus.
Timer	Gibt einen Wert vom Typ <code>Single</code> zurück, der die Anzahl der seit Mitternacht vergangenen Sekunden angibt.
TimeSerial	Setzt einen Datums-/Zeitwert aus Ganzzahlwerten (Sekunden, Minuten und Stunden) zusammen.
TimeValue	Wandelt eine Zeichenfolge in einen gültigen Zeitwert um.
WeekDay	Gibt den Wochentag aus einem Datumswert zurück (1-7).
WeekDayName	Gibt eine Zeichenfolge zurück, die den festgelegten Wochentag angibt.
Year	Extrahiert die Jahresinformation aus einem Datumswert.

Tabelle 3.1: Datums- und Zeitfunktionen (Forts.)

CDate-Funktion

Mithilfe der Funktion `CDate` können Zeichenfolgen in einen gültigen Datumswert konvertiert werden.

► Syntax

`CDate(Ausdruck)`

Im Argument `Ausdruck` wird eine Zeichenfolge angegeben. Der Rückgabewert der Funktion stellt einen gültigen Datumswert dar.

► Beispiele

Im folgenden Beispiel aus Listing 3.1 wird eine Zeichenfolge umgewandelt und in eine Zelle eingefügt.

Listing 3.1: Mit der Funktion CDate Zeichenfolgen in gültige Datumswerte umwandeln

```
Sub CDate_Beispiel()  
Dim s As String  
  
s = "14. März 2004"  
Sheets("Tabelle1").Range("A1").Value = CDate(s)  
End Sub
```

Selbstverständlich funktioniert dasselbe auch mit Zeitwerten:

Listing 3.2: Mit der Funktion CDate Zeichenfolgen in gültige Zeitwerte umwandeln

```
Sub CDate_Beispiel2()  
Dim s As String  
  
s = "18:45:00"  
Sheets("Tabelle1").Range("A2").Value = CDate(s)  
End Sub
```

Dringend gebraucht wird diese Funktion, wenn beispielsweise aus einer Userform ein Datum, das in einem Textfeld erfasst wurde, in einer Tabelle gespeichert werden soll. Ohne die Umwandlung würde das Datum in der Tabelle linksbündig angeordnet, also als Textwert interpretiert werden.

Der folgende Code aus Listing 3.3 wird direkt hinter die Schaltfläche OK/SPEICHERN gelegt.

Listing 3.3: Ein Datum aus einer Userform in eine Tabelle zurückschreiben

```
Private Sub CommandButton1_Click()  
Sheets("Tabelle1").Range("A3").Value = _  
CDate(TextBox1.Value)  
Unload Me  
End Sub
```

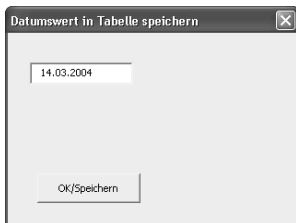


Abbildung 3.1: Das Datum soll korrekt in eine Tabellenzelle geschrieben werden.

► Verwandte Funktionen

CInt, CSng, CBool, CByte, CCur, CDb1, CDec, CLng, CVar, CStr

Date-Funktion

Über die Funktion `Date` können Sie das aktuelle Systemdatum, das in der Systemsteuerung von Windows hinterlegt ist, abrufen.

► Syntax

`Date`

Diese Funktion hat keine weiteren Argumente.

► Beispiele

Ein typisches Beispiel für den Gebrauch dieser Funktion ist das Einfügen des aktuellen Tagesdatums in eine Zelle.

Listing 3.4: Das aktuelle Tagesdatum in eine Zelle einfügen

```
Sub Date_Funktion()  
    Sheets("Tabelle1").Range("A4").Value = Date  
End Sub
```

Ein weiteres praktisches Beispiel stellt die Benennung der aktiven Tabelle nach dem aktuellen Tagesdatum dar.

Listing 3.5: Die Tabelle nach dem aktuellen Tagesdatum benennen

```
Sub Date_Funktion2()  
    ActiveSheet.Name = Date  
End Sub
```

► Verwandte Funktionen

Time, Now, Timer, Format

DateAdd-Funktion

Die Funktion `DateAdd` liefert einen Wert vom Typ `Variant (Date)` zurück, der ein Datum enthält, zu dem ein bestimmtes Zeitintervall addiert wurde.

► Syntax

`DateAdd(interval, number, date)`

Die Syntax für die `DateAdd`-Funktion besteht aus den folgenden benannten Argumenten:

Teil	Beschreibung
<code>interval</code>	Zeichenfolgenausdruck, der das zu addierende Zeitintervall ergibt.
<code>number</code>	Numerischer Ausdruck, der die Anzahl der zu addierenden Intervalle ergibt. Er kann positiv (für ein zukünftiges Datum) oder negativ (für ein vergangenes Datum) sein.
<code>date</code>	Ein Wert vom Typ <code>Variant (Date)</code> oder ein als Literal dargestelltes Datum, zu dem das Intervall hinzuaddiert wird.

Tabelle 3.2: Argumente der Funktion `DateAdd`

► Beispiel

Im folgenden Makro aus Listing 3.6 wird, ausgehend von einem Bestelldatum, der Liefertermin errechnet und im Direktfenster der Entwicklungsumgebung ausgegeben.

Listing 3.6: Das Enddatum mit der Funktion `DateAdd` ausrechnen

```
Sub DateAdd_Funktion()  
Dim Datum1 As Date  
Dim sIntervall As String  
Dim Zahl As Integer  
  
Debug.Print "Bestelldatum: " & Date  
sIntervall = "m"
```

```
Datum1 = "14.03.2004"
Zahl = 3
Debug.Print "Lieferdatum: " & _
DateAdd(sIntervall, Zahl, Datum1)
End Sub
```



Abbildung 3.2: Drei Monate später wird geliefert

► Verwandte Funktionen

DateDiff, DatePart

DatePart-Funktion

Über die Funktion `Datepart` können Sie einen bestimmten Teil des Datums extrahieren, indem Sie auf die Konstanten dieser Funktion zurückgreifen.

► Syntax

```
DatePart(interval,date[,firstdayofweek  
[, firstweekofyear]])
```

Im Argument `Intervall` müssen Sie genau angeben, welchen Teil des Datums Sie extrahieren möchten. Die einzelnen Möglichkeiten sehen Sie in den folgenden Möglichkeiten in Tabelle 3.3.

Einstellung	Beschreibung
yyyy	Jahr
q	Quartal
m	Monat
y	Tag des Jahres
d	Tag
w	Wochentag
ww	Woche
h	Stunde

Tabelle 3.3: Intervall-Konstanten der Funktion `DatePart`

Einstellung	Beschreibung
n	Minute
s	Sekunde

Tabelle 3.3: Intervall-Konstanten der Funktion DatePart (Forts.)

Im Argument `Date` geben Sie den Wert an, den Sie berechnen möchten.

Im Argument `firstdayofweek` müssen Sie den ersten Tag der Woche angeben. Denken Sie beispielsweise daran, dass der jüdische Kalender mit dem Sonntag als erstem Tag der Woche beginnt. Für unseren europäischen Bereich müssen Sie daher den Wert 2 bzw. die Konstante `vbMonday` einsetzen. Wenn Sie die ganze Sache etwas variabler halten möchten, dann setzen Sie die Konstante `vbUseSystem` ein. Damit wird die Einstellung des ersten Tags der Woche direkt aus den Einstellungen Ihrer Windows-Systemsteuerung herausgelesen. Sehen Sie die einzelnen Belegungen der Konstanten in Tabelle 3.4.

Konstante	Wert	Beschreibung
<code>VbUseSystem</code>	0	Die NLS API-Einstellung wird verwendet
<code>VbSunday</code>	1	Sonntag (Voreinstellung)
<code>VbMonday</code>	2	Montag
<code>vbTuesday</code>	3	Dienstag
<code>vbWednesday</code>	4	Mittwoch
<code>vbThursday</code>	5	Donnerstag
<code>vbFriday</code>	6	Freitag
<code>vbSaturday</code>	7	Samstag

Tabelle 3.4: FirstDayOfWeek-Konstanten der Funktion DatePart

Im letzten Argument `firstweekofyear` legen Sie die erste Woche eines Jahres fest. Danach richtet sich auch jeweils die Nummerierung der Kalenderwoche. Dabei können Sie folgende Einstellungen treffen:

Konstante	Wert	Beschreibung
VbUseSystem	0	Die NLS API-Einstellung aus der Systemsteuerung von Windows wird verwendet.
vbFirstJan1	1	Anfang in der Woche mit dem 1. Januar (Voreinstellung)
vbFirstFourDays	2	Anfang in der ersten Woche, die mindestens vier Tage im neuen Jahr enthält
VbFirstFullWeek	3	Anfang in der ersten vollen Woche des Jahres

Tabelle 3.5: FirstWeekOfYear-Konstanten der Funktion DatePart

► Beispiele

Im folgenden Beispiel aus Listing 3.7 wird, ausgehend von einem Datum, das dazugehörige Quartal ausgegeben.

Listing 3.7: Das Quartal eines Datums wird ausgegeben

```
Sub DatePart_Funktion()
Dim Datum1 As Date

Datum1 = "14.03.2004"
Debug.Print "Das Datum " & Datum1 & vbLf & _
" liegt im Quartal: " & DatePart("q", Datum1)
End Sub
```

Übergeben Sie der Funktion `Datepart` das Intervall `q`, um die Quartalskennziffer eines Datums zu ermitteln.

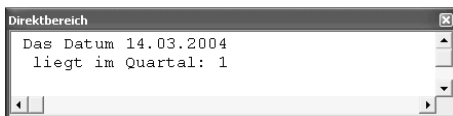


Abbildung 3.3: Quartalsermittlung mit DatePart

Beim nächsten Beispiel aus Listing 3.8 soll anhand des aktuellen Tagesdatums die dazugehörige Wochennummer ermittelt werden.

Listing 3.8: Die Wochennummer aus einem Datum ermitteln

```
Sub DatePart_Funktion2()  
Debug.Print "Heute ist der " & Date  
Debug.Print "Wir befinden uns in der Woche " & _  
    DatePart("ww", Date)  
End Sub
```

Übergeben Sie der Funktion `Datepart` das Intervall `ww`, um die Wochennummer eines Datums zu ermitteln.

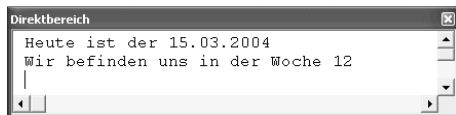


Abbildung 3.4: Ermittlung der Wochennummer, ebenfalls über die Funktion `DatePart`

► Verwandte Funktionen

`DateAdd`, `DateDiff`

DateDiff-Funktion

Mithilfe der Funktion `DateDiff` können Sie die Anzahl der Zeitintervalle, die zwischen zwei bestimmten Terminen liegen, angeben.

► Syntax

```
DateDiff(interval, date1, date2[, firstdayofweek  
[, firstweekofyear]])
```

Da die Argumente der Funktion `DateDiff` fast gleich sind wie bei der Funktion `DatePart`, lesen Sie weiter oben im Kapitel nach.

► Beispiel

Beim folgenden Beispiel aus Listing 3.9 soll die Differenz in Tagen vom aktuellen Tag bis zum Jahresende errechnet werden.

Listing 3.9: Datumsdifferenzen errechnen über die Funktion DateDiff

```
Sub DateDiff_Funktion()  
Dim StartDatum As Date  
Dim EndDatum As Date  
  
StartDatum = Date  
EndDatum = "31.12.2004"  
Debug.Print "Anzahl Tage von heute " & _  
    StartDatum & vbCrLf & " bis zum Jahresende: " & _  
    DateDiff("d", StartDatum, EndDatum)  
End Sub
```

Übergeben Sie der Funktion `DateDiff` das Intervall `d`, um die Differenz der beiden Datumsangaben in Tagen auszugeben.

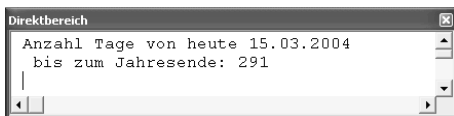


Abbildung 3.5: Die Tage bis zum Jahresende berechnen

► Verwandte Funktionen

`DateAdd`, `DatePart`

DateSerial-Funktion

Die Funktion `DateSerial` liefert einen Wert zurück, der die angegebene Jahres-, Monats- und Tageszahl enthält.

► Syntax

`DateSerial(year, month, day)`

Teil	Beschreibung
year	Wert vom Typ Integer. Zahl im Bereich von 100 bis 9999 (einschließlich) oder ein numerischer Ausdruck.
month	Wert vom Typ Integer. Beliebiger numerischer Ausdruck.
day	Wert vom Typ Integer. Beliebiger numerischer Ausdruck.

Tabelle 3.6: Argumente der Funktion `DateSerial`

► Beispiel

Beim folgenden Beispiel aus Listing 3.10 soll für einen Monat für jeden Tag genau ein Tabellenblatt angelegt werden.

Listing 3.10: Die Funktion DateSerial zum Benennen von Excel-Tabellen heranziehen

```
Sub TabellenNamen()  
Dim i As Integer  
  
Workbooks.Add  
For i = 29 To 1 Step -1  
    Worksheets.Add  
    ActiveSheet.Name = Format(DateSerial _  
        (2004, 2, i), "dd.mm.yy")  
Next i  
End Sub
```

Übergeben Sie der Funktion DateSerial einzeln und nacheinander die drei benötigten Argumente Jahr, Monat und Tag. Der Tag wird in diesem Beispiel über eine dynamische Variable übergeben.

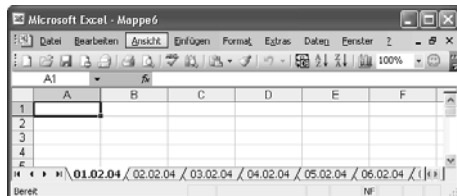


Abbildung 3.6: Schnell Tabellen einfügen und benennen

► Verwandte Funktionen

DateValue, TimeSerial, TimeValue

DateValue-Funktion

Die Funktion `DateValue` wandelt eine Zeichenfolge in einen gültigen Datumswert um.

► Syntax

`DateValue(Datum)`

Das erforderliche Argument `Datum` ist normalerweise ein Zeichenfolgenausdruck, der ein Datum aus dem Bereich vom 1. Januar 100 bis zum 31. Dezember 9999 repräsentiert. Alternativ können Sie für `Datum` aber auch einen beliebigen Ausdruck angeben, der ein Datum, eine Zeit oder beides aus diesem Bereich darstellen kann.

► Beispiele

Beim folgenden Beispiel aus Listing 3.11 wird ein etwas ungewöhnliches Datum in ein gängiges Datumsformat umgesetzt.

Listing 3.11: Datumskonvertierung über die Funktion `DateValue` durchführen

```
Sub DateValue_Beispiel()  
Dim Datum1 As Date  
  
Datum1 = DateValue("1/1/04")  
MsgBox Datum1  
End Sub
```

Wenden Sie die Funktion `DateValue` auch an, um Datumsformate umzusetzen.

Im nächsten Beispiel aus Abbildung 3.7 wird eine Prüfung durchgeführt, ob ein 30-Tage-Zeitraum bereits überschritten wurde.



Abbildung 3.7: Prüfzeitraum überschritten?

Legen Sie das Makro aus Listing 3.12 direkt hinter die Schaltfläche PRÜFEN.

Listing 3.12: Abfrage mithilfe von DateValue, ob ein Prüfzeitraum bereits abgelaufen ist

```
Private Sub CommandButton1_Click()  
If Date > DateValue(TextBox1.Value) + 30 Then  
    MsgBox "Abgelaufen"  
Else  
    MsgBox "Noch gültig"  
End If  
End Sub
```

Vergleichen Sie das aktuelle Tagesdatum mit dem Wert, der in der TEXTBOX1 einer Userform steht, und reagieren Sie je nach Fall.

► Verwandte Funktionen

DateSerial, TimeSerial, TimeValue

Day-Funktion

Mit der Funktion Day können Sie den Tag als ganzzahligen Wert (1-31) aus einem Datumswert extrahieren.

► Syntax

Day(Datum)

Das erforderliche Argument Datum ist ein beliebiger Wert vom Typ Variant, ein numerischer Ausdruck, ein Zeichenfolgenausdruck oder eine beliebige Kombination, die ein Datum darstellen kann.

► Beispiele

Im folgenden Beispiel aus Listing 3.13 wird ein Datum zerlegt und anschließend wieder zusammengesetzt.

Listing 3.13: Datumswerte zerlegen und in anderer Form wieder zusammensetzen

```
Sub Day_Funktion()
Dim Datum1 As Date
Dim iTag As Integer
Dim iMonat As Integer
Dim iJahr As Integer

Datum1 = Date
iTag = Day(Datum1)
iMonat = Month(Datum1)
iJahr = Year(Datum1)

Debug.Print "Aus: " & Datum1
Debug.Print "wird: " & iTag & "-" & iMonat & _
 "-" & iJahr
End Sub
```

Analog zur Funktion `Day` werden hier die Funktionen `Month` und `Year` eingesetzt, um die jeweiligen Datumsinformationen (Tag, Monat und Jahr) zu extrahieren.



Abbildung 3.8: Mithilfe der Funktion `Day` den Tag aus einem Datum extrahieren

► Verwandte Funktionen

`Month`, `Year`, `Now`, `Date`, `Time`

FileDateTime-Funktion

Die Funktion `FileDateTime` liefert das Erstellungsdatum bzw. das letzte Änderungsdatum einer Datei, ohne dass die Datei geöffnet sein muss.

► Syntax

`FileDateTime(Pfadname)`

Das erforderliche Argument `Pfadname` ist ein Zeichenfolgenausdruck, der einen Dateinamen angibt. `Pfadname` kann ein Verzeichnis oder einen Ordner sowie ein Laufwerk enthalten.

► Beispiele

Beim folgenden Beispiel aus Listing 3.14 wird das letzte Änderungsdatum der aktiven Arbeitsmappe in die erste Tabelle der Mappe geschrieben.

Listing 3.14: Das letzte Änderungsdatum einer Mappe ermitteln und in die Zelle schreiben

```
Sub FileDateTime_Beispiel()
    ThisWorkbook.Save
    Sheets(1).Range("A5").Value = _
        FileDateTime(ThisWorkbook.FullName)
End Sub
```

Über die Eigenschaft `FullName` können Sie den kompletten Pfadnamen einer Mappe ermitteln und an die Funktion `FileDateTime` übergeben.

Soll diese Information anstatt in eine Zelle in die Fußzeile der Tabelle geschrieben werden, dann lautet das Makro für diesen Zweck wie folgt:

Listing 3.15: Das letzte Änderungsdatum einer Mappe ermitteln und in die Fußzeile schreiben

```
Sub FileDateTime_Beispiel2()
    ThisWorkbook.Save
    Sheets(1).PageSetup.LeftFooter = _
        FileDateTime(ThisWorkbook.FullName)
End Sub
```

► **Verwandte Funktionen**

FileLen, GettAttr

FormatDateTime-Funktion

Für das Anzeigen von Datums- und Zeitangaben stehen Ihnen fertige Systemkonstanten zur Verfügung, die die Formatierung des Datums bzw. des Zeitwerts für Sie übernehmen.

Diese Datums-/Zeitkonstanten werden im Zusammenspiel mit der Funktion `FormatDateTime` verwendet.

► **Syntax**

```
FormatDateTime(Datum[, BenanntesFormat])
```

Im Argument `Datum` übergeben Sie der Funktion einen Datumswert.

Im Argument `BenanntesFormat` wählen Sie eine der in der Tabelle folgenden Datums-/Zeitkonstanten.

Konstante	Wert	Beschreibung
<code>vbGeneralDate</code>	0	Zeigt ein Datum und/oder eine Uhrzeit an. Wenn es ein Datum gibt, wird es in Kurzform angezeigt. Wenn es eine Uhrzeit gibt, wird sie im langen Format angezeigt. Falls vorhanden werden beide Teile angezeigt.
<code>vbLongDate</code>	1	Zeigt ein Datum im langen Datumsformat an, das in den Ländereinstellungen des Computers festgelegt ist.
<code>VbShortDate</code>	2	Zeigt ein Datum im kurzen Datumsformat an, das in den Ländereinstellungen des Computers festgelegt ist.
<code>vbLongTime</code>	3	Zeigt eine Uhrzeit in dem Zeitformat an, das in den Ländereinstellungen des Computers festgelegt ist.
<code>vbShortTime</code>	4	Zeigt eine Uhrzeit im 24-Stunden-Format (hh:mm) an.

Tabelle 3.7: Datumskonstanten

► Beispiel

Im folgenden Beispiel aus Listing 3.16 wird ein Datum auf unterschiedlichste Art und Weise im Direktfenster der Entwicklungsumgebung ausgegeben.

Listing 3.16: Verschiedene Datumsformat über die Funktion FormatDateTime

```
Sub FormatDateTime_Beispiel()  
Dim Datum1 As Date  
  
Datum1 = Now  
Debug.Print FormatDateTime(Datum1, vbGeneralDate)  
Debug.Print FormatDateTime(Datum1, vbLongDate)  
Debug.Print FormatDateTime(Datum1, vbShortDate)  
Debug.Print FormatDateTime(Datum1, vbLongTime)  
Debug.Print FormatDateTime(Datum1, vbShortTime)  
End Sub
```

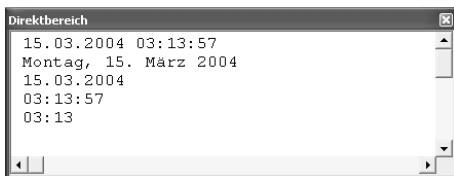


Abbildung 3.9: Unterschiedliche Möglichkeiten, ein Datum zu formatieren

► Verwandte Funktionen

Format, FormatCurrency, FormatNumber, FormatPercent

Hour-Funktion

Die Funktion `Hour` liefert die Stunde aus einem Datums-/Zeitwert als ganzzahligen Wert (0-23).

► Syntax

`Hour(Uhrzeit)`

Das erforderliche Argument `Uhrzeit` ist ein beliebiger Wert vom Typ `Variant`, ein numerischer Ausdruck, ein Zeichenfolgenausdruck oder eine beliebige Kombination, die eine Uhrzeit darstellen kann.

► Beispiele

Im folgenden Beispiel aus Listing 3.17 wird eine Zeitverzögerung in Excel eingebaut. Dabei soll die Verarbeitung für genau 10 Sekunden unterbrochen werden.

Listing 3.17: Die Makroverarbeitung für 10 Sekunden unterbrechen

```
Sub Hour_Beispiel()  
Dim dStunde As Date  
Dim dMinute As Date  
Dim dSekunde As Date  
Dim dPause As Date  
  
MsgBox "Bitte OK klicken, dann 10 Sekunden warten!"  
dStunde = Hour(Now())  
dMinute = Minute(Now())  
dSekunde = Second(Now()) + 10  
dPause = TimeSerial(dStunde, dMinute, dSekunde)  
Application.Wait dPause  
MsgBox "Die 10 Sekunden sind abgelaufen", _  
vbInformation  
End Sub
```

Speichern Sie zunächst die einzelnen Zeitwerte, indem Sie die Funktionen `Hour`, `Minute` und `Second` einsetzen. Addieren Sie zu der letzten Zeitanzeige den Wert 10, um die Makroverarbeitung um 10 Sekunden zu unterbrechen, und definieren Sie somit die Pausenzeit mithilfe der Funktion `TimeSerial`. Übergeben Sie diese Pausenzeit der Methode `Wait`.

Eine ähnliche Aufgabenstellung liefert das Makro aus Listing 3.18. Dabei wird ein Bild von der Festplatte für 3 Sekunden in eine Tabelle eingefügt und dann wieder entfernt.

Listing 3.18: Ein Logo für 3 Sekunden einfügen und dann wieder entfernen

```
Sub Hour_Beispiel2()  
Dim Pic As Picture  
Dim dStunde As Date  
Dim dMinute As Date  
Dim dSekunde As Date  
Dim dPause As Date  
  
Sheets("Tabelle1").Pictures.Insert ("c:\Excel.jpg")  
dStunde = Hour(Now())  
dMinute = Minute(Now())  
dSekunde = Second(Now()) + 3  
dPause = TimeSerial(dStunde, dMinute, dSekunde)  
Application.Wait dPause  
Sheets("Tabelle1").Pictures(1).Delete  
End Sub
```

Wenden Sie die Methode `Insert` an, um ein Bild einzufügen. Speichern Sie danach die einzelnen Zeitwerte und addieren beim Sekundenwert den Wert 3 (Sekunden) darauf. Bilden Sie daraus über die Funktion `TimeSerial` die Pausenzeit und lassen Sie Excel mithilfe der Methode `Wait` warten. Löschen Sie nach der Pause das gerade eingefügte Bild über die Methode `Delete`.

► **Verwandte Funktionen**

Minute, Second, TimeSerial, Now, Time, Timer

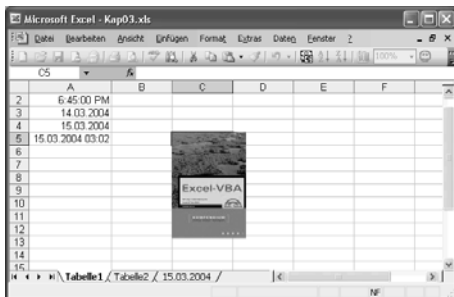


Abbildung 3.10: Eine Grafik einfügen und wieder verschwinden lassen

Minute-Funktion

Die Funktion `Minute` liefert die Minute aus einem Datums-/Zeitwert als ganzzahligen Wert (0-59).

► Syntax

`Minute(Uhrzeit)`

Das erforderliche Argument `Uhrzeit` ist ein beliebiger Wert vom Typ Variant, ein numerischer Ausdruck, ein Zeichenfolgenausdruck oder eine beliebige Kombination, die eine Uhrzeit darstellen kann.

► Beispiel

Sehen Sie sich dazu die Beispiele der Funktion `Hour` an.

► Verwandte Funktionen

`Hour`, `Second`, `Time`, `Timer`, `Date`, `Day`, `Month`, `Year`

Month-Funktion

Die Funktion `Month` extrahiert den Monat aus einem Datumswert als ganzzahligen Wert (1-12).

► Syntax

`Month(Datum)`

Das erforderliche Argument `Datum` ist ein beliebiger Wert vom Typ `Variant`, ein numerischer Ausdruck, ein Zeichenfolgenausdruck oder eine beliebige Kombination, die ein Datum darstellen kann.

► Beispiel

Im nächsten Beispiel aus Listing 3.19 wird die Aktivierung einer Tabelle abhängig vom Tagesdatum vorgenommen. Dazu existieren in einer Mappe 12 Tabellen, von denen die jeweilige Monatstabelle aktiviert wird.

Listing 3.19: Tabellen aktivieren in Abhängigkeit vom Monat

```
Sub Month_Beispiel()  
Dim i As Integer  
  
On Error GoTo Fehler  
    i = Month(Now())  
    Sheets("Tabelle" & i).Activate  
Exit Sub  
Fehler:  
MsgBox "Tabelle konnte nicht gefunden werden!", _  
    vbCritical  
End Sub
```

► Verwandte Funktionen

`Day`, `Year`, `Date`

MonthName-Funktion

Die Funktion `MonthName` gibt eine Zeichenfolge zurück, die den festgelegten Monat angibt.

► Syntax

`MonthName(Monat[, abkürzen])`

Die Syntax der `MonthName`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Monat	Die numerische Bezeichnung des Monats, z.B. 1 für Januar, 2 für Februar usw.
Abkürzen	Optionales Argument. Boolescher Wert, der angibt, ob der Monatsname abgekürzt wird. Wird er ausgelassen, ist die Standardeinstellung <code>False</code> , d.h. der Monatsname wird nicht abgekürzt.

Tabelle 3.8: Argumente der Funktion MonthName

► Beispiel

Im folgenden Beispiel aus Listing 3.20 wird der Monatsname aus dem aktuellen Tagesdatum ermittelt und im Direktbereich der Entwicklungsumgebung ausgegeben.

Listing 3.20: Die Funktion MonthName ermittelt den Monatsnamen

```
Sub MonthName_Beiispiel()
    Debug.Print "verkürzt: " & _
        MonthName(Month(Date), True)
    Debug.Print "normal: " & _
        MonthName(Month(Date), False)
End Sub
```

► Verwandte Funktionen

`Date`, `Month`, `Day`, `Format`



Abbildung 3.11: Die Ermittlung des Monatsnamens auf zwei unterschiedliche Arten durchgeführt

Now-Funktion

Die Funktion `Now` liefert das Systemdatum inklusive der Uhrzeit.

► Syntax

`Now` (ohne weitere Argumente)

► Beispiel

Im folgenden Beispiel aus Listing 3.21 wird eine neue Arbeitsmappe angelegt, unter dem heutigen Tagesdatum abgespeichert und wieder geschlossen.

Listing 3.21: Arbeitsmappe unter Tagesdatum speichern

```
Sub Now_Beiispiel()  
    Dim sDatum As String  
    Dim SName As String  
  
    Workbooks.Add  
    sDatum = Application.Text(Now(), "mm-dd-yy")  
    SName = ActiveWorkbook.Name  
    ActiveWorkbook.SaveAs SName & sDatum & ".xls"  
    ActiveWorkbook.Close  
End Sub
```

Über die Funktion `Now` ermitteln Sie das aktuelle Tagesdatum sowie die aktuelle Uhrzeit. Danach speichern Sie die Mappe mithilfe der Methode `SaveAs` unter dem neuen Namen ab.

► Verwandte Funktionen

`Date`, `Time`, `Timer`, `Hour`, `Minute`, `Second`, `Month`, `Year`

Second-Funktion

Die Funktion `Second` liefert die Sekunde aus einem Datums-/Zeitwert als ganzzahligen Wert (0-59).

► Syntax

`Second(Uhrzeit)`

Das erforderliche Argument `Uhrzeit` ist ein beliebiger Wert vom Typ `Variant`, ein numerischer Ausdruck, ein Zeichenfolgenausdruck oder eine beliebige Kombination, die eine Uhrzeit darstellen kann.

► Beispiel

Sehen Sie sich dazu die Beispiele der Funktion `Hour` an.

► Verwandte Funktionen

`Minute`, `Second`, `Time`, `Timer`, `Date`, `Day`, `Month`, `Year`

Time-Funktion

Die Funktion `Time` gibt die aktuelle Systemzeit aus.

► Syntax

`Time` (ohne weitere Argumente)

► Beispiel

Beim folgenden Beispiel aus Listing 3.22 wird jede Minute die aktuelle Uhrzeit in die Statusleiste von Excel geschrieben.

Listing 3.22: Die aktuelle Uhrzeit in die Statusleiste schreiben

```
Sub Zeitmakro()
    Application.OnTime Now + _
        TimeValue("00:01:00"), "Time_Beispiel"
End Sub

Sub Time_Beispiel()
    Application.StatusBar = Date & ", " & Time
    Call Zeitmakro
End Sub
```

Starten Sie das Makro `Zeitmakro` und warten Sie eine Minute. Danach wird regelmäßig das Makro `Time_Beispiel` aufgerufen, welches die Uhrzeit in die Statusleiste von Excel schreibt.

► Verwandte Funktionen

`Date`, `TimeValue`, `Timer`

Timer-Funktion

Die Funktion `Timer` gibt einen Wert vom Typ `Single` zurück, der die Anzahl der seit Mitternacht vergangenen Sekunden angibt.

► Syntax

`Timer` (ohne weitere Argumente)

► Beispiel

Im folgenden Beispiel aus Listing 3.23 wird ein Zellenbereich abwechselnd mit diversen Hintergrundfarben formatiert. Zwischen den einzelnen Farbvarianten wird eine Pause von 2 Sekunden eingelegt.

Listing 3.23: Bereich unterschiedlich einfärben mit Zeitverzögerung

```
Sub Pause(i)
Dim time As Single
    time = Timer
    Do While Timer < time + i
        DoEvents
    Loop
End Sub

Sub Timer_Beispiel()
Dim i As Integer
For i = 2 To 5
    Sheets("Tabelle1"). _
        Range("A1:A10").Interior.ColorIndex = i
    Pause 2
Next
Sheets("Tabelle1"). _
    Range("A1:A10").Interior.ColorIndex = _
xlColorIndexNone
End Sub
```

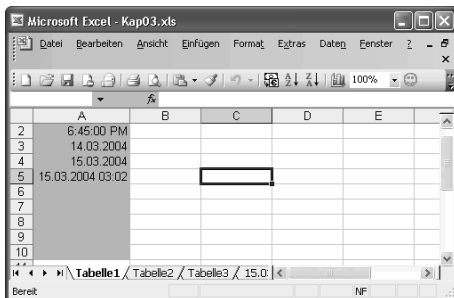


Abbildung 3.12: Zellenbereich färben

► Verwandte Funktionen

Time, Now

TimeSerial-Funktion

Die Funktion `TimeSerial` setzt einen Datums-/Zeitwert aus Ganzzahlwerten (Sekunden, Minuten und Stunden) zusammen.

► Syntax

`TimeSerial(hour, minute, second)`

Die Syntax für die `TimeSerial`-Funktion besteht aus folgenden benannten Argumenten:

Teil	Beschreibung
Hour	Variant (Integer). Zahl im Bereich von 0 (00:00) bis 23 (23:00) oder ein numerischer Ausdruck.
Minute	Variant (Integer). Beliebiger numerischer Ausdruck.
Second	Variant (Integer). Beliebiger numerischer Ausdruck.

Tabelle 3.9: Argumente der Funktion `TimeSerial`

► Beispiel

Im folgenden Beispiel aus Listing 3.24 werden verschiedene Zahlenwerte aus einer Tabelle zu einem gültigen Zeitwert zusammengesetzt.

Listing 3.24: Einen gültigen Zeitwert zusammenbasteln

```

Sub TimeSerial_Beispiel()
Dim Zeit1 As Date

With Sheets("Tabelle2")
    Zeit1 = TimeSerial(.Cells(1, 1), _
        .Cells(1, 2), .Cells(1, 3))
    .Cells(2, 1).Value = Zeit1
End With
End Sub

```

Übergeben Sie der Funktion `TimeSerial` die einzelnen Argumente und geben diese am Ende in der Zelle A2 aus.

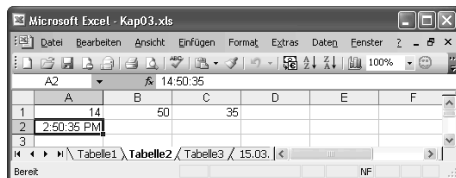


Abbildung 3.13: Aus drei Zelleninhalten wird ein gültiger Zeitwert

► Verwandte Funktionen

`DateSerial`, `DateValue`, `TimeValue`, `Hour`, `Minute`, `Second`, `Now`

TimeValue-Funktion

Die Funktion `TimeValue` wandelt eine Zeichenfolge in einen gültigen Zeitwert um.

► Syntax

`TimeValue(Uhrzeit)`

Das erforderliche Argument `Uhrzeit` ist normalerweise ein Zeichenfolgenausdruck, der eine Zeit im Bereich von 0:00:00 bis 23:59:59 darstellt. `Uhrzeit` kann aber auch ein beliebiger Ausdruck sein, der eine Zeit aus diesem Bereich angibt.

► Beispiele

Beim folgenden Beispiel aus Listing 3.25 wird ein Zeichenfolgenausdruck in einen gültigen Zeitwert umgewandelt.

Listing 3.25: Zeichenfolge in einen gültigen Zeitwert wandeln

```
Sub TimeValue_Beispiel()  
Dim Zeit1 As Date
```

```
Zeit1 = TimeValue("3:21:59 PM")  
Debug.Print Zeit1  
End Sub
```

Beim nächsten Beispiel aus Listing 3.26 wird ein bestimmtes Makro alle 60 Sekunden automatisch gestartet.

Listing 3.26: Makro wird alle 60 Sekunden automatisch gestartet

```
Sub Beginn()  
Zeit1 = Now  
TimeValue_Beispiel2  
End Sub
```

```
Sub TimeValue_Beispiel2()  
Debug.Print Format(Zeit1, "hh:mm:ss")  
Zeit1 = Zeit1 + TimeValue("00:01:00")  
Application.OnTime Zeit1, "Beginn"  
End Sub
```

Wenn Sie das Makro `Beginn` starten, dann wird das Makro `TimeValue_Beispiel2` nach jeweils 60 Sekunden ausgeführt. Dabei wird mithilfe der Anweisung `Debug.Print` immer die aktuelle Uhrzeit in das Direktfenster der Entwicklungsumgebung geschrieben.

► Verwandte Funktionen

`DateSerial`, `DateValue`, `TimeSerial`, `Hour`, `Minute`, `Second`, `Now`

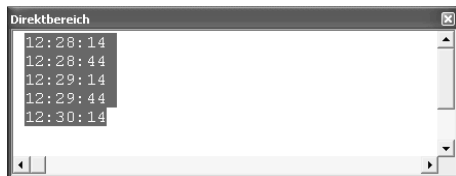


Abbildung 3.14: Alle 60 Sekunden wird ein »Satz« geschrieben

Weekday-Funktion

Die Funktion `Weekday` gibt den Wochentag aus einem Datumswert zurück (1-7).

► Syntax

`Weekday(date, [firstdayofweek])`

Die Syntax für die `Weekday`-Funktion besteht aus folgenden benannten Argumenten:

Teil	Beschreibung
Date	Erforderlich. Wert vom Typ <code>Variant</code> , numerischer Ausdruck, Zeichenfolgenausdruck oder eine beliebige Kombination, die ein Datum darstellen kann. Wenn <code>Date</code> den Wert <code>Null</code> enthält, wird <code>Null</code> zurückgegeben.
Firstdayofweek	Optional. Eine Konstante, die den ersten Wochentag angibt. Wird die Konstante nicht angegeben, so wird <code>vbSunday</code> angenommen.

Tabelle 3.10: Argumente der Funktion `Weekday`

Das Argument `firstdayofweek` hat folgende Einstellungen:

Konstante	Wert	Beschreibung
<code>vbUseSystem</code>	0	NLS API-Einstellung wird verwendet
<code>VbSunday</code>	1	Sonntag (Voreinstellung)
<code>vbMonday</code>	2	Montag
<code>vbTuesday</code>	3	Dienstag
<code>vbWednesday</code>	4	Mittwoch

Tabelle 3.11: Argumente von `Firstdayofweek`

Konstante	Wert	Beschreibung
vbThursday	5	Donnerstag
vbFriday	6	Freitag
vbSaturday	7	Samstag

Tabelle 3.11: Argumente von Firstdayofweek (Forts.)

► Beispiel

Im folgenden Listing 3.27 werden, ausgehend von einem vorgegebenen Datum, der dazugehörige Wochentag sowie das Quartal ermittelt.

Listing 3.27: Den Wochentag ermitteln

```
Function Quartal(DatAngabe)
    Quartal = DatePart("q", DatAngabe)
End Function

Function Wochentag(DatAngabe)
    Wochentag = WeekdayName _
        (Weekday(DatAngabe, vbUseSystemDayOfWeek), False)
End Function

Sub Weekday_Beispiel()
    Debug.Print " Dieses Datum liegt im " & _
        Quartal("20.04.2004") & " .Quartal!"
    Debug.Print " Der Wochentag ist ein " & _
        Wochentag("20.04.2004")
End Sub
```

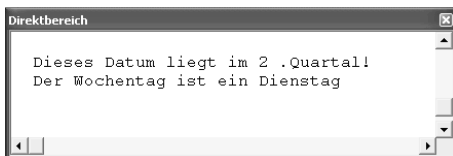


Abbildung 3.15: Wochentag und Quartal ermitteln

► Verwandte Funktionen

WeekdayName, Date, Day, Month, Year, Now

WeekdayName-Funktion

Die Funktion `WeekdayName` gibt eine Zeichenfolge zurück, die den festgelegten Wochentag angibt.

► Syntax

`WeekdayName(Wochentag, abkürzen, ErsterWochentag)`

Die Syntax der `WeekdayName`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Wochentag	Erforderlich. Die numerische Bezeichnung des Wochentags. Der numerische Wert der einzelnen Tage hängt von der Einstellung für <code>ErsterWochentag</code> ab.
Abkürzen	Optional. Boolescher Wert, der angibt, ob der Name des Wochentags abgekürzt wird. Wird er ausgelassen, ist die Standardeinstellung <code>False</code> , d.h. der Name des Wochentags wird nicht abgekürzt.
ErsterWochentag	Optional. Numerischer Wert, der den ersten Tag der Woche angibt (siehe Tabelle 3.11).

Tabelle 3.12: Argumente der Funktion `WeekdayName`

► Beispiel

Im folgenden Beispiel aus Listing 3.28 wird der heutige Tag in Kurzform und der morgige Tag in ausgeschriebener Syntax im Direktfenster der Entwicklungsumgebung ausgegeben.

Listing 3.28: Den Wochentag von heute und morgen ausgeben

```
Sub WeekdayName_Beispiel()
    Debug.Print "Heute ist " & _
        WeekdayName(Weekday(Date), True)
    Debug.Print "Morgen ist " & _
        WeekdayName(Weekday(Date + 1), False)
End Sub
```

► Verwandte Funktionen

`Weekday`, `Date`, `Day`, `Month`, `Year`, `Now`

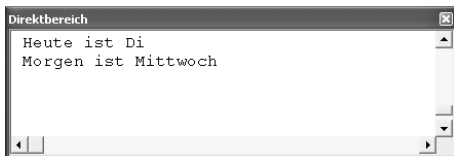


Abbildung 3.16: Wochentag über die Funktion `WeekdayName` bestimmen

Year-Funktion

► Syntax

`Year(Datum)`

Das erforderliche Argument `Datum` ist ein beliebiger Wert vom Typ `Variant`, ein numerischer Ausdruck, ein Zeichenfolgenausdruck oder eine beliebige Kombination, die ein Datum darstellen kann.

► Beispiel

Sehen Sie sich dazu die Beispiele der Funktion `Month` und `Day` an.

3.2 Textfunktionen

Um vorab einen Überblick über diese Funktionen zu bekommen, werfen Sie einen Blick auf die Tabelle. Danach erfolgt die Erklärung der Syntax und die Demonstration anhand eines kurzen Beispiels.

Funktion	Kurzbeschreibung
<code>Asc</code>	Gibt einen Wert vom Typ <code>Integer</code> zurück, der den Zeichencode entsprechend dem ersten Buchstaben in einer Zeichenfolge darstellt.
<code>Choose</code>	Wählt einen Wert aus einer Liste von Argumenten aus und gibt ihn zurück.
<code>Chr</code>	Gibt einen Wert vom Typ <code>String</code> zurück, der das Zeichen enthält, das dem angegebenen Zeichen-Code zugeordnet ist.

Tabelle 3.13: Textfunktionen

Funktion	Kurzbeschreibung
InStr	Gibt einen Wert vom Typ Variant (Long) zurück, der die Position des ersten Auftretens einer Zeichenfolge innerhalb einer anderen Zeichenfolge angibt.
InStrRev	Gibt die Position eines Vorkommnisses einer Zeichenfolge in einer anderen Zeichenfolge, vom Ende der Zeichenfolge gesehen, an.
Join	Gibt eine Zeichenfolge zurück, die sich aus der Kombination einer Reihe von untergeordneten Zeichenfolgen, die in einem Datenfeld enthalten sind, ergibt.
LCase	Wandelt Großbuchstaben in Kleinbuchstaben um.
Left	Gibt einen Wert vom Typ Variant (String) zurück, der eine bestimmte Anzahl von Zeichen ab dem ersten (linken) Zeichen einer Zeichenfolge enthält.
Len	Gibt einen Wert vom Typ Long zurück, der die Anzahl der Zeichen in einer Zeichenfolge oder die zum Speichern einer Variablen erforderlichen Bytes enthält.
Mid	Gibt einen Wert vom Typ Variant (String) zurück, der eine bestimmte Anzahl von Zeichen aus einer Zeichenfolge enthält.
Replace	Gibt eine Zeichenfolge zurück, in der eine festgelegte, untergeordnete Zeichenfolge mit einer festgelegten Häufigkeit durch eine andere untergeordnete Zeichenfolge ersetzt wurde.
Right	Gibt einen Wert vom Typ Variant (String) zurück, der eine bestimmte Anzahl von Zeichen von der rechten Seite (dem Ende) einer Zeichenfolge enthält.
Space	Gibt eine Zeichenfolge vom Typ Variant (String) zurück, die aus einer angegebenen Anzahl von Leerzeichen besteht.
Spc	Fügt in einer Textdatei eine bestimmte Anzahl von Leerzeichen ein.
Split	Gibt ein nullbasiertes, eindimensionales Datenfeld zurück, das eine festgelegte Anzahl an untergeordneten Zeichenfolgen enthält.
Str	Gibt einen Wert vom Typ Variant (String) zurück, der eine Zahl darstellt.
StrComp	Gibt einen Wert vom Typ Variant (Integer) zurück, der das Ergebnis eines Zeichenfolgenvergleichs anzeigt.

Tabelle 3.13: Textfunktionen (Forts.)

Funktion	Kurzbeschreibung
StrConv	Gibt einen Wert vom Typ Variant (String) zurück, der wie angegeben umgewandelt wurde.
StrReverse	Gibt eine Zeichenfolge zurück, in der die Reihenfolge der Zeichen einer bestimmten Zeichenfolge umgekehrt wurde.
String	Gibt eine Zeichenfolge vom Typ Variant (String) zurück, die ein sich wiederholendes Zeichen der angegebenen Länge enthält.
Switch	Wertet eine Liste von Ausdrücken aus und gibt einen Wert vom Typ Variant oder einen Ausdruck zurück, der dem ersten Ausdruck in der Liste zugeordnet ist, der True ergibt.
Trim	Gibt einen Wert vom Typ Variant (String) zurück, der eine Kopie einer bestimmten Zeichenfolge enthält, die keine führenden Leerzeichen (LTrim), keine nachgestellten Leerzeichen (RTrim) sowie keine Kombination aus führenden und nachgestellten Leerzeichen (Trim) enthält.
UCase	Wandelt Kleinbuchstaben in Großbuchstaben um.

Tabelle 3.13: Textfunktionen (Forts.)

Asc-Funktion

Die Funktion Asc gibt einen Wert vom Typ Integer zurück, der den Zeichencode entsprechend dem ersten Buchstaben in einer Zeichenfolge darstellt.

► Syntax

Asc(Zeichenfolge)

Das erforderliche Zeichenfolge-Argument ist ein beliebiger gültiger Zeichenfolgenausdruck. Wenn Zeichenfolge keine Zeichen enthält, tritt ein Laufzeitfehler auf.

► Beispiel

Im folgenden Beispiel aus Listing 3.29 werden über eine benutzerdefinierte Funktion alle Buchstaben aus Zellen entfernt, sodass nur noch Zahlenwerte in den Zellen zurückbleiben.

Listing 3.29: Buchstaben aus Zellen entfernen

```

Function BuchstRaus(Zelle) As Integer
Dim i As Integer

Application.Volatile
For i = 1 To Len(Zelle)
    Select Case Asc(Mid(Zelle, i, 1))
        Case 0 To 64, 123 To 197
            BuchstRaus = BuchstRaus & Mid(Zelle, i, 1)
    End Select
Next i
End Function

```

Prüfen Sie mithilfe der Funktion `Asc` das jeweils aktuelle Zeichen der Zelle, indem Sie dieses in einen Integer-Wert umwandeln. Mit der Funktion `Mid` extrahieren Sie jeweils das nächste Zeichen aus der Zelle. Dabei entsprechen die Werte 65-90 Kleinbuchstaben, die Werte 97-122 den Großbuchstaben und die restlichen Werte den Sonderzeichen.

► Verwandte Funktionen

`Chr` und Umwandlungsfunktionen

Choose-Funktion

Die Funktion `Choose` wählt einen Wert aus einer Liste von Argumenten aus und gibt ihn zurück.

► Syntax

`Choose(Index, Auswahl-1[, Auswahl-2, ... [, Auswahl-n]])`

Die Syntax für die `Choose`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Index	Erforderlich. Numerischer Ausdruck oder Feld, der oder das einen Wert von 1 bis zur Anzahl der möglichen Auswahlwerte ergibt.
Auswahl	Erforderlich. Ein <code>Variant</code> -Ausdruck, der einen der möglichen Auswahlwerte enthält.

Tabelle 3.14: Argumente der Funktion `Choose`

► Beispiel

Im folgenden Beispiel aus Listing 3.30 wird über eine Funktion die Zahlungsart ermittelt.

Listing 3.30: Eine übergebene Zahl wird in einen Text gewandelt.

```
Function Auswahl(i As Integer)
    Auswahl = Choose(i, "Bar", "EC", "Rechnung")
End Function

Sub Choose_Beispiel()
    Debug.Print Auswahl(1)
    Debug.Print Auswahl(2)
    Debug.Print Auswahl(3)
End Sub
```

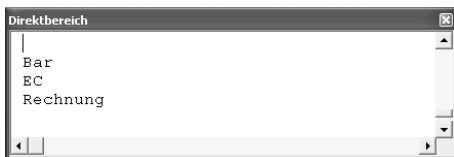


Abbildung 3.17: Aus 1 wird Bar, aus 2 wird EC, usw.

► Verwandte Funktionen

Select Case, Switch

Chr-Funktion

Die Funktion **Chr** gibt einen Wert vom Typ **String** zurück, der das Zeichen enthält, das dem angegebenen Zeichencode zugeordnet ist.

► Syntax

Chr(Zeichencode)

Das erforderliche Argument **Zeichencode** ist ein Wert vom Typ **Long**, der ein Zeichen festlegt.

► Beispiel

Im folgenden Beispiel aus Listing 3.31 werden mithilfe der Funktion `Chr` aus Zahlen Buchstaben gewonnen und in eine Tabelle geschrieben.

Listing 3.31: Aus Zahlenwerten Buchstaben erstellen

```
Sub Chr_Beispiel()
Dim i As Integer
Dim e As Integer

e = 1
For i = 65 To 90
    With Sheets("Tabelle3")
        .Cells(1, e).Value = Chr(i)
        e = e + 1
    End With
Next i
End Sub
```

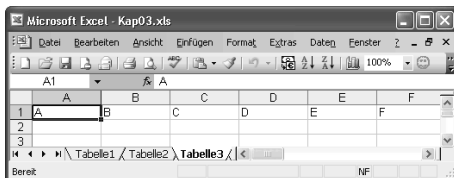


Abbildung 3.18: Eine Buchstabenreihe erstellen

Im nächsten Beispiel aus Listing 3.32 wird eine mehrzeilige Meldung auf dem Bildschirm angezeigt.

Listing 3.32: Einen Zeilenvorschub über Chr(13) generieren

```
Sub Chr_Beispiel2()
MsgBox "Hallo " & Application.UserName & Chr(13) & _
"Heute ist " & Date & Chr(13) & _
"Aktuelle Uhrzeit: " & time
End Sub
```



Abbildung 3.19: Mehrzeilige MsgBox anzeigen

► Verwandte Funktionen

Acc, Str, Typ-Umwandlungsfunktionen

InStr-Funktion

Über die Funktion `InStr` können Sie einen Wert vom Typ `Variant (Long)` zurückgeben, der die Position des ersten Auftretens einer Zeichenfolge innerhalb einer anderen Zeichenfolge angibt.

► Syntax

`InStr([Start,]Zeichenfolge1, Zeichenfolge2
[, Vergleich])`

Die Syntax der `InStr`-Funktion verwendet die folgenden Argumente:

Teil	Beschreibung
Start	Optional. Numerischer Ausdruck, der die Startposition für die Suche festlegt. Wird <code>Start</code> nicht angegeben, so beginnt die Suche mit dem ersten Zeichen in der Zeichenfolge. Wenn <code>Start</code> den Wert <code>Null</code> enthält, tritt ein Fehler auf. Bei Angabe von <code>Vergleich</code> muss auch das Argument <code>Start</code> angegeben werden.
Zeichenfolge1	Erforderlich. Durchsuchter Zeichenfolgenausdruck.
Zeichenfolge2	Erforderlich. Gesuchter Zeichenfolgenausdruck.
Vergleich	Optional. Legt die Art des Zeichenfolgenvergleichs fest. Der Wert <code>Null</code> für <code>Vergleich</code> führt zu einem Fehler.

Tabelle 3.15: Argumente der Funktion `InStr`

Konstante	Wert	Beschreibung
vbUseCompareOption	-1	Führt einen Vergleich mithilfe der Option <code>Compare</code> -Anweisung durch.
vbBinaryCompare	0	Führt einen binären Vergleich durch.
vbTextCompare	1	Führt einen textbasierten Vergleich durch.
vbDatabaseCompare	2	Nur Microsoft Access. Führt einen Vergleich durch, der auf Informationen in Ihrer Datenbank basiert.

Tabelle 3.16: Vergleichstypen der Funktion `InStr`

Als Rückgabewerte der Funktion `InStr` gelten folgende Argumente.

Fall	Rückgabewerte von <code>InStr</code>
Zeichenfolge1 hat die Länge Null	0
Zeichenfolge1 ist Null	Null
Zeichenfolge2 hat die Länge Null	start
Zeichenfolge2 ist Null	Null
Zeichenfolge2 ist nicht vorhanden	0
Zeichenfolge2 ist in Zeichenfolge1 enthalten	Position, an der Übereinstimmung beginnt
Start > Zeichenfolge2	0

Tabelle 3.17: Rückgabewerte der Funktion `InStr`

► Beispiele

Im folgenden Beispiel aus Listing 3.33 wird geprüft, ob es sich um eine gültige E-Mail-Adresse handelt.

Listing 3.33: Überprüfen auf gültige E-Mail-Adresse

```
Sub InStr_Beispiel()
Dim s As String

s = Held-office@t-online.de
```

```

If InStr(s, "@") > 1 Then
    MsgBox "E-Mail gültig!"
Else
    MsgBox "E-Mail ungültig!"
End If
End Sub

```

Wird die Zeichenfolge @ in der Variablen s gefunden, dann meldet die Funktion einen Rückgabewert größer 1 zurück. Dabei wird die genaue Position des gesuchten Zeichens zurückgegeben. Fängt die E-Mail-Adresse mit der Zeichenfolge @ an, so handelt es sich um eine nicht gültige E-Mail-Adresse.

Im folgenden Beispiel aus Listing 3.34 werden in einer Tabelle alle Zellen gelöscht, die einen Bindestrich enthalten.

Listing 3.34: Zellen mit bestimmtem Inhalt löschen

```

Sub InStr_Beispiel2()
    Dim zelle As Range
    Dim iPos As Integer

    For Each zelle In ActiveSheet.UsedRange
        iPos = InStr(1, zelle, "-", 1)
        If iPos > 0 Then zelle.ClearContents
    Next zelle
End Sub

```

► Verwandte Funktionen

InStrRev, StrComp

InStrRev-Funktion

Die Funktion InStrRev gibt die Position eines Vorkommnisses einer Zeichenfolge in einer anderen Zeichenfolge vom Ende der Zeichenfolge gesehen an.

► Syntax

InstrRev(stringcheck, stringmatch [, start[, compare]])

Die Syntax der InstrRev-Funktion besteht aus folgenden benannten Argumenten:

Teil	Beschreibung
Stringcheck	Erforderlich. Der zu durchsuchende Zeichenfolgenausdruck.
Stringmatch	Erforderlich. Zeichenfolgenausdruck, nach dem gesucht wird.
Start	Optional. Numerischer Ausdruck, der die Anfangsposition für jede Suche festlegt. Wird er ausgelassen, wird -1 verwendet, d.h. die Suche beginnt an der letzten Zeichenposition.
Compare	Optional. Numerischer Wert, der die Art des Vergleichs angibt, der beim Beurteilen von untergeordneten Zeichenfolgen verwendet werden soll. Wird er ausgelassen, wird ein binärer Vergleich ausgeführt.

Tabelle 3.18: Argumente der Funktion InStrRev

► Beispiele

Im folgenden Beispiel aus Listing 3.35 werden der komplette Pfad, der Pfad sowie der Dateiname der aktiven Arbeitsmappe ermittelt.

Listing 3.35: Pfad- und Dateinamen extrahieren (ab Excel 2000)

```
Sub InStrRev_Beispiel()
    Dim iPos As Integer
    Dim sPfad As String
    Dim sName As String

    iPos = InStrRev(ActiveWorkbook.FullName, "\")
    sPfad = Left(ActiveWorkbook.FullName, iPos)
    sName = Mid(ActiveWorkbook.FullName, iPos + 1)
    MsgBox "Komplett: " & ActiveWorkbook.FullName & _
        vbCrLf & "Pfad: " & sPfad & vbCrLf & "Datei: " & sName
End Sub
```

Aus der Eigenschaft `FullName`, in welcher der Name der Datei sowie der komplette Speicherpfad verzeichnet sind, werden die einzelnen Informationen herausgezogen.



Abbildung 3.20: Die InstrRev-Funktion hilft beim Separieren der einzelnen Elemente

Anwender der Version Excel 97 haben beim Makro aus Listing 3.35 Pech. Leider gibt es diese Funktion erst ab Excel 2000. Für Excel 97-Anwender muss daher folgende Syntax aus Listing 3.36 gelten.

Listing 3.36: Pfad- und Dateinamen extrahieren (für Excel 97)

```
Sub Excel97()
Dim iPos As Integer
Dim AltPos As Integer
Dim sPfad As String
Dim sName As String

iPos = 1
Do Until iPos = 0
    AltPos = iPos
    iPos = Instr(iPos + 1, _
        ActiveWorkbook.FullName, "\")
Loop
sPfad = Left(ActiveWorkbook.FullName, AltPos)
sName = Mid(ActiveWorkbook.FullName, AltPos + 1)
MsgBox "Komplett: " & ActiveWorkbook.FullName & _
vbLf & "Pfad: " & sPfad & vbLf & "Datei: " & sName
End Sub
```

► Verwandte Funktionen

InStr, StrComp

Join-Funktion

Die Funktion `Join` gibt eine Zeichenfolge zurück, die sich aus der Kombination einer Reihe von untergeordneten Zeichenfolgen, die in einem Datenfeld enthalten sind, ergibt.

► Syntax

`Join(sourcearray [, delimiter])`

Die Syntax der `Join`-Funktion besteht aus folgenden benannten Argumenten:

Teil	Beschreibung
Sourcearray	Erforderlich. Eindimensionales Datenfeld, das die zu kombinierenden, untergeordneten Zeichenfolgen enthält.
Delimiter	Optional. Zeichen einer Zeichenfolge, mit dem die untergeordneten Zeichenfolgen in der zurückgegebenen Zeichenfolge getrennt werden. Wird es ausgelassen, wird das Leerstellenzeichen (» «) verwendet. Wenn <code>delimiter</code> eine Zeichenfolge der Länge Null (»«) ist, werden alle Elemente in der Liste ohne Trennzeichen verkettet.

Tabelle 3.19: Argumente der Funktion `Join`

► Beispiel

Beim folgenden Beispiel aus Listing 3.37 wird eine Adresse über die Funktion `Split` zerlegt, aufbereitet und anschließend mithilfe der Funktion `Join` wieder zusammengebastelt.

Listing 3.37: Text auseinander nehmen und wieder zusammensetzen

```
Sub JoinSplitt_Beispiel()
    Dim sText As String
    Dim Varray As Variant
    Dim i As Integer
    Dim iGesamt As Integer

    sText = "Bernd;Held;Teststr. 15;70839;Gerlingen"
    Debug.Print "Vor Konvertierung: " & sText
    Varray = Split(sText, ";")
```

```

iGesamt = UBound(Varray)
For i = 0 To iGesamt
    Varray(i) = UCase(Varray(i))
Next i
Varray = Join(Varray, ";")
Debug.Print "Nach Konvertierung: " & Varray
End Sub

```



Abbildung 3.21: Texte splitten und wieder zusammensetzen

► Verwandte Funktionen

Split

LCase-Funktion

Die Funktion `LCase` wandelt Großbuchstaben in Kleinbuchstaben um.

► Syntax

`LCase(Zeichenfolge)`

Das erforderliche Argument `Zeichenfolge` ist ein beliebiger gültiger Zeichenfolgenausdruck.

► Beispiele

Im folgenden Beispiel aus Listing 3.38 wird der benutzte Bereich einer Tabelle überarbeitet. Dabei werden alle Zellen, die Texte enthalten, in Kleinbuchstaben konvertiert.

Listing 3.38: Alle Texte der benutzten Zellen werden in Kleinbuchstaben umgewandelt.

```
Sub LCase_Bispiel()  
Dim Zelle As Range  
  
For Each Zelle In ActiveSheet.UsedRange  
    Zelle.Value = LCase(Zelle.Value)  
Next Zelle  
End Sub
```

Im nächsten Beispiel werden alle Eingaben in der TABELLE4 automatisch direkt nach der Eingabe in Kleinbuchstaben umgewandelt.

Um diese Aufgabe umzusetzen, verfahren Sie wie folgt:

1. Klicken Sie mit der rechten Maustaste auf den Tabellenreiter der TABELLE4 und wählen Sie aus dem Kontextmenü den Befehl CODE ANZEIGEN.
2. Stellen Sie im Code-Fenster oben aus dem ersten Dropdown-Feld den Eintrag WORKSHEET ein.
3. Wählen Sie aus dem zweiten Dropdown-Feld den Befehl CHANGE.
4. Ergänzen Sie den eingestellten Ereignisrahmen wie folgt:

Listing 3.39: Jede Eingabe in der Tabelle4 wird automatisch in Kleinbuchstaben gewandelt.

```
Private Sub Worksheet_Change(ByVal Target As Range)  
    Target.Value = LCase(Target.Value)  
End Sub
```

► **Verwandte Funktionen**

UCase, StrConv

Left-Funktion

Die Funktion **Left** gibt einen Wert vom Typ **Variant (String)** zurück, der eine bestimmte Anzahl von Zeichen ab dem ersten (linken) Zeichen einer Zeichenfolge enthält.

► Syntax

Left(string, length)

Die **Left**-Funktion verwendet die folgenden benannten Argumente:

Teil	Beschreibung
String	Erforderlich. Zeichenfolgenausdruck, aus dem die ersten (linken) Zeichen zurückgegeben werden. Wenn string den Wert Null enthält, wird Null zurückgegeben.
Length	Erforderlich; Wert vom Typ Variant (Long) . Numerischer Ausdruck, der die Anzahl der zurückzugebenden Zeichen angibt. Der Wert 0 führt zur Rückgabe einer Null-Zeichenfolge (»«). Ist length größer oder gleich der Zeichenanzahl in string , so wird die gesamte Zeichenfolge zurückgegeben.

Tabelle 3.20: Argumente der Funktion Left

► Beispiel

Im folgenden Beispiel aus Listing 3.40 wird eine Versionsprüfung in Excel durchgeführt. Manchmal machen gerade ältere Excel-Versionen bei der Programmierung Kummer.

Listing 3.40: Excel-Versionscheck durchführen

```
Sub Left_Beispiel()
Select Case Left(Application.Version, 1)
Case "5"
MsgBox "Sie haben Excel 5 im Einsatz"
Case "7"
MsgBox "Sie haben Excel 95 im Einsatz"
Case "8"
MsgBox "Sie haben Excel 97 im Einsatz"
Case Else
'Keine Probleme
End Select
End Sub
```

► Verwandte Funktionen

Right, Mid, Len

Len-Funktion

Die Funktion `Len` gibt einen Wert vom Typ `Long` zurück, der die Anzahl der Zeichen in einer Zeichenfolge oder die zum Speichern einer Variablen erforderlichen Bytes enthält.

► Syntax

`Len(Zeichenfolge | Variablenname)`

Die Syntax der `Len`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Zeichenfolge	Beliebiger gültiger Zeichenfolgenausdruck. Wenn Zeichenfolge den Wert Null enthält, wird Null zurückgegeben.
Variablenname	Beliebiger gültiger Name für eine Variable. Wenn Variablenname den Wert Null enthält, wird Null zurückgegeben. Wenn Variablenname ein Wert vom Typ <code>Variant</code> ist, wird er von <code>Len</code> genauso behandelt wie ein Wert vom Typ <code>String</code> , und es wird immer die Anzahl der darin enthaltenen Zeichen zurückgegeben.

Tabelle 3.21: Argumente der Funktion `Len`

► Beispiele

Im folgenden Makro aus Listing 3.41 wird mithilfe der Funktion `Len` geprüft, ob in einem Eingabedialog überhaupt eine Eingabe vorgenommen wurde.

Listing 3.41: Welche Schaltfläche wurde geklickt?

```
Sub Len_Beiispiel()
Dim s As String

s = InputBox("Bitte Ihren Namen eingeben", _
    "Eingabe")

If Len(s) <> 0 Then
    'Alles in Ordnung
```

```
Else
```

```
    MsgBox "Sie haben nichts eingegeben!"
```

```
End If
```

```
End Sub
```

Wenn im Dialog aus Abbildung 3.22 nichts eingegeben wird, dann meldet die Funktion Len den Wert 0. In diesem Fall geben Sie eine Warnmeldung am Bildschirm aus.



Abbildung 3.22: Überprüfung, ob die Eingabe erfolgt ist

Im nächsten Beispiel aus Listing 3.42 wird die Spalte A der TABELLE5 überprüft. Dabei dürfen Eingaben in dieser Spalte nur genau 8 Zeichen enthalten.

Listing 3.42: Ungültige Längen von Eingaben kennzeichnen

```
Sub Len_Beispiel2()
```

```
Dim i As Integer
```

```
With Sheets("Tabelle5")
```

```
    For i = 1 To .UsedRange.Rows.Count
```

```
        If Len(.Cells(i, 1).Value) <> 8 Then
```

```
            .Cells(i, 2).Value = "Falsche Länge!"
```

```
        End If
```

```
    Next i
```

```
End With
```

```
End Sub
```

► Verwandte Funktionen

InStr, Left, Mid, Right

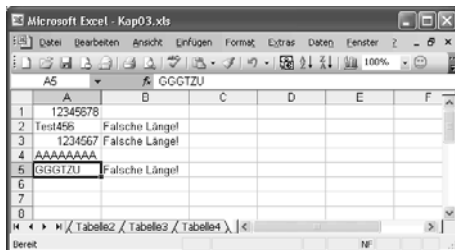


Abbildung 3.23: Alle Zellen mit einer Zeichenlänge ungleich 8 werden gekennzeichnet

Mid-Funktion

Die Funktion **Mid** gibt einen Wert vom Typ **Variant (String)** zurück, der eine bestimmte Anzahl von Zeichen aus einer Zeichenfolge enthält.

Syntax

Mid(string, start[, length])

Die Syntax der **Mid**-Funktion verwendet die folgenden benannten Argumente:

Teil	Beschreibung
String	Erforderlich. Zeichenfolgenausdruck, aus dem Zeichen zurückgegeben werden. Wenn string den Wert Null enthält, wird Null zurückgegeben.
Start	Erforderlich; Wert vom Typ Long . Position in string , an der die zurückzugebende Zeichenfolge beginnt. Ist start größer als die Anzahl der Zeichen in string , so gibt Mid eine leere Zeichenfolge (») zurück.
Length	Optional; Wert vom Typ Variant (Long) . Anzahl der zurückzugebenden Zeichen. Wird length nicht angegeben oder befinden sich weniger Zeichen im Text (das Zeichen an der Stelle start eingeschlossen), als durch length angegeben, so werden alle Zeichen ab start bis zum Ende der Zeichenfolge zurückgegeben.

Tabelle 3.22: Argumente der Funktion **Mid**

► **Beispiele**

Im folgenden Beispiel aus Listing 3.43 wird in einer Zelle nach jedem vierten Zeichen ein Leerzeichen eingefügt.

Listing 3.43: Text aufteilen, Leerzeichen einfügen und wieder zusammensetzen

```
Sub Mid_Beispiel()  
Dim i As Integer  
Dim s As String  
  
For i = 1 To Len(ActiveCell.Value)  
    If i Mod 4 = 0 Then  
        s = s & " " & Mid(ActiveCell.Value, i - 3, 4)  
    End If  
Next i  
ActiveCell.Value = s  
End Sub
```

Beim nächsten Beispiel aus Listing 3.44 wird der Text aus einer Zelle Buchstabe für Buchstabe zerlegt und einzeln in die Nebenspalten eingefügt.

Listing 3.44: Text zerlegen und einzelne Buchstaben ausgeben

```
Sub Mid_Beispiel2()  
Dim i As Integer  
  
With Sheets("Tabelle6")  
    For i = 1 To Len(ActiveCell.Value)  
        ActiveCell.Offset(0, i).Value = _  
            Mid(ActiveCell.Value, i, 1)  
    Next i  
End With  
End Sub
```

► **Verwandte Funktionen**

Left, Right, Len, Instr

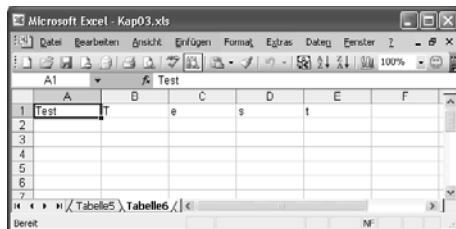


Abbildung 3.24: Der Text aus Zelle A1 wurde aufgeteilt

Replace-Funktion

Die Funktion `Replace` gibt eine Zeichenfolge zurück, in der eine festgelegte, untergeordnete Zeichenfolge mit einer festgelegten Häufigkeit durch eine andere untergeordnete Zeichenfolge ersetzt wurde.

Syntax

`Replace(expression, find, replace[, start[, count[, compare]])`

Die Syntax der `Replace`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Expression	Erforderlich. Zeichenfolgenausdruck, der die zu ersetzende, untergeordnete Zeichenfolge enthält.
Find	Erforderlich. Die untergeordnete Zeichenfolge, nach der gesucht wird.
Replace	Erforderlich. Die untergeordnete Ersatzzeichenfolge.
Start	Optional. Position in <code>expression</code> , an der die Suche nach der untergeordneten Zeichenfolge beginnt. Wird diese Angabe ausgelassen, wird bei 1 begonnen.

Tabelle 3.23: Argumente der Funktion `Replace`

Teil	Beschreibung
Count	Optional. Anzahl der durchzuführenden Ersetzungen der untergeordneten Zeichenfolge. Wird diese Angabe ausgelassen, ist die Standardeinstellung -1, d.h. alle möglichen Zeichenfolgen werden ersetzt.
Compare	Optional. Numerischer Wert, der die Art des Vergleichs angibt, der beim Beurteilen von untergeordneten Zeichenketten verwendet werden soll.

Tabelle 3.23: Argumente der Funktion Replace (Forts.)

Das Argument `compare` kann folgende Werte haben:

Konstante	Wert	Beschreibung
<code>vbUseCompareOption</code>	-1	Führt einen Vergleich unter Verwendung der Option <code>Compare</code> -Anweisung durch.
<code>vbBinaryCompare</code>	0	Führt einen binären Vergleich durch.
<code>vbTextCompare</code>	1	Führt einen Textvergleich durch.
<code>vbDatabaseCompare</code>	2	Nur Microsoft Access. Führt einen Vergleich anhand der Informationen in Ihrer Datenbank durch.

Tabelle 3.24: Die möglichen Vergleichskonstanten

► Beispiele

Im folgenden Beispiel aus Listing 3.45 werden alle Zellen, die sich momentan in der Markierung befinden, untersucht. Dabei werden Zellen, die ein bestimmtes Zeichen enthalten, mit einem Ersatzzeichen versorgt.

Listing 3.45: Zeichen austauschen mit der Funktion Replace

```
Sub Replace_Beispiel()
    Dim Zelle As Range

    For Each Zelle In Selection
        Zelle.Value = Replace(Zelle.Value, "/", "-")
    Next Zelle
End Sub
```

Sollen zusätzlich noch alle Leerzeichen aus den Zellen verschwinden, dann starten Sie das Makro aus Listing 3.46.

Listing 3.46: Sonderzeichen und Leerzeichen austauschen

```
Sub Replace_Beispiel2()
Dim Zelle As Range

    For Each Zelle In Selection
        Zelle.Value = Replace(Zelle.Value, "/", "-")
        Zelle.Value = Replace(Zelle.Value, " ", "")
    Next Zelle
End Sub
```

► Verwandte Funktionen

Filter

Right-Funktion

Die Funktion **Right** gibt einen Wert vom Typ **Variant (String)** zurück, der eine bestimmte Anzahl von Zeichen von der rechten Seite (dem Ende) einer Zeichenfolge enthält.

► Syntax

Right(string, length)

Die Syntax der **Right**-Funktion verwendet die folgenden benannten Argumente:

Teil	Beschreibung
String	Erforderlich. Zeichenfolgenausdruck, von dem die letzten (rechtsstehenden) Zeichen zurückgegeben werden. Wenn string den Wert Null enthält, wird Null zurückgegeben.
Length	Erforderlich; Wert vom Typ Variant (Long) . Numerischer Ausdruck, der die Anzahl der zurückzugebenden Zeichen angibt. Der Wert 0 führt zur Rückgabe einer Null-Zeichenfolge (»«). Ist Length größer oder gleich der Zeichenanzahl in String , so wird die gesamte Zeichenfolge zurückgegeben.

Tabelle 3.25: Argumente der Funktion **Right**

► Beispiele

Im folgenden Beispiel aus Listing 3.47 soll ein Dateiname in eine Inputbox eingegeben werden. Danach wird geprüft, ob es sich um einen Excel-Dateinamen handelt. Diese Information können Sie an der Dateierweiterung .XLS sehen.

Listing 3.47: Prüfung, ob eine Excel-Datei eingegeben wurde

```
Sub Right_Beispiel()
Dim s As String

s = InputBox("Bitte Dateinamen eingeben", _
    "Dateinamen:")

If Len(s) <> 0 Then
    If UCase(Right(fname, 3)) <> "XLS" Then
        MsgBox "Es handelt sich um eine Excel-Datei!"
    Else
        MsgBox "Keine Excel-Datei!"
    End If
End If
End Sub
```

Schneiden Sie am Dateinamen die letzten drei Zeichen über die Funktion `Right` ab und werten diese Information dann aus.

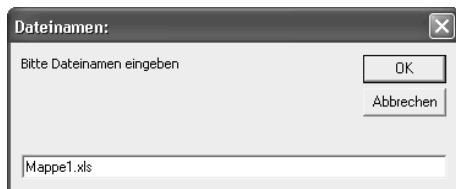


Abbildung 3.25: Bei XLS-Dateien handelt es sich um Excel-Dateien

Beim nächsten Beispiel aus Listing 3.48 werden die einzelnen Buchstaben eines Textes aus einer Zelle ständig durcheinander geworfen und immer wieder neu gemischt.

Listing 3.48: Buchstaben durcheinander wirbeln

```

Sub Right_Beispiel2()
Dim i As Integer

For i = 1 To 10
    ActiveCell.Value = Right(ActiveCell.Value, _
        Len(ActiveCell.Value) - 1) + _
        Left(ActiveCell.Value, 1)
    Application.Wait Now + TimeSerial(0, 0, 1)
Next i
End Sub

```

Nach jedem Zeichentausch wird genau eine Sekunde pausiert.

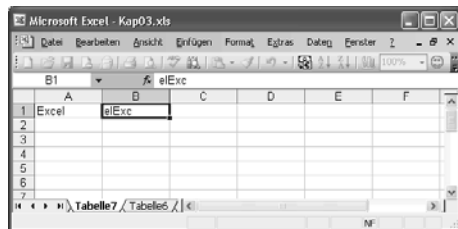


Abbildung 3.26: Zeichenfolgen auseinander reißen und wieder neu zusammensetzen

► Verwandte Funktionen

Left, Mid, Len

Space-Funktion

Die Funktion **Space** gibt eine Zeichenfolge vom Typ Variant (String) zurück, die aus einer angegebenen Anzahl von Leerzeichen besteht.

► Syntax

Space(Zahl)

Das erforderliche Argument **Zahl** entspricht der Anzahl der gewünschten Leerzeichen in der Zeichenfolge.

► **Beispiel**

Im folgenden Beispiel aus Listing 3.49 werden alle markierten Zellen mit Leerzeichen aufgefüllt, sofern die Gesamtlänge der Zelle weniger als 10 Zeichen beträgt.

Listing 3.49: Zellen werden mit Leerzeichen aufgefüllt

```
Sub Space_Beispiel()
Dim Zelle As Range
Dim s As String

For Each Zelle In Selection
    s = Zelle.Value
    If Len(s) < 10 Then
        Zelle.Value = s + Space(10 - Len(s))
    End If
Next Zelle
End Sub
```

► **Verwandte Funktionen**

Spc, String

Spc-Funktion

Die Funktion Spc fügt in eine Textdatei eine bestimmte Anzahl von Leerzeichen ein.

► **Syntax**

Spc(n)

Das erforderliche Argument n gibt die Anzahl der Leerzeichen an, die vor dem Anzeigen oder Ausgeben des nächsten Ausdrucks in eine Liste eingefügt werden sollen.

► **Beispiel**

Im folgenden Beispiel wird die TABELLE8 aus Abbildung 3.27 in einer Textdatei mit dem Namen AUSGABE.TXT geschrieben. Dabei soll jede zweite Zeile in der Textdatei eine Leerzeile sein.

	A	B	C	D	E
1	Nr	Bezeichnung	Preis		
2		4711 Hammer	19.95		
3		4712 Säge	13.99		
4					
5					
6					
7					

Abbildung 3.27: Diese Daten sollen in eine Textdatei geschrieben werden

Um diese Aufgabe zu lösen, starten Sie das Makro aus Listing 3.50.

Listing 3.50: Tabelle als Textdatei speichern

```
Sub TabelleAlsTextdatei()
    Dim Daten As Range
    Dim Zeile As Range
    Dim Zelle As Range
    Dim s As String

    Set Daten = Sheets("Tabelle8").UsedRange
    Open _
    "c:\Eigene Dateien\Ausgabe.txt" For Output As #1

    For Each Zeile In Daten.Rows
        For Each Zelle In Zeile.Cells
            s = s & CStr(Zelle.Text) & ";"
        Next Zelle
        s = Left(s, Len(s) - 1)
        Print #1, s
        Print #1, Spc(10)
        s = ""
    Next
    Close #1
End Sub
```

Über die Anweisung `Print #1, Spc(10)` wird eine Leerzeile, bestehend aus 10 Leerzeichen, in die Textdatei eingefügt.



Abbildung 3.28: Die Daten wurden in einer Textdatei gespeichert

► Verwandte Funktionen

Space

Split-Funktion

Die Funktion `Split` gibt ein nullbasiertes, eindimensionales Datenfeld zurück, das eine festgelegte Anzahl an untergeordneten Zeichenfolgen enthält.

► Syntax

`Split(expression[, delimiter[, limit[, compare]])`

Teil	Beschreibung
Expression	Erforderlich. Zeichenfolgenausdruck, der untergeordnete Zeichenfolgen und Trennzeichen enthält. Wenn <code>expression</code> eine Zeichenfolge der Länge Null (»«) ist, gibt <code>Split</code> ein leeres Datenfeld zurück, d.h. ein Datenfeld ohne Elemente und ohne Daten.
Delimiter	Optional. Zeichen einer Zeichenfolge, mit der die Grenzen von untergeordneten Zeichenfolgen identifiziert werden. Wird es ausgelassen, wird das Leerstellenzeichen (» «) als Trennzeichen verwendet. Wenn <code>delimiter</code> eine Zeichenfolge der Länge Null ist, wird ein aus einem Element bestehendes Datenfeld, das die gesamte Zeichenfolge von <code>expression</code> enthält, zurückgegeben.
Limit	Optional. Anzahl der zurückzugebenden untergeordneten Zeichenfolgen; -1 gibt an, dass alle untergeordneten Zeichenfolgen zurückgegeben werden.
Compare	Optional. Numerischer Wert, der die Art des Vergleichs angibt, der beim Beurteilen von untergeordneten Zeichenketten verwendet werden soll.

Tabelle 3.26: Argumente der Funktion `Split`

► Beispiel

Siehe Funktion `Join`

► Verwandte Funktionen

`Join`

Str-Funktion

Die Funktion `Str` gibt einen Wert vom Typ `Variant (String)` zurück, der eine Zahl darstellt.

► Syntax

`Str(Zahl)`

Das erforderliche Argument `Zahl` ist ein Wert vom Typ `Long`, der einen beliebigen numerischen Ausdruck enthält. Beim Konvertieren von Zahlen in Texte wird ein führendes Leerzeichen für das Vorzeichen der Zahl reserviert. Ist die Zahl positiv, so enthält der zurückgegebene Text ein führendes Leerzeichen.

► Beispiel

Im folgenden Beispiel aus Listing 3.51 wird eine Zahl in einen Textwert umgewandelt.

Listing 3.51: Zahl in Text umwandeln

```
Sub Str_Beispiel()  
Dim zahl As Long  
zahl = 8989
```

```
Debug.Print Str(zahl)  
End Sub
```

► Verwandte Funktionen

`Format`, `Val`, Umwandlungsfunktionen

StrComp-Funktion

Mit der Funktion `StrComp` können Sie Zeichenfolgen miteinander vergleichen.

► Syntax

`StrComp(string1, string2[, compare])`

Teil	Beschreibung
<code>string1</code>	Erforderlich. Ein beliebiger gültiger Zeichenfolgenausdruck.
<code>string2</code>	Erforderlich. Ein beliebiger gültiger Zeichenfolgenausdruck.
<code>Compare</code>	Optional. Legt die Art des Zeichenfolgenvergleichs fest. Der Wert Null für <code>compare</code> führt zu einem Fehler. Die möglichen Vergleichstypen können Sie Tabelle 3.28 entnehmen.

Tabelle 3.27: Argumente der Funktion `StrComp`

Konstante	Wert	Beschreibung
<code>vbUseCompareOption</code>	-1	Führt einen Vergleich mithilfe der Option <code>Compare</code> -Anweisung durch.
<code>vbBinaryCompare</code>	0	Führt einen binären Vergleich durch.
<code>vbTextCompare</code>	1	Führt einen textbasierten Vergleich durch.
<code>vbDatabaseCompare</code>	2	Nur Microsoft Access. Führt einen Vergleich durch, der auf Informationen in Ihrer Datenbank basiert.

Tabelle 3.28: Vergleichstypen der Funktion `StrComp`

Die Funktion `StrComp` liefert folgende Rückgabewerte.

Fall	Rückgabewert
<code>string1</code> liegt im Alphabet vor <code>string2</code>	-1
<code>string1</code> entspricht <code>string2</code>	0

Tabelle 3.29: Rückgabewerte der Funktion `StrComp`

Fall	Rückgabewert
string1 liegt im Alphabet hinter string2	1
string1 oder string2 ist Null	Null

Tabelle 3.29: Rückgabewerte der Funktion StrComp (Forts.)

► Beispiel

Im folgenden Beispiel aus Listing 3.52 werden zwei Zeichenfolgen miteinander verglichen und überprüft, welche der beiden Zeichenfolgen im Alphabet weiter vorn angeordnet ist.

Listing 3.52: Textvergleich über die Funktion StrComp durchführen

```
Sub StrComp_Beispiel()
Dim Text1 As String
Dim Text2 As String

Text1 = "RTS"
Text2 = "STS"

If StrComp(Text1, Text2, 1) = -1 Then
    MsgBox "Text1 liegt vor Text2!"
Else
    MsgBox "Text2 liegt vor Text1"
End If
End Sub
```

► Verwandte Funktionen

InStr

StrConv-Funktion

Die Funktion `StrConv` gibt einen Wert vom Typ `Variant (String)` zurück, der wie angegeben umgewandelt wurde.

► Syntax

`StrConv(string, conversion, LCID)`

Die Syntax der `StrConv`-Funktion besteht aus folgenden benannten Argumenten:

Teil	Beschreibung
String	Erforderlich. Zeichenfolgenausdruck, der umgewandelt werden soll.
Conversion	Erforderlich. Wert vom Typ <code>Integer</code> . Die Summe der Werte, die den Typ der auszuführenden Umwandlung angibt.
LCID	Optional. Die Gebietsschema-ID, wenn diese sich von der des Systems unterscheidet. (Die Gebietsschema-ID des Systems ist die Voreinstellung.)

Tabelle 3.30: Argumente der Funktion `StrConv`

Konstante	Wert	Beschreibung
<code>vbUpperCase</code>	1	Wandelt die Zeichenfolge in Großbuchstaben um.
<code>VbLowerCase</code>	2	Wandelt die Zeichenfolge in Kleinbuchstaben um.
<code>VbProperCase</code>	3	Wandelt den ersten Buchstaben jedes Wortes innerhalb der Zeichenfolge in einen Großbuchstaben um.
<code>vbWide*</code>	4	Wandelt schmale (Einzel-Byte) Zeichen innerhalb der Zeichenfolge in breite (Doppel-Byte) Zeichen um.
<code>vbNarrow</code>	8	Wandelt breite (Doppel-Byte) Zeichen innerhalb der Zeichenfolge in schmale (Einzel-Byte) Zeichen um.

Tabelle 3.31: Einstellungen für das Argument `conversion`

Konstante	Wert	Beschreibung
vbKatakana	16	Wandelt Hiragana-Zeichen innerhalb der Zeichenfolge in Katakana-Zeichen um.
vbHiragana	32	Wandelt Katakana-Zeichen innerhalb der Zeichenfolge in Hiragana-Zeichen um.
vbUnicode	64	Wandelt die Zeichenfolge in Unicode um und verwendet dabei die Voreinstellungen aus der Zeichenumsetzungstabelle des Systems. (Nicht verfügbar auf dem Macintosh.)
vbFromUnicode	128	Wandelt die Zeichenfolge von Unicode in die Voreinstellungen aus der Zeichenumsetzungstabelle des Systems um. (Nicht verfügbar auf dem Macintosh.)

Tabelle 3.31: Einstellungen für das Argument conversion (Forts.)

► Beispiel

Im folgenden Beispiel aus Listing 3.53 werden alle markierten Zellen abgearbeitet und deren Inhalte in Kleinbuchstaben gewandelt.

Listing 3.53: Texte konvertieren über die Funktion StrConv

```
Sub StrConv_Beispiel()
    Dim zelle As Range

    For Each zelle In Selection
        zelle.Value = StrConv(zelle.Value, vbLowerCase)
    Next zelle
End Sub
```

► Verwandte Funktionen

LCase, Ucase, Str

StrReverse-Funktion

Die Funktion `StrReverse` gibt eine Zeichenfolge zurück, in der die Reihenfolge der Zeichen einer bestimmten Zeichenfolge umgekehrt wurde.

► Syntax

`StrReverse(expression)`

Das Argument `expression` ist die Zeichenfolge, deren Zeichen umgekehrt werden sollen. Wenn es sich bei `expression` um eine Zeichenfolge der Länge Null ("") handelt, wird eine Zeichenfolge der Länge Null zurückgegeben. Wenn `expression` Null ist, kommt es zu einem Fehler.

► Beispiel

Im folgenden Beispiel aus Listing 3.54 wird ein Text gespiegelt.

Listing 3.54: Text spiegeln mit der Funktion StrReverse

```
Sub StrReverse_Beispiel()  
Dim Text1 As String  
  
Text1 = "Esel"  
Debug.Print "Vorher: " & Text1  
Debug.Print "Nachher: " & StrReverse(Text1)  
End Sub
```



Abbildung 3.29: Ein Text wurde gedreht.

► Verwandte Funktionen

Keine

String-Funktion

Die Funktion `String` gibt eine Zeichenfolge vom Typ `Variant (String)` zurück, die ein sich wiederholendes Zeichen der angegebenen Länge enthält.

► Syntax

`String(number, character)`

Die Syntax der `String`-Funktion besteht aus folgenden benannten Argumenten:

Teil	Beschreibung
Number	Erforderlich; Wert vom Typ <code>Long</code> . Länge der zurückgegebenen Zeichenfolge. Wenn <code>number</code> den Wert Null enthält, wird Null zurückgegeben.
Character	Erforderlich; Wert vom Typ <code>Variant</code> . Ein Zeichen-Code, der ein Zeichen oder einen Zeichenfolgenausdruck angibt, dessen erstes Zeichen verwendet wird, um die zurückzugebende Zeichenfolge zu erstellen. Wenn <code>character</code> den Wert Null enthält, wird Null zurückgegeben.

Tabelle 3.32: Argumente der Funktion `String`

► Beispiel

Im folgenden Beispiel aus Listing 3.55 werden alle markierten Zellen verarbeitet. Haben die Zellen eine Länge kleiner 10, dann werden die restlichen Zeichen durch einen Stern vervollständigt.

Listing 3.55: Texte vervollständigen über die Funktion `String`

```
Sub String_Beiispiel()
    Dim Zelle As Range

    For Each Zelle In Selection
        If Len(Zelle.Value) < 10 Then
            Zelle.Value = _
                Zelle.Value & String(10 - Len(Zelle.Value), "*")
        End If
    Next Zelle
End Sub
```

► **Verwandte Funktionen**

Mod, Space, Chr

Switch-Funktion

Die Funktion `Switch` wertet eine Liste von Ausdrücken aus und gibt einen Wert vom Typ `Variant` oder einen Ausdruck zurück, der dem ersten Ausdruck in der Liste zugeordnet ist, der `True` ergibt

► **Syntax**

`Switch(Ausdr-1, Wert-1[, Ausdr-2, Wert-2 ...
[, Ausdr-n, Wert-n]])`

Die Syntax der `Switch`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Ausdr	Erforderlich. <code>Variant</code> -Ausdruck, der ausgewertet werden soll.
Wert	Erforderlich. Wert oder Ausdruck, der zurückgegeben werden soll, wenn der entsprechende Ausdruck <code>True</code> ergibt.

Tabelle 3.33: Argumente der Funktion `Switch`

► **Beispiel**

Im folgenden Beispiel aus Listing 3.56 soll über eine Inputbox ein Land eingegeben werden. Über die Funktion `Switch` wird dann die dazugehörige Sprache des Landes ermittelt.

Listing 3.56: Zuordnungen treffen über die Funktion `Switch`

```
Sub Switch_Beiispiel()  
Dim s As String  
  
s = InputBox("Bitte ein Land eingeben!")  
On Error GoTo Fehler  
s = Switch _  
    (s = "Deutschland", "Deutsch", _  
    s = "Frankreich", "Französisch", _  
    s = "Italien", "Italienisch", _  
    s = "Spanien", "Spanisch")  
MsgBox "Sie sprechen " & s
```

```
Exit Sub
Fehler:
  MsgBox "Das Land ist nicht eingepflegt!"
End Sub
```

Trim-Funktion

Die Funktion `Trim` gibt einen Wert vom Typ `Variant (String)` zurück, der eine Kopie einer bestimmten Zeichenfolge enthält, die keine führenden Leerzeichen (`LTrim`), keine nachgestellten Leerzeichen (`RTrim`) sowie keine Kombination aus führenden und nachgestellten Leerzeichen (`Trim`) enthält.

► Syntax

```
LTrim(Zeichenfolge)
RTrim(Zeichenfolge)
Trim(Zeichenfolge)
```

Das erforderliche Argument `Zeichenfolge` ist ein beliebiger gültiger Zeichenfolgenausdruck. Wenn `Zeichenfolge` den Wert `Null` enthält, wird `Null` zurückgegeben.

► Beispiel

Im folgenden Beispiel aus Listing 3.57 werden in allen benutzten Zellen der `TABELLE10` führende sowie nachfolgende Leerzeichen entfernt.

Listing 3.57: Alle führenden und nachfolgenden Leerzeichen entfernen

```
Sub Trim_Beiispiel()
Dim Zelle As Range

For Each Zelle In Sheets("Tabelle10").UsedRange
  Zelle.Value = Trim(Zelle.Value)
Next Zelle
End Sub
```

► Verwandte Funktionen

`Left`, `Right`

UCase-Funktion

Die Funktion `UCase` wandelt Kleinbuchstaben in Großbuchstaben um.

► Syntax

`UCase(Zeichenfolge)`

Das erforderliche Argument `Zeichenfolge` ist ein beliebiger gültiger Zeichenfolgenausdruck. Wenn `Zeichenfolge` den Wert Null enthält, wird Null zurückgegeben.

► Beispiel

Siehe Funktion `LCase`

► Verwandte Funktionen

`LCase`, `StrConv`

3.3 Dateifunktionen und Anweisungen

Um vorab einen Überblick über diese Funktionen zu bekommen, werfen Sie einen Blick auf die Tabelle 3.34. Danach erfolgen die Erklärung der Syntax und die Demonstration anhand eines kurzen Beispiels.

Funktion/ Anweisung	Kurzbeschreibung
<code>ChDir (A)</code>	Wechselt das aktuelle Verzeichnis oder den aktuellen Ordner.
<code>ChDrive (A)</code>	Wechselt das aktuelle Laufwerk.
<code>Close (A)</code>	Beendet das Lesen aus und das Schreiben in eine Datei, die mit der <code>Open</code> -Anweisung geöffnet wurde.
<code>CurDir (A)</code>	Gibt einen Wert vom Typ <code>Variant (String)</code> zurück, der den aktuellen Pfad darstellt.

Tabelle 3.34: Funktionen und Anweisungen der Kategorie Dateifunktionen und -anweisungen

Funktion/ Anweisung	Kurzbeschreibung
Dir (F)	Gibt eine Zeichenfolge (String) zurück, die den Namen einer Datei, eines Verzeichnisses oder eines Ordners darstellt, der mit einem bestimmten Suchmuster, einem Dateiattribut oder mit der angegebenen Datenträger- bzw. Laufwerkbezeichnung übereinstimmt
Environ (F)	Gibt die mit einer Betriebssystem-Umgebungsvariablen verbundene Zeichenfolge (String) zurück.
EOF (F)	Gibt einen Wert vom Typ Integer zurück, der den Boolean-Wert True enthält, wenn das Ende einer Datei, die im Zugriffsmodus Random oder Input geöffnet wurde, erreicht worden ist.
FileAttr (F)	Gibt einen Wert vom Typ Long zurück, der den Zugriffsmodus für mit der Open-Anweisung geöffnete Dateien darstellt.
FileCopy (A)	Kopiert eine Datei.
FileDateTime (F)	Gibt einen Wert vom Typ Variant (Date) zurück, der den Tag und die Uhrzeit der Erstellung bzw. der letzten Änderung der Datei anzeigt.
FileLen (F)	Gibt einen Wert vom Typ Long zurück, der die Länge einer Datei in Bytes angibt.
FreeFile (F)	Gibt einen Wert vom Typ Integer zurück, der die nächste verfügbare Dateinummer darstellt, die die Open-Anweisung zum Öffnen einer Datei verwenden kann.
Get (A)	Liest Daten aus einer geöffneten Datenträgerdatei in eine Variable ein.
GetAttr (F)	Gibt einen Wert vom Typ Integer zurück, der die Attribute einer Datei, eines Verzeichnisses darstellt.
Input (F)	Gibt einen Wert vom Typ String zurück, der Zeichen aus einer im Modus Input oder Binary geöffneten Datei enthält.
Kill (A)	Löscht eine Datei ohne Rückfrage.
Line Input (A)	Liest eine einzelne Zeile aus einer geöffneten sequentiellen Datei und weist sie einer Variablen vom Typ String zu.

Tabelle 3.34: Funktionen und Anweisungen der Kategorie Dateifunktionen und -anweisungen (Forts.)

**Funktion/
Anweisung****Kurzbeschreibung**

Loc (F)	Gibt einen Wert vom Typ <code>Long</code> zurück, der die aktuelle Schreib-/Leseposition innerhalb einer geöffneten Datei angibt.
Lock (A)	Regelt die Zugriffsmöglichkeiten anderer Prozesse auf eine Datei (oder auf Teile einer Datei), die mit der <code>Open</code> -Anweisung geöffnet wurde.
Unlock (A)	
LOF (F)	Gibt einen Wert vom Typ <code>Long</code> zurück, der die Größe einer mit der <code>Open</code> -Anweisung geöffneten Datei in Bytes angibt.
LSet (A)	Richtet eine Zeichenfolge innerhalb einer Zeichenfolgenvariablen links aus oder kopiert eine Variable eines benutzerdefinierten Datentyps in eine Variable eines anderen benutzerdefinierten Datentyps.
MkDir (A)	Erstellt ein neues Verzeichnis.
Name (A)	Benennt eine Datei, ein Verzeichnis oder einen Ordner um.
Open (A)	Öffnet eine Datei für die Ein- bzw. Ausgabe.
Print (A)	Schreibt Daten, die für die Ausgabe formatiert sind, in eine sequentielle Datei.
Put (A)	Schreibt Daten aus einer Variablen in eine Datenträgerdatei.
Reset (A)	Schließt alle Datenträgerdateien, die mit der <code>Open</code> -Anweisung geöffnet wurden.
RmDir (A)	Löscht ein Verzeichnis.
Seek (F)	Gibt einen Wert vom Typ <code>Long</code> zurück, der die aktuelle Schreib-/Leseposition in einer Datei festlegt, die mit der <code>Open</code> -Anweisung geöffnet wurde.
SetAttr (A)	Legt die Dateiattribute fest.
Shell (F)	Führt ein ausführbares Programm aus. Falls erfolgreich, gibt sie einen Wert vom Typ <code>Variant (Double)</code> zurück, der die Task-ID des Programms darstellt. Andernfalls wird Null zurückgegeben.
Write (A)	Schreibt Daten in eine sequentielle Datei.

Tabelle 3.34: Funktionen und Anweisungen der Kategorie Dateifunktionen und -anweisungen (Forts.)

ChDir-Anweisung

Mithilfe der Anweisung `ChDir` können Sie in das aktuelle Verzeichnis wechseln.

► Syntax

`ChDir Pfad`

Das erforderliche Argument `Pfad` ist ein Zeichenfolgenausdruck, der angibt, welches Verzeichnis zum neuen Standardverzeichnis wird.

► Beispiel

Im folgenden Beispiel aus Listing 3.58 wird das Standardverzeichnis neu gesetzt.

Listing 3.58: Mit ChDir das Standardverzeichnis setzen

```
Sub Chdir_Beispiel()
On Error GoTo Fehler
    ChDir "C:\Held\"
Exit Sub
Fehler:
    MsgBox "Dieses Verzeichnis gibt es nicht!"
End Sub
```

► Verwandte Funktionen/Anweisungen

`ChDrive`, `CurDir`, `Dir`, `MkDir`, `RmDir`

ChDrive-Anweisung

Mithilfe der Anweisung `ChDrive` stellen Sie das aktuelle Laufwerk ein.

► Syntax

`ChDrive Laufwerk`

Das erforderliche Argument `Laufwerk` ist ein Zeichenfolgenausdruck, der ein existierendes Laufwerk angibt.

► Beispiel

Im folgenden Beispiel aus Listing 3.59 wird das Laufwerk D voreingestellt.

```
Sub Chdrive_Beispiel()  
On Error GoTo Fehler  
    ChDrive "d:"  
    Exit Sub  
Fehler:  
    MsgBox "Dieses Laufwerk gibt es nicht!"  
End Sub
```

► Verwandte Funktionen/Anweisungen

ChDir, CurDir, Dir, Mkdir, Rmdir

Close-Anweisung

Die Anweisung `Close` beendet das Lesen aus und das Schreiben in eine(r) Datei und schließt diese.

► Syntax

`Close [Dateinummerliste]`

Das optionale Argument `Dateinummerliste` entspricht einer oder mehreren Dateinummern mit der folgenden Syntax, in der `Dateinummer` eine beliebige gültige Dateinummer ist:

`[[#]Dateinummer] [, [#]Dateinummer] . . .`

► Beispiel

Siehe Beispiel Anweisung `Open`

► Verwandte Funktionen/Anweisungen

`Open`, `Reset`

CurDir-Anweisung

Über die Anweisung `CurDir` können Sie das aktuell eingestellte Verzeichnis ermitteln.

► Syntax

`CurDir[(Laufwerk)]`

Das optionale Argument `Laufwerk` ist ein Zeichenfolgenausdruck, der ein existierendes Laufwerk angibt. Ist kein Laufwerk angegeben,

oder enthält das Laufwerk eine Null-Zeichenfolge (""), so gibt die CurDir-Funktion den Pfad des aktuellen Laufwerks zurück.

► Beispiel

Im folgenden Beispiel aus Listing 3.60 wird das momentan eingestellte Verzeichnis am Bildschirm ausgegeben.

Listing 3.60: Mit CurDir das aktuell eingestellte Verzeichnis auslesen

```
Sub CurDir_Beispiel()  
    MsgBox _  
    "Das aktuell eingestellte Verzeichnis lautet: " & _  
    CurDir("C")  
End Sub
```

► Verwandte Funktionen/Anweisungen

ChDir, ChDrive, Dir, Mkdir, Rmdir

Dir-Funktion

Die Funktion Dir gibt eine Zeichenfolge (String) zurück, die den Namen einer Datei, eines Verzeichnisses oder eines Ordners darstellt, der mit einem bestimmten Suchmuster, einem Dateiattribut oder mit der angegebenen Datenträger- bzw. Laufwerkbezeichnung übereinstimmt.

► Syntax

Dir[(Pfadname[, Attribute])]

Die Syntax der Dir-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Pfadname	Optional. Zeichenfolgendruck, der einen Dateinamen angibt. Der Dateiname kann ein Verzeichnis oder einen Ordner sowie ein Laufwerk enthalten. Eine Null-Zeichenfolge (») wird zurückgegeben, wenn der Pfad aus Pfadname nicht gefunden werden kann

Tabelle 3.35: Argumente der Funktion Dir

Teil	Beschreibung
Attribute	Optional. Eine Konstante oder ein numerischer Ausdruck, deren Summe die Dateiattribute angibt. Wenn das Argument nicht angegeben wird, werden alle Dateien, die dem Argument <code>Pfadname</code> entsprechen, zurückgegeben.

Tabelle 3.35: Argumente der Funktion Dir (Forts.)

Das Argument `Attribute` hat die folgenden Einstellungen:

Konstante	Wert	Beschreibung
<code>vbNormal</code>	0	(Voreinstellung) Dateien ohne Attribute
<code>vbReadOnly</code>	1	Schreibgeschützte Dateien, zusätzlich zu Dateien ohne Attribute
<code>vbHidden</code>	2	Versteckte Dateien, zusätzlich zu Dateien ohne Attribute
<code>VbSystem</code>	4	Systemdatei, zusätzlich zu Dateien ohne Attribute; beim Macintosh nicht verfügbar.
<code>vbVolume</code>	8	Datenträgerbezeichnung. Falls andere Attribute angegeben wurden, wird <code>vbVolume</code> ignoriert; beim Macintosh nicht verfügbar.
<code>vbDirectory</code>	16	Verzeichnis oder Ordner, zusätzlich zu Dateien ohne Attribute.

Tabelle 3.36: Die möglichen Einstellungen beim Argument Attribute

► Beispiel

Im folgenden Beispiel aus Listing 3.61 werden aus einem Verzeichnis alle Dateien ausgelesen und das Datum der Erstellung einer jeden Datei protokolliert.

Listing 3.61: Dateien aus einem Verzeichnis auslesen

```
Sub Dir_Beispiel()
Dim blatt As Worksheet
Dim sAttr As String
Dim sPfad As String
Dim sName As String
Dim i As Integer
```

```

Set blatt = ThisWorkbook.Worksheets("Tabelle10")
sAttr = vbNormal + vbReadOnly + vbHidden

sPfad = "c:\Eigene Dateien\"
sName = Dir(sPfad, sAttr)
With blatt
    .Cells(1, 1) = "Name"
    .Cells(2, 1) = sName
    .Cells(1, 2) = "Datum"
    i = 2
On Error Resume Next
Do While sName <> ""
    If sName <> "." And sName <> ".." Then
        .Cells(i, 1) = sName
        .Cells(i, 2) = FileDateTime(sPfad & sName)
        i = i + 1
    End If
    sName = Dir
Loop
End With
End Sub

```

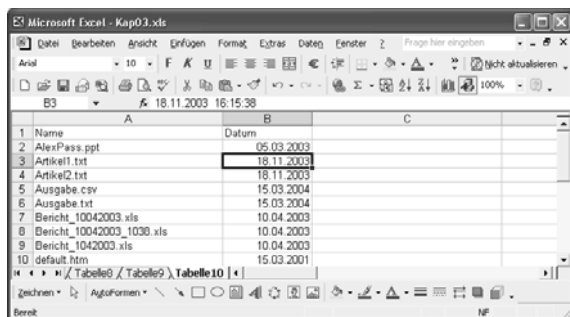


Abbildung 3.30: Dateien eines Verzeichnisses wurden aufgelistet

► Verwandte Funktionen/Anweisungen

ChDir, CurDir

Environ-Anweisung

Die Anweisung `Environ` gibt die mit einer Betriebssystem-Umgebungsvariablen verbundene Zeichenfolge (String) zurück.

► Syntax

`Environ({envstring | number})`

Die Syntax der `Environ`-Funktion verwendet die folgenden benannten Argumente:

Teil	Beschreibung
Envstring	Optional. Ein Zeichenfolenausdruck, der den Namen einer Umgebungsvariablen enthält.
Number	Optional. Ein numerischer Ausdruck entsprechend der numerischen Reihenfolge der Umgebungszeichenfolge in der Tabelle für Umgebungszeichenfolgen. Das Argument <code>number</code> kann ein beliebiger numerischer Ausdruck sein, der aber vor der Auswertung auf eine ganze Zahl gerundet wird.

Tabelle 3.37: Argumente der Anweisung `Environ`

► Beispiel

Im folgenden Beispiel aus Listing 3.62 werden alle Umgebungsvariablen des Systems am Bildschirm ausgegeben.

Listing 3.62: Umgebungsvariablen ermitteln und ausgeben

```
Sub Environ_Beispiel()
    Dim i As Integer
    Dim s As String

    i = 1
    Do Until Environ(i) = ""
        s = s & vbCrLf & Environ(i)
        i = i + 1
    Loop
    MsgBox s
End Sub
```



Abbildung 3.31: Alle Umgebungsvariablen des Systems im Überblick

EOF-Funktion

Diese Funktion gibt den Wert `True` zurück, wenn das Ende einer Datei erreicht ist.

► Syntax

`EOF(Dateinummer)`

Das erforderliche Argument `Dateinummer` ist ein Wert vom Typ `Integer`, der eine beliebige gültige Dateinummer enthält.

► Beispiel

Im folgenden Beispiel aus Listing 3.63 wird eine Textdatei Zeile für Zeile gelesen und im Direktfenster der Entwicklungsumgebung ausgegeben.

Listing 3.63: Textdatei Zeile für Zeile auslesen

```
Sub EOF_Beispiel()  
Dim s As String  
  
Open "C:\Artikel.txt" For Input As #1
```

```

Do While Not EOF(1)
    Line Input #1, s
    Debug.Print s
Loop
Close #1
End Sub

```

► Verwandte Funktionen/Anweisungen

LOF, Loc, Get, Line Input, Open, Seek

FileAttr-Funktion

Über die Funktion `FileAttr` können Sie den Zugriffsmodus einer geöffneten Datei feststellen.

► Syntax

`FileAttr(filenummer, returntype)`

Die Syntax der `FileAttr`-Funktion verwendet die folgenden benannten Argumente:

Teil	Beschreibung
Filenummer	Erforderlich; ein Wert vom Typ <code>Integer</code> . Eine beliebige gültige Dateinummer.
Returntype	Erforderlich; ein Wert vom Typ <code>Integer</code> . Zahl, die die Art der zurückzugebenden Informationen festlegt. Dabei sind folgende Rückgaben möglich: Input = 1 Output = 2 Random = 4 Append = 8 Binary = 32

Tabelle 3.38: Argumente der Funktion `FileAttr`

► Verwandte Funktionen/Anweisungen

Open

FileCopy-Anweisung

Mithilfe der Anweisung `FileCopy` können Sie Dateien kopieren.

► Syntax

`FileCopy source, destination`

Die Syntax der `FileCopy`-Anweisung verwendet die folgenden benannten Argumente:

Teil	Beschreibung
Source	Erforderlich. Zeichenfolgenausdruck, der den Namen der zu kopierenden Datei angibt. <i>Source</i> kann ein Verzeichnis oder einen Ordner sowie ein Laufwerk enthalten.
Destination	Erforderlich. Zeichenfolgenausdruck, der den Namen der Zieldatei angibt. <i>Destination</i> kann ein Verzeichnis oder einen Ordner sowie ein Laufwerk enthalten.

Tabelle 3.39: Argumente der Anweisung `FileCopy`

► Beispiel

Beim folgenden Beispiel aus Listing 3.64 wird eine Textdatei kopiert.

Listing 3.64: Textdatei kopieren mit der Anweisung `FileCopy`

```
Sub KopierenDateien()
    Dim sDat As String
    Dim sDat_Kopie As String

    sDat = "C:\Artikel.txt"
    sDat_Kopie = "Kopie.txt"

    FileCopy sDat, sDat_Kopie
    Exit Sub

fehler1:
    MsgBox "Fehler beim Kopieren aufgetreten!"
End Sub
```

FileDateTime-Funktion

Die Funktion `FileDateTime` gibt einen Wert vom Typ `Variant (Date)` zurück, der den Tag und die Uhrzeit der Erstellung bzw. der letzten Änderung der Datei anzeigt.

► Syntax

`FileDateTime(Pfadname)`

Das erforderliche Argument `Pfadname` ist ein Zeichenfolgenausdruck, der einen Dateinamen angibt. `Pfadname` kann ein Verzeichnis sowie ein Laufwerk enthalten.

► Beispiel

Im folgenden Beispiel aus Listing 3.65 wird das Erstellungsdatum der aktiven Arbeitsmappe ausgegeben.

Listing 3.65: Das Erstellungsdatum der aktiven Mappe ermitteln

```
Sub FileDateTime_Beiispiel()  
    MsgBox FileDateTime(ActiveWorkbook.FullName)  
End Sub
```

► Verwandte Funktionen/Anweisungen

`FileLen`, `GetAttr`

FileLen-Funktion

Die Funktion `FileLen` gibt einen Wert vom Typ `Long` zurück, der die Länge einer Datei in Bytes angibt.

► Syntax

`FileLen(Pfadname)`

Das erforderliche Argument `Pfadname` ist ein Zeichenfolgenausdruck, der eine Datei angibt. `Pfadname` kann ein Verzeichnis oder ein Laufwerk enthalten.

► Beispiel

Beim folgenden Beispiel aus Listing 3.66 wird die Funktion `FileLen` eingesetzt, um zu prüfen, ob eine bestimmte Datei überhaupt existiert.

Listing 3.66: Existenzprüfung einer Datei über die Funktion FileLen

```
Sub FileLen_Beispiel()  
Dim i As Integer  
  
On Error GoTo Fehler  
i = FileLen("C:\Artikel.txt")  
'Weitere Aktionen  
Exit Sub  
  
Fehler:  
MsgBox "Diese Datei existiert nicht!"  
End Sub
```

► Verwandte Funktionen/Anweisungen

FileDateTime, GetAttr, LOF

FreeFile-Funktion

Die Funktion **FreeFile** gibt einen Wert vom Typ **Integer** zurück, der die nächste verfügbare Dateinummer darstellt, die die **Open**-Anweisung zum Öffnen einer Datei verwenden kann.

► Syntax

FreeFile[(Bereichsnummer)]

Das optionale Argument **Bereichsnummer** ist ein Wert vom Typ **Variant**, der den Bereich festlegt, aus dem die nächste Dateinummer zurückgegeben werden wird. Bei 0 (Voreinstellung) wird eine Dateinummer im Bereich 1-255 (einschließlich) zurückgegeben. Bei 1 wird eine Dateinummer im Bereich 256-511 zurückgegeben.

► Beispiel

Siehe Beispiel **Open**-Anweisung

► Verwandte Funktionen/Anweisungen

Open, **Close**

Get-Anweisung

Die Anweisung `Get` liest Daten aus einer geöffneten Datenträgerdatei in eine Variable ein.

► Syntax

`Get [#]Dateinummer, [Satznummer], Variablennummer`

Die Syntax der `Get`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.
Satznummer	Optional. Wert vom Typ <code>Variant (Long)</code> . Datensatznummer (Dateien im Modus <code>Random</code>) oder Byte-Nummer (Dateien im Modus <code>Binary</code>), an der der Lesevorgang beginnt.
Variablennummer	Erforderlich. Name einer gültigen Variablen, in die die Daten eingelesen werden

Tabelle 3.40: Argumente der Anweisung `Get`

► Beispiel

Beim folgenden Beispiel aus Listing 3.67 wird die erste Zeile einer Textdatei gelesen und in eine Variable geschrieben.

Listing 3.67: Erste Zeile einer Textdatei lesen

```
Type Datensatz
    Kennung As Integer
    Name As String * 37
End Type

Sub Get_Beiispiel()
    Dim DSatz1 As Datensatz
    Dim Position As Integer
    Dim s As String

    Open "C:\Artikel.txt" For Random As #1
    Position = 1
    Get #1, Position, DSatz1
    Close #1
End Sub
```

GetAttr-Anweisung

Die Anweisung `GetAttr` gibt einen Wert vom Typ `Integer` zurück, der die Attribute einer Datei bzw. eines Verzeichnisses darstellt.

► Syntax

`GetAttr(Pfadname)`

Das erforderliche Argument `Pfadname` ist ein Zeichenfolgenausdruck, der einen Dateinamen angibt. `Pfadname` kann ein Verzeichnis oder ein Laufwerk enthalten.

Der von `GetAttr` zurückgegebene Wert ist die Summe der folgenden Attributwerte:

Konstante	Wert	Beschreibung
<code>vbNormal</code>	0	Normal
<code>vbReadOnly</code>	1	Schreibgeschützt
<code>vbHidden</code>	2	Versteckt
<code>vbSystem</code>	4	Systemdatei; beim Macintosh nicht verfügbar.
<code>vbDirectory</code>	16	Verzeichnis oder Ordner
<code>vbArchive</code>	32	Datei wurde seit dem letzten Sichern geändert.
<code>vbAlias</code>	64	Angegebener Dateiname ist ein Alias; nur beim Macintosh verfügbar.

Tabelle 3.41: Argumente der Anweisung `GetAttr`

► Beispiel

Im folgenden Beispiel aus Listing 3.68 wird eine bestimmte Datei aus dem Windows-Verzeichnis überprüft.

Listing 3.68: Handelt es sich um eine versteckte Datei?

```
Sub GetAttr_Beispiel()
    Dim attribut As VbFileAttribute

    attribut = vbHidden
    If attribut And GetAttr _
        ("C:\Windows\WindowsShell.Manifest") Then
        MsgBox "Datei ist verborgen"
    Else
```

```
MsgBox "Datei ist nicht verborgen"
End If
End Sub
```

► Verwandte Funktionen/Anweisungen

SetAttr, FileAttr

3.3

Input-Funktion

Die Funktion `Input` gibt einen Wert vom Typ `String` zurück, der Zeichen aus einer im Modus `Input` oder `Binary` geöffneten Datei enthält.

► Syntax

```
Input(Zahl, [#]Dateinummer)
```

Die Syntax der `Input`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Zahl	Erforderlich. Ein beliebiger gültiger numerischer Ausdruck, der die Zahl der zurückzugebenden Zeichen angibt.
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.

Tabelle 3.42: Argumente der Funktion Input

► Beispiel

Im folgenden Beispiel aus Sub `Input_Beispiel()`

```
Dim Zeichen As String
Open »C:\Artikel.txt« For Input As #1
Zeichen = Input(7, #1)
Debug.Print Zeichen
Close #1
End Sub
```

In Listing 3.69 werden die ersten 7 Zeichen der ersten Zeile einer Textdatei in das Direktfenster der Entwicklungsumgebung gelesen.

Listing 3.69: Erste Zeile einer Textdatei einlesen über die Anweisung Input

```
Sub Input_Beispiel()  
Dim Zeichen As String  
  
Open "C:\Artikel.txt" For Input As #1  
Zeichen = Input(7, #1)  
Debug.Print Zeichen  
Close #1  
End Sub
```

► Verwandte Funktionen/Anweisungen

Line Input

Kill-Anweisung

Die Anweisung `Kill` löscht eine Datei ohne Rückfrage von der Festplatte.

► Syntax

`Kill Pfadname`

Das erforderliche Argument `Pfadname` ist ein Zeichenfolgenausdruck, der eine oder mehrere zu löschende Dateien angibt. `Pfadname` kann ein Verzeichnis sowie ein Laufwerk enthalten.

► Beispiel

Im folgenden Beispiel wird eine Datei aus einem bestimmten Verzeichnis entfernt.

Listing 3.70: Mit der Anweisung Kill Dateie(en) löschen

```
Sub Kill_Beispiel()  
On Error Resume Next  
Kill "C:\Eigene Dateien\Artikel.txt"  
End Sub
```

► Verwandte Funktionen/Anweisungen

Rmdir

Line Input-Anweisung

Mithilfe der Anweisung `Line Input` können Sie eine einzelne Zeile aus einer geöffneten sequentiellen Datei auslesen.

► Syntax

`Line Input #Dateinummer, Variablenname`

Die Syntax der `Line Input #-Anweisung` besteht aus folgenden Teilen:

Teil	Beschreibung
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.
Variablenname	Erforderlich. Ein gültiger Variablenname vom Typ Variant oder String.

Tabelle 3.43: Argumente der Anweisung `Line Input`

► Beispiel

Im folgenden Beispiel aus Listing 3.71 wird die erste Zeile einer Textdatei gelesen und im Direktfenster der Entwicklungsumgebung ausgegeben.

Listing 3.71: Die erste Zeile einer Textdatei auslesen

```
Sub LineInput_Beiispiel()
Dim sTextzeile As String

Open "c:\Artikel.txt" For Input As #1
Line Input #1, sTextzeile
Debug.Print sTextzeile
Close #1
End Sub
```

► Verwandte Funktionen/Anweisungen

`Input`

LOC-Funktion

Die Funktion `LOC` gibt einen Wert vom Typ `Long` zurück, der die aktuelle Schreib-/Lese-Position innerhalb einer geöffneten Datei angibt.

► **Syntax**

Loc(Dateinummer)

Das erforderliche Argument `Dateinummer` ist eine beliebige gültige Dateinummer vom Typ Integer.

Die folgende Aufstellung beschreibt die Rückgabewerte für die einzelnen Dateizugriffsmodi:

Zugriffsmodus	Rückgabewert
Random	Nummer des letzten Datensatzes, der aus der Datei gelesen oder in die Datei geschrieben wurde.
Sequential	Aktuelle Byte-Position in der Datei dividiert durch 128. Informationen, die von <code>Loc</code> für sequentielle Dateien zurückgegeben werden, werden jedoch nicht verwendet und sind auch nicht erforderlich.
Binary	Position des letzten gelesenen oder geschriebenen Byte.

Tabelle 3.44: Rückgabewerte der Funktion LOC

► **Beispiel**

Im folgenden Beispiel aus Listing 3.72 wird eine Textdatei in das Direktfenster der Entwicklungsumgebung eingelesen. Dabei wird die Schreib-/Lese-Position mit ausgegeben.

Listing 3.72: Textdatei einlesen und Schreibposition festhalten

```
Sub LOC_Beispiel()
    Dim Position1 As Integer
    Dim Zeile1 As Variant

    Open "C:\Artikel.txt" For Binary As #1
    Do While Not EOF(1)
        Line Input #1, Zeile1
        Position1 = Loc(1)
        Debug.Print Zeile1; Tab; Position1
    Loop
    Close #1
End Sub
```

► **Verwandte Funktionen/Anweisungen**

EOF, Seek

Lock- und Unlock-Anweisung

Diese beiden Anweisungen regeln die Zugriffsmöglichkeiten anderer Prozesse auf eine Datei (oder auf Teile einer Datei), die mit der `Open`-Anweisung geöffnet wurde.

► Syntax

`Lock [#]Dateinummer[, Satzbereich]`

`Unlock [#]Dateinummer[, Satzbereich]`

Die Syntax der Anweisungen `Lock` und `Unlock` besteht aus folgenden Teilen:

Teil	Beschreibung
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.
Satzbereich	Optional. Der Datensatzbereich, für den die Sperre eingerichtet oder aufgehoben werden soll.

Tabelle 3.45: Argumente der Anweisungen Lock und Unlock

Die Einstellungen für das Argument `Satzbereich` sind:

`Satznummer | [Anfang] To Ende`

Einstellung	Beschreibung
Satznummer	Nummer des Datensatzes (in Dateien mit dem Modus <code>Random</code>) oder Byte-Nummer (in Dateien mit dem Modus <code>Binary</code>), an der das Einrichten/Aufheben der Sperre beginnt.
Anfang	Nummer des ersten Datenelements (Datensatz oder Byte), für das die Sperre eingerichtet/aufgehoben werden soll.
Ende	Nummer des letzten Datenelements (Datensatz oder Byte), für das die Sperre eingerichtet/aufgehoben werden soll.

Tabelle 3.46: Einstellungen für das Argument Satzbereich

► Verwandte Funktionen/Anweisungen

`Open`

LSet-Anweisung

Die Anweisung `LSet` richtet eine Zeichenfolge innerhalb einer Zeichenfolgenvariablen links aus oder kopiert eine Variable eines benutzerdefinierten Datentyps in eine Variable eines anderen benutzerdefinierten Datentyps.

► Syntax

`LSet ZnFVariable = Zeichenfolge`

`LSet VariablenName1 = VariablenName2`

Die Syntax der `LSet`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
<code>ZnFVariable</code>	Erforderlich. Name einer Zeichenfolgenvariablen.
Zeichenfolge	Erforderlich. Zeichenfolgenderausdruck, der innerhalb von <code>ZnFVariable</code> links ausgerichtet werden soll.
<code>VariablenName1</code>	Erforderlich. Name einer Variablen des benutzerdefinierten Datentyps, die die Zielvariable für den Kopiervorgang darstellt.
<code>VariablenName2</code>	Erforderlich. Name einer Variablen des benutzerdefinierten Datentyps, die die Quellvariable für den Kopiervorgang darstellt.

Tabelle 3.47: Argumente der Anweisung `LSet`

► Verwandte Funktionen/Anweisungen

`RSet`

MkDir-Anweisung

Mithilfe der Anweisung `MkDir` können Sie ein neues Verzeichnis erstellen.

► Syntax

`MkDir Pfad`

Das erforderliche Argument `Pfad` ist ein Zeichenfolgenderausdruck, der das zu erstellende Verzeichnis angibt. `Pfad` kann das Laufwerk enthalten. Wenn kein Laufwerk angegeben ist, erstellt die `MkDir`-Anweisung ein neues Verzeichnis oder einen neuen Ordner auf Basis des aktuellen Laufwerks.

► **Beispiel**

Im folgenden Beispiel aus Listing 3.73 wird ein neues Verzeichnis angelegt. Für den Fall, dass dieses Verzeichnis bereits existiert, fangen Sie den Fehler über die `On Error Resume Next`-Anweisung ab.

Listing 3.73: Neues Verzeichnis über die Anweisung Mkdir erstellen

```
Sub Mkdir_Beispiel()  
On Error Resume Next  
Mkdir "C:\Eigene Dateien\Versuch"  
End Sub
```

► **Verwandte Funktionen/Anweisungen**

ChDir, CurDir, Rmdir

Name-Anweisung

Über die Anweisung `Name` können Sie eine Datei umbenennen.

► **Syntax**

`Name AlterPfadname As NeuerPfadname`

Die Syntax der `Name`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
AlterPfadname	Erforderlich. Zeichenfolgenausdruck, der einen existierenden Dateinamen und seinen Pfad angibt. In dem Wert kann ein Verzeichnis oder Ordner sowie ein Laufwerk enthalten sein.
NeuerPfadname	Erforderlich. Zeichenfolgenausdruck, der einen neuen Dateinamen und seinen Pfad angibt. In dem Wert kann ein Verzeichnis oder Ordner sowie ein Laufwerk enthalten sein. Die in NeuerPfadname angegebene Datei darf noch nicht existieren.

Tabelle 3.48: Argumente der Anweisung Name

► **Beispiel**

Im folgenden Beispiel aus Listing 3.74 wird eine Textdatei umbenannt.

Listing 3.74: Dateien umbenennen über die Anweisung Name

```

Sub Name_Beispiel()
Dim AlterName As String
Dim Neuename As String

AlterName = "C:\Artikel1.txt"
Neuename = "c:\ArtikelX.txt"

Name AlterName As Neuename
End Sub

```

► **Verwandte Funktionen/Anweisungen**

Kill

Open-Anweisung

Die Anweisung `Open` öffnet eine Datei für die Ein- bzw. Ausgabe.

► **Syntax**

`Open Pfadname For Modus [Access Zugriff] [Sperre] As [#]Dateinummer [Len=Satzlänge]`

Die Syntax der `Open`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
Pfadname	Erforderlich. Zeichenfolgenausdruck, der einen Dateinamen festlegt und auch Verzeichnis- oder Ordner- sowie Laufwerkangaben enthalten kann.
Modus	Erforderlich. Schlüsselwort, das den Zugriffsmodus für die Datei festlegt: <code>Append</code> , <code>Binary</code> , <code>Input</code> , <code>Output</code> oder <code>Random</code> . Wenn kein Modus festgelegt ist, wird die Datei im Zugriffsmodus <code>Random</code> geöffnet.
Zugriff	Optional. Schlüsselwort, das die Operationen festlegt, die auf der geöffneten Datei ausgeführt werden können: <code>Read</code> , <code>Write</code> oder <code>Read Write</code> .
Sperre	Optional. Schlüsselwort, das die Operationen festlegt, die von anderen Prozessen auf der geöffneten Datei ausgeführt werden können: <code>Shared</code> , <code>Lock Read</code> , <code>Lock Write</code> und <code>Lock Read Write</code> .

Tabelle 3.49: Argumente der Anweisung Open

Teil	Beschreibung
Dateinummer	Erforderlich. Eine gültige Dateinummer im Bereich von 1 bis 511 (einschließlich). Mit der <code>FreeFile</code> -Funktion erhalten Sie die nächste verfügbare Dateinummer.
Satzlänge	Optional. Zahl kleiner oder gleich 32.767 (Byte). Bei Dateien mit wahlfreiem Zugriff ist dies die Datensatzlänge, bei sequentiellen Dateien die Anzahl der gepufferten Zeichen.

Tabelle 3.49: Argumente der Anweisung `Open` (Forts.)

► Beispiel

Im folgenden Beispiel aus Listing 3.75 werden alle Formelzellen der TABELLE12 in eine Textdatei geschrieben.

Listing 3.75: Alle Formeln in einer Textdatei protokollieren

```
Sub Open_Beispiel()
Dim Zelle As Range

Open "C:\Formeln.txt" For Output As #1
For Each Zelle In Sheets("Tabelle12").UsedRange
    If Zelle.HasFormula Then Print #1, _
        Zelle.Address; Tab; Zelle.FormulaLocal
Next
Close #1
End Sub
```

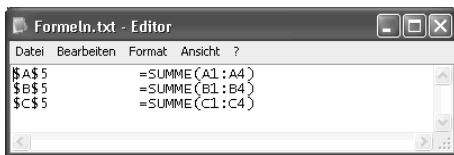


Abbildung 3.32: Formeln auslesen und in eine Textdatei schreiben

► Verwandte Funktionen/Anweisungen

`Close`, `FreeFile`

Print-Anweisung

Die Anweisung `Print` schreibt Daten, die für die Ausgabe formatiert sind, in eine sequentielle Datei.

► Syntax

`Print [#Dateinummer, [Ausgabeliste]`

Die Syntax der `Print`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.
Ausgabeliste	Optional. Ausdruck oder Liste mit Ausdrücken, die ausgegeben werden sollen.

Tabelle 3.50: Argumente der Anweisung `Print`

► Beispiel

Siehe Anweisung `Open`

► Verwandte Funktionen/Anweisungen

`Write`, `Spc`, `Tab`

Put-Anweisung

Die Anweisung `Put` schreibt Daten aus einer Variablen in eine Datenträgerdatei.

► Syntax

`Put [#]Dateinummer, [Satznummer], Variablennummer`

Die Syntax der `Put`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.
Satznummer	Optional. Wert vom Typ <code>Variant (Long)</code> . Datensatznummer (für Dateien im Modus <code>Random</code>) oder Byte-Nummer (für Dateien im Modus <code>Binary</code>), bei der der Schreibvorgang beginnt.
Variablenname	Erforderlich. Name der Variablen, welche die auf den Datenträger zu schreibenden Daten enthält.

Tabelle 3.51: Argumente der Anweisung `Put`

► **Beispiel**

Im folgenden Beispiel aus Listing 3.76 wird eine Datei angelegt und ein Mustersatz hineingeschrieben.

Listing 3.76: Datensatz schreiben mit der Anweisung Put

```
Type MeinDatensatz
    NName As String * 30
    VName As String * 30
End Type

Sub Put_Beiispiel()
    Dim Datensatz As MeinDatensatz
    Dim FileNum As Integer
    Dim i As Integer

    If Dir("C:\ArtikelX.txt") <> "" Then
        Kill "C:\ArtikelX.txt"
    End If
    FileNum = FreeFile
    Open "C:\ArtikelX.txt" For _
        Random As #FileNum Len = Len(Datensatz)
        With Datensatz
            .NName = "Vorname"
            .VName = "Nachname"
        End With
        Put #FileNum, , Datensatz
    Close #FileNum
End Sub
```

► **Verwandte Funktionen/Anweisungen**

Get, Open, Seek

Reset-Anweisung

Die Anweisung `Reset` schließt alle Datenträgerdateien, die mit der `Open`-Anweisung geöffnet wurden.

► Syntax

Reset

► Beispiel

Im folgenden Beispiel aus Listing 3.77 werden hintereinander drei Textdateien angelegt, die aktuelle Uhrzeit erfasst und am Ende über die Anweisung `Reset` gespeichert und geschlossen.

Listing 3.77: Mehrere Textdateien über die Anweisung Reset speichern und schließen

```
Sub Reset_Beispiel()  
Dim iDateiNr As Integer  
  
ChDir "C:\Eigene Dateien\  
For iDateiNr = 1 To 3  
    Open "Datei" & iDateiNr & ".txt" _  
        For Output As #iDateiNr  
        Write #iDateiNr, Now  
    Next iDateiNr  
    Reset  
End Sub
```

► Verwandte Funktionen/Anweisungen

Close

RmDir-Anweisung

Über die Anweisung `RmDir` können Sie ein Verzeichnis löschen.

► Syntax

`RmDir Pfad`

Das erforderliche Argument `Pfad` ist ein Zeichenfolgenausdruck, der das zu löschende Verzeichnis oder den zu löschenden Ordner angibt. `Pfad` kann das Laufwerk beinhalten. Wenn kein Laufwerk angegeben ist, löscht `RmDir` das Verzeichnis oder den Ordner auf dem aktuellen Laufwerk.

► **Hinweis**

Rmdir führt zu einem Fehler, wenn Sie die Anweisung für ein Verzeichnis ausführen, in dem Dateien enthalten sind. Löschen Sie zuerst alle Dateien mit der Kill-Anweisung, bevor Sie ein Verzeichnis oder einen Ordner entfernen.

► **Beispiel**

Im folgenden Beispiel aus Listing 3.78 wird ein Verzeichnis entfernt.

Listing 3.78: Über die Anweisung Rmdir ein Verzeichnis löschen

```
Sub Rmdir_Beispiel()  
On Error Resume Next  
  Rmdir "C:\Test"  
End Sub
```

► **Verwandte Funktionen/Anweisungen**

Kill, ChDir, CurDir, Mkdir

Seek-Funktion

Die Funktion Seek gibt einen Wert vom Typ Long zurück, der die aktuelle Schreib-/Lese-Position in einer Datei festlegt, die mit der Open-Anweisung geöffnet wurde.

► **Syntax**

```
Seek(Dateinummer)
```

Das erforderliche Argument *Dateinummer* ist ein Wert vom Typ Integer, der eine gültige Dateinummer enthält.

► **Beispiel**

Im folgenden Beispiel aus Listing 3.79 wird eine Textdatei ausgelesen und die entsprechenden Byte-Positionen jeweils am Satzende ermittelt.

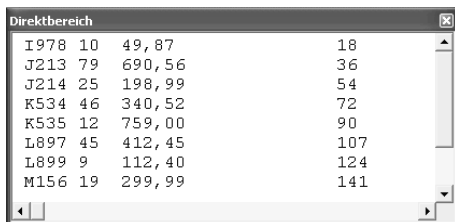
Listing 3.79: Byteposition über die Funktion Seek ermitteln

```
Sub Seek_Beispiel()  
Dim zeile As String
```

```

Open "C:\Artikel2.txt" For Input As #1
Do While Not EOF(1)
    Line Input #1, zeile
    Debug.Print zeile; Tab; Seek(1)
Loop
Close #1
End Sub

```



Line	Content	Position (Seek)
1	I978 10	49,87
2	J213 79	690,56
3	J214 25	198,99
4	K534 46	340,52
5	K535 12	759,00
6	L897 45	412,45
7	L899 9	112,40
8	M156 19	299,99

Abbildung 3.33: Inhalt und Position eines Datensatzes werden festgehalten

► Verwandte Funktionen/Anweisungen

Get, Loc, Put

SetAttr-Anweisung

Über die SetAttr-Anweisung können Sie Dateiattribute setzen.

► Syntax

SetAttr pathname, attributes

Die Syntax der SetAttr-Anweisung verwendet die folgenden benannten Argumente:

Teil	Beschreibung
Pathname	Erforderlich. Zeichenfolgenausdruck, der einen Dateinamen angibt. Der Dateiname kann ein Verzeichnis oder einen Ordner sowie ein Laufwerk enthalten.
Attributes	Erforderlich. Konstante oder numerischer Ausdruck, die/der die Summe der Dateiattribute angibt.

Tabelle 3.52: Argumente der Anweisung SetAttr

Das Argument `attributes` hat die folgenden Einstellungen:

Konstante	Wert	Beschreibung
<code>vbNormal</code>	0	Normal (Voreinstellung)
<code>vbReadOnly</code>	1	Schreibgeschützt
<code>vbHidden</code>	2	Versteckt
<code>vbSystem</code>	4	Systemdatei; beim Macintosh nicht verfügbar
<code>vbArchive</code>	32	Datei wurde seit dem letzten Speichern geändert

Tabelle 3.53: Einstellungen für das Argument `attributes`

► Beispiel

Im folgenden Beispiel aus Listing 3.80 wird die Datei `ARTIKEL.TXT` mit dem Attribut `Versteckt` ausgestattet.

Listing 3.80: Eine Datei mit einem Attribut über die Anweisung `SetAttr` versorgen

```
Sub SetAttr_Beispiel()
    SetAttr "c:\Artikel.txt", vbHidden
End Sub
```

► Verwandte Funktionen/Anweisungen

`FileAttr`, `GetAttr`

Shell-Funktion

Mithilfe der Funktion `Shell` können Sie ein externes Programm aus Excel heraus aufrufen.

► Syntax

```
Shell(pathname[,windowstyle])
```

Die Syntax der `Shell`-Funktion verwendet die folgenden benannten Argumente:

Teil	Beschreibung
Pathname	Erforderlich. Wert vom Typ Variant (String). Name des auszuführenden Programms sowie alle erforderlichen Argumente oder Befehlszeilen-Optionen. Auch Verzeichnis- oder Laufwerkangaben können enthalten sein.
Windowstyle	Optional. Wert vom Typ Variant (Integer), der dem Stil des Fensters entspricht, in dem das Programm ausgeführt werden soll. Wenn <code>windowstyle</code> nicht angegeben wird, erhält das Programm den Fokus und wird im minimierten Zustand gestartet.

Tabelle 3.54: Argumente der Funktion *Shell*

Die Werte des benannten Arguments `windowstyle` lauten:

Konstante	Wert	Beschreibung
<code>vbHide</code>	0	Das Fenster ist ausgeblendet, und das ausgeblendete Fenster erhält den Fokus.
<code>vbNormalFocus</code>	1	Das Fenster hat den Fokus, und die ursprüngliche Größe und Position wird wiederhergestellt.
<code>VbMinimizedFocus</code>	2	Das Fenster wird als Symbol mit Fokus angezeigt.
<code>vbMaximizedFocus</code>	3	Das Fenster wird maximiert mit Fokus angezeigt.
<code>vbNormalNoFocus</code>	4	Die zuletzt verwendete Größe und Position des Fensters wird wiederhergestellt. Das momentan aktive Fenster bleibt aktiv.
<code>vbMinimizedNoFocus</code>	6	Das Fenster wird als Symbol angezeigt. Das momentan aktive Fenster bleibt aktiv.

Tabelle 3.55: Einstellungen für das Argument *windowstyle*

► Beispiel

Im folgenden Beispiel aus Listing 3.81 wird das Programm `NOTEPAD` gestartet. Danach wird versucht, eine bestimmte Datei zu laden.

Listing 3.81: Externes Programm starten mit der Funktion Shell

```
Sub Shell_Beispiel()  
Dim x As Variant  
  
x = Shell _  
("C:\Windows\notepad.exe C:\Artikel2.txt", 3)  
End Sub
```

3.3

Dateifunktionen und Anweisungen

► Verwandte Funktionen/Anweisungen

AppActivate

Write-Anweisung

Mit der Anweisung `Write` können Sie Daten in eine sequentielle Datei schreiben.

► Syntax

`Write #Dateinummer, [Ausgabeliste]`

Die Syntax der `Write`-Anweisung besteht aus folgenden Teilen:

Teil	Beschreibung
Dateinummer	Erforderlich. Eine beliebige gültige Dateinummer.
Ausgabeliste	Optional. Eine oder mehrere (durch Kommas getrennte) numerische Ausdrücke oder Zeichenfolgenausdrücke, die in eine Datei geschrieben werden sollen.

Tabelle 3.56: Argumente der Anweisung `Write`

► Beispiel

Siehe Beispiel zur Anweisung `Reset`

► Verwandte Funktionen/Anweisungen

`Print`, `Input`

3.4 Mathematische Funktionen

Zur letzten Kategorie an Funktionen in diesem Kapitel gehören die mathematischen Funktionen.

Um vorab eine Übersicht über diese Funktionen zu bekommen, werfen Sie einen Blick auf die Tabelle 3.57. Danach erfolgen die Erklärung der Syntax und die Demonstration anhand eines kurzen Beispiels.

Funktion	Kurzbeschreibung
Abs	Gibt den Absolutwert einer Zahl zurück; das Vorzeichen wird dabei ignoriert.
Atn	Gibt den Arcustangens einer Zahl als Winkel im Bogenmaß zurück.
Cos	Gibt den Kosinus zu einem Winkel im Bogenmaß zurück.
DDB	Errechnet den Abschreibungswert bei geometrisch degressiver Abschreibung.
Exp	Gibt den Wert der Exponentialfunktion zu einer Zahl zurück.
Fix	Gibt den Vorkommawert einer Zahl zurück.
FV	Errechnet den zukünftigen Wert einer Annuität bei konstanter Zahlung, festem Zins und fester Laufzeit.
Int	Gibt den ganzzahligen Wert einer Zahl mit Dezimalen zurück.
IPmt	Errechnet den Zinsanteil für eine bestimmte Periode bei Anpassung bzw. Abzahlung bei konstanten Raten und festem Zinssatz.
IRR	Gibt den internen Ertragssatz für eine Folge regelmäßiger Ein- und Auszahlungen zurück.
Log	Liefert den Logarithmus einer Zahl.
MIRR	Liefert den modifizierten internen Ertragssatz für eine Folge regelmäßiger, mit unterschiedlichen Zinssätzen ausgestatteter Ein- und Auszahlungen.
NPer	Errechnet die Anzahl der Zeiträume für eine Annuität bei konstanter Zahlung und festem Zinssatz.
NPV	Errechnet den Nettobarwert einer Investition bei regelmäßigen Ein- und Auszahlungen und festem Diskontsatz.
Pmt	Errechnet den Auszahlungswert für eine Annuität bei festen Zeiträumen, konstanten Zahlungen und festem Zinssatz.

Tabelle 3.57: Funktionen der Kategorie Mathematik

Funktion	Kurzbeschreibung
PPmt	Errechnet den Kapitalanteil eines Zeitraums für eine Annuität bei einer festen Anzahl von Zeiträumen, konstanten Zahlungen und festem Zinssatz.
PV	Errechnet den Barwert für eine Annuität bei konstanter Zahlung, festem Zinssatz und einer festen Laufzeit.
Randomize	Initialisiert den Zufallsgenerator.
Rate	Errechnet den Zinssatz zu einer Annuität bei fester Anzahl von Zeiträumen, konstanten Zahlungen und festem Zinssatz.
Rnd	Liefert eine Zufallszahl mit Dezimalstellen zwischen 0 und 1.
Round	Rundet einen Zahlenwert auf Basis einer Dezimalstelle.
Sgn	Gibt das Vorzeichen eines Werts als Faktor (-10 oder 1) zurück.
Sin	Gibt den Sinus zu einem Winkel im Bogenmaß zurück.
SLN	Errechnet den periodischen Abschreibungswert bei linearer Abschreibung über einen bestimmten Zeitraum.
Sqr	Errechnet die Quadratwurzel einer Zahl.
SYD	Errechnet den Abschreibungswert für eine Periode nach dem Modell der Jahressummengewichtung.
Tan	Liefert den Tangens zu einem Winkel im Bogenmaß.

Tabelle 3.57: Funktionen der Kategorie Mathematik (Forts.)

ABS-Funktion

Die Funktion **ABS** gibt den Absolutwert einer Zahl zurück. Das Vorzeichen wird dabei ignoriert

► Syntax

Abs(Zahl)

Das erforderliche Argument **Zahl** kann ein beliebiger zulässiger numerischer Ausdruck sein. Wenn **Zahl** den Wert Null enthält, wird Null zurückgegeben. Wenn die Variable nicht initialisiert ist, wird der Wert Null zurückgegeben.

► Beispiel

Im folgenden Beispiel aus Listing 3.82 wird der Absolutwert einiger Zahlen im Direktfenster der Entwicklungsumgebung ausgegeben.

Listing 3.82: Den Absolutwert einer Zahl über die Funktion ABS ermitteln

```
Sub ABS_Beispiel()  
    Debug.Print Abs(12.59)  
    Debug.Print Abs(-12.59)  
End Sub
```

In beiden Fällen lautet das Ergebnis 12,59.

► Verwandte Funktionen

Sgn

Atn-Funktion

Die Funktion **Atn** gibt den Arcustangens einer Zahl als Winkel im Bogenmaß zurück.

► Syntax

Atn(Zahl)

Das erforderliche Argument **Zahl** ist ein Wert vom Typ **Double** oder ein anderer zulässiger numerischer Ausdruck.

► Beispiel

Im folgenden Beispiel aus Listing 3.83 wird der Wert **Pi** errechnet.

Listing 3.83: Die Funktion Atn einsetzen, um Pi zu errechnen

```
Sub Atn_Beispiel()  
    Dim pi As Single  
  
    pi = 4 * Atn(1)  
    MsgBox pi  
End Sub
```

► Verwandte Funktionen

Cos, Sin, Tan

Cos-Funktion

Die Funktion `Cos` gibt den Kosinus zu einem Winkel im Bogenmaß zurück.

► Syntax

`Cos(Zahl)`

Das erforderliche Argument `Zahl` ist ein Wert vom Typ `Double` oder ein beliebiger zulässiger numerischer Ausdruck, der den Winkel im Bogenmaß angibt.

► Beispiel

Im folgenden Beispiel aus Listing 3.84 wird der Kosinus eines Winkels errechnet.

Listing 3.84: Über die Funktion Cos den Kosinus für einen Winkel errechnen

```
Sub Cos_Beiispiel()  
Dim Winkel As Single  
Dim Sekans As Single  
  
Winkel = 1.5  
Sekans = 1 / Cos(Winkel)  
MsgBox Sekans  
End Sub
```

► Verwandte Funktionen

`Atn`, `Sin`, `Tan`

DDB-Funktion

Die Funktion `DDB` errechnet den Abschreibungswert bei geometrisch degressiver Abschreibung.

► Syntax

`DDB(cost, salvage, life, period[, factor])`

Die DDB-Funktion hat die folgenden benannten Argumente:

Teil	Beschreibung
Cost	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Anschaffungskosten des Vermögenswertes angibt.
Salvage	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Wert des Vermögenswertes am Ende seiner Nutzungsdauer angibt.
Life	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Länge der Nutzungsdauer des Vermögenswertes angibt.
Period	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zeitraum angibt, für den die Abschreibung des Vermögenswertes berechnet wird.
Factor	Optional. Ein Wert vom Typ <code>Variant</code> , der den Faktor angibt, um den der Wert vermindert wird. Wird der Wert nicht angegeben, so wird 2 (geometrisch degressive Methode) angenommen.

Tabelle 3.58: Argumente der Funktion DDB

► Beispiel

Im folgenden Beispiel aus Listing 3.85 wird eine Maschine nach der geometrisch degressiven Abschreibungsmethode abgeschrieben.

Listing 3.85: Degressive Abschreibungen über die Funktion DDB abwickeln

```
Sub DDB_Beispiel()
    Dim Wert As Currency
    Dim Endwert As Currency
    Dim Dauer As Integer
    Dim iPeriode As Integer

    Dauer = 5
    Wert = 2500
    Endwert = 1
    For iPeriode = 1 To Dauer - 1
        Debug.Print DDB(Wert, Endwert, Dauer, iPeriode)
    Next iPeriode
End Sub
```

► Verwandte Funktionen

FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, PV, Rate, SLN, SYG

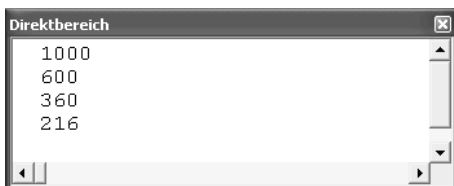


Abbildung 3.34: Abschreibungsraten

Exp-Funktion

Die Funktion **Exp** gibt den Wert der Exponentialfunktion zu einer Zahl zurück.

► Syntax

Exp(Zahl)

Das erforderliche Argument **Zahl** ist ein Wert vom Typ **Double** oder ein beliebiger zulässiger numerischer Ausdruck.

► Beispiel

Im folgenden Beispiel aus Listing 3.86 wird die Exponentialfunktion für die Zahl 1 errechnet.

*Listing 3.86: Exponentialfunktion für eine Zahl über die Funktion **Exp** errechnen*

```
Sub Exp_Beispiel()  
    Debug.Print Exp(1)  
End Sub
```

► Verwandte Funktionen

Log

Fix-Funktion

Die Funktion **Fix** gibt den Vorkommawert einer Zahl zurück.

► Syntax

Fix(Zahl)

Das erforderliche Argument `Zahl` ist ein Wert vom Typ `Double` oder ein beliebiger zulässiger numerischer Ausdruck. Wenn `Zahl` den Wert Null enthält, wird Null zurückgegeben.

► Beispiel

Im folgenden Beispiel aus Listing 3.87 wird der ganzzahlige Wert einiger Zahlen ermittelt.

Listing 3.87: Ganzzahligen Wert über die Funktion `Fix` ermitteln

```
Sub Fix_Beiispiel()  
    Debug.Print Fix(10.1)  
    Debug.Print Fix(-10.7)  
    Debug.Print Fix(-10.3)  
End Sub
```



Abbildung 3.35: Das Ergebnis im Direktfenster

► Verwandte Funktionen

`Int`, `Round`

FV-Funktion

Die Funktion `FV` errechnet den zukünftigen Wert einer Annuität bei konstanter Zahlung, festem Zins und fester Laufzeit.

► Syntax

`FV(rate, nper, pmt[, pv[, type]])`

Die FV-Funktion hat die folgenden benannten Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zinssatz pro Zeitraum angibt. Wenn Sie beispielsweise einen Kredit mit einem Jahreszins von 10 Prozent aufnehmen und monatliche Zahlungen vereinbart haben, beträgt der Zinssatz pro Zeitraum 0,1 dividiert durch 12 oder 0,0083.
Nper	Erforderlich. Ein Wert vom Typ <code>Integer</code> , der die Gesamtanzahl der Zahlungszeiträume für die Annuität angibt. Wenn Sie beispielsweise monatliche Zahlungen für einen Autokredit mit 4 Jahren Laufzeit vereinbart haben, beträgt die Summe der Zahlungszeiträume für Ihren Kredit $4 * 12$ (oder 48).
Pmt	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Zahlung pro Zeitraum angibt. Die Zahlungen enthalten gewöhnlich Kapital und Zinsen und ändern sich während der Laufzeit einer Annuität nicht.
Pv	Optional. Ein Wert vom Typ <code>Variant</code> , der den Barwert (oder Gesamtbetrag) einer Folge zukünftiger Zahlungen zum jetzigen Zeitpunkt angibt. Wenn Sie beispielsweise Geld aufnehmen, stellt die Kredithöhe für den Kreditgeber den Barwert der von Ihnen zu leistenden monatlichen Zahlungen dar. Wird der Barwert nicht angegeben, so wird 0 angenommen.
Type	Optional. Ein Wert vom Typ <code>Variant</code> , der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende des Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.

Tabelle 3.59: Argumente der Funktion FV

► Beispiel

Im folgenden Beispiel wird errechnet, wie sich ein angelegtes Kapital von 10.000 Euro über einen Zeitraum von 12 Jahren bei einem festen Zinssatz von 4,5% im Jahr bei einer monatlichen Einzahlung von 100 Euro entwickelt.

Listing 3.88: Den zukünftigen Wert einer Annuität über die Funktion FV errechnen

```
Sub FV_Beispiel()  
Dim Wert As Currency  
Dim Dauer As Integer  
Dim Rate As Currency
```

```
Dim Zins As Single
```

```
Wert = 10000
```

```
Dauer = 12 * 12
```

```
Rate = 100
```

```
Zins = 0.04 / 12
```

```
Debug.Print FV(Zins, Rate, Dauer, Wert)
```

```
End Sub
```

► Verwandte Funktionen

DDB, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, PV, Rate, SLN, SYG

Int-Funktion

Die Funktion `Int` gibt den Vorkommawert einer Zahl zurück.

► Syntax

```
Int(Zahl)
```

Das erforderliche Argument `Zahl` ist ein Wert vom Typ `Double` oder ein beliebiger zulässiger numerischer Ausdruck. Wenn `Zahl` den Wert Null enthält, wird Null zurückgegeben.

► Beispiel

Im folgenden Beispiel aus Listing 3.89 wird der ganzzahlige Wert einiger Zahlen ermittelt.

Listing 3.89: Den ganzzahligen Wert einer Dezimalzahl über die Funktion `Int` ermitteln

```
Sub Int_Beispiel()  
    Debug.Print Int(10.1)  
    Debug.Print Int(-10.7)  
    Debug.Print Int(-10.3)  
End Sub
```



Abbildung 3.36: Vergleiche Ergebnis mit Abbildung 3.35

Hinweis

Der Unterschied zwischen `Int` und `Fix` besteht darin, dass bei einem negativen Wert von `Zahl` `Int` die erste negative ganze Zahl zurückgibt, die kleiner oder gleich `Zahl` ist, während `Fix` die erste negative ganze Zahl zurückgibt, die größer oder gleich `Zahl` ist.

► Verwandte Funktionen

`Fix`, `Round`

IPmt-Funktion

Die Funktion `IPmt` errechnet den Zinsanteil für eine bestimmte Periode bei Ansparung bzw. Abzahlung bei konstanten Raten und festem Zinssatz.

► Syntax

`IPmt(rate, per, nper, pv[, fv[, type]])`

Die `IPmt`-Funktion hat die folgenden benannten Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zinssatz pro Zeitraum angibt.
Per	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zahlungszeitraum im Bereich von 1 bis <code>nper</code> angibt.
Nper	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Gesamtanzahl der Zahlungszeiträume für die Annuität angibt.
Pv	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Barwert oder heutigen Wert einer Folge zukünftiger Aus- oder Einzahlungen angibt.

Tabelle 3.60: Argumente der Funktion `IPmt`

Teil	Beschreibung
Fv	Optional. Ein Wert vom Typ <code>Variant</code> , der den Endwert oder Kontostand angibt, der nach der letzten Zahlung erreicht sein soll. Der Endwert eines Kredits ist zum Beispiel 0 Euro, da dies die Kredithöhe nach der letzten Zahlung ist.
Type	Optional. Ein Wert vom Typ <code>Variant</code> , der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende eines Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.

Tabelle 3.60: Argumente der Funktion *IPmt* (Forts.)

► Verwandte Funktionen

DDB, FV, IRR, MIRR, NPer, NPV, Pmt, PV, Rate, SLN, SYG

IRR-Funktion

Die Funktion **IRR** gibt den internen Ertragssatz für eine Folge regelmäßiger Ein- und Auszahlungen zurück.

► Syntax

`IRR(values()[, guess])`

Die **IRR**-Funktion hat die folgenden benannten Argumente:

Teil	Beschreibung
<code>values()</code>	Erforderlich. Ein Datenfeld mit Werten des Typs <code>Double</code> , der Cash Flow-Werte angibt. Dieses Datenfeld muss mindestens einen negativen Wert (Zahlungsausgang) und einen positiven Wert (Zahlungseingang) enthalten.
<code>Guess</code>	Optional. Ein Wert vom Typ <code>Variant</code> , der einen von Ihnen geschätzten Wert enthält, der von IRR zurückgegeben wird. Wird der Wert nicht angegeben, so ist <code>guess</code> gleich 0,1 (10 Prozent).

Tabelle 3.61: Argumente der Funktion *IRR*

► Beispiel

Im folgenden Beispiel aus Listing 3.90 wird der interne Zinsfuß anhand dreier vorliegender Cash-Flows ermittelt.

Listing 3.90: Den internen Zinsfuß über die Funktion IRR ausrechnen

```
Sub IRR_Beispiel()  
Dim Schätz as Double  
Dim IntZins as double  
Static Werte(3) As Double  
  
Schätz = 0.1  
Werte(0) = -10000  
Werte(1) = 12000: Werte(2) = 18000  
  
IntZins = IRR(Werte(), Schätz) * 100  
MsgBox "Zinsfuß für die Cash Flows beträgt " & _  
Format(IntZins, "0,00 %")  
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PV, Rate, SLN, SYG

Log-Funktion

Die Funktion Log liefert den Logarithmus einer Zahl.

► Syntax

Log(Zahl)

Das erforderliche Argument *Zahl* ist ein Wert vom Typ *Double* oder ein beliebiger zulässiger numerischer Ausdruck mit einem Wert größer als Null.

► Verwandte Funktionen

Exp

MIRR-Funktion

Die Funktion MIRR liefert den modifizierten internen Ertragssatz für eine Folge regelmäßiger, mit unterschiedlichen Zinssätzen ausstatteter Ein- und Auszahlungen.

► Syntax

MIRR(values(), finance_rate, reinvest_rate)

Die MIRR-Funktion hat die folgenden benannten Argumente:

Teil	Beschreibung
values()	Erforderlich. Ein Datenfeld mit Werten des Typs <code>Double</code> , die Cash Flow-Werte angeben. Das Datenfeld muss mindestens einen negativen Wert (Zahlungsausgang) und einen positiven Wert (Zahlungseingang) enthalten.
finance_rate	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zinssatz enthält, der bei der Finanzierung einer Anlage bezahlt werden muss.
reinvest_rate	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zinssatz angibt, der bei erneuter Anlage von Kapital erzielt werden kann.

Tabelle 3.62: Argumente der Funktion MIRR

► Beispiel

Im folgenden Beispiel aus Listing 3.91 wird der modifizierte interne Zinsfuß bei einer Folge von Cash-Flows ausgerechnet, die im Datenfeld `Werte()` gespeichert sind. `KreditZins` stellt den Finanzierungszinssatz dar und `InvZins` den Zinssatz, der bei erneuter Investition erzielt werden kann.

Listing 3.91: Den modifizierten internen Ertragssatz über die Funktion MIRR errechnen

```
Sub MIRR_Beispiel()
Dim KreditZins As Double
Dim InvZins As Double
Dim IntZins As Double
Static Werte(5) As Double

KreditZins = 0.1
InvZins = 0.1
Werte(0) = -50000
Werte(1) = 12000: Werte(2) = 18000
Werte(3) = 23000: Werte(4) = 29000
```

```
IntZins = MIRR(Werte(), KreditZins, InvZins)
MsgBox _
"Der modifizierte interne Zinsfuß lautet: " & _
Format(Abs(IntZins) * 100, "0,00 %")
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, NPer, NPV, Pmt, PPmt, PV, Rate, SLN, SYG

NPer-Funktion

Die Funktion **NPer** errechnet die Anzahl der Zeiträume für eine Annuität bei konstanter Zahlung und festem Zinssatz.

► Syntax

NPer(rate, pmt, pv[, fv[, type]])

Die **NPer**-Funktion hat folgende benannten Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ <i>Double</i> , der den Zinssatz pro Zeitraum angibt.
Pmt	Erforderlich. Ein Wert vom Typ <i>Double</i> , der die Zahlung pro Zeitraum angibt. Die Zahlungen enthalten gewöhnlich Kapital und Zinsen und ändern sich während der Laufzeit einer Annuität nicht.
Pv	Erforderlich. Ein Wert vom Typ <i>Double</i> , der den Barwert oder heutigen Wert einer Folge zukünftiger Aus- oder Einzahlungen angibt.
Fv	Optional. Ein Wert vom Typ <i>Variant</i> , der den Endwert oder Kontostand angibt, der nach der letzten Zahlung erreicht sein soll.
Type	Optional. Ein Wert vom Typ <i>Variant</i> , der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende eines Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.

Tabelle 3.63: Argumente der Funktion NPer

► Beispiel

Beim folgenden Beispiel aus Listing 3.92 soll ein Kredit von 25.000 Euro aufgenommen werden. Bekannt sind die Kredithöhe, der dafür zu zahlende Zinssatz (5%) und die monatlichen Zahlungen (250 Euro), um den Kredit abzuzahlen. In wie vielen Monaten ist der Kredit dann zurückgezahlt?

Listing 3.92: Die Laufzeit eines Kredits über die Funktion NPer ermitteln

```
Sub NPer_Beispiel()  
Dim EndWert As Currency  
Dim AnfangWert As Currency  
Dim Zins As Single  
Dim Zahlung As Currency  
Dim ZahlTyp As Integer  
Dim Monat As Integer  
  
EndWert = 0  
AnfangWert = 25000  
Zins = 5  
Zins = Zins / 100  
Zahlung = 250  
ZahlTyp = 0  
  
Monat = NPer(Zins / 12, -Zahlung, AnfangWert, _  
EndWert, ZahlTyp)  
MsgBox "Ihr Kredit ist nach " & Monat & _  
" Monaten abgezahlt."  
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPV, Pmt, PPmt, PV, Rate, SLN, SYG

NPV-Funktion

Die Funktion **NPV** errechnet den Nettobarwert einer Investition bei regelmäßigen Ein- und Auszahlungen und festem Diskontsatz.

► Syntax

NPV(rate, values())

Die NPV-Funktion hat folgende benannte Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Diskontsatz bezogen auf die Länge des Zeitraums (ausgedrückt als Dezimalzahl) enthält.
values()	Erforderlich. Ein Datenfeld mit Werten des Typs <code>Double</code> , die Cash Flow-Werte enthalten. Das Datenfeld muss mindestens einen negativen Wert (Auszahlung) und einen positiven Wert (Einzahlung) enthalten.

Tabelle 3.64: Argumente der Funktion NPV

► Beispiel

Beim folgenden Beispiel aus Listing 3.93 wird der Nettobarwert von einer Folge von Cash-Flows ermittelt. Die Werte liegen im Datenfeld `Werte()` vor. Die Variable `IntZins` ist ein fester interner Zinsfuß.

Listing 3.93: Den Nettobarwert über die Funktion NPV ermitteln

```
Sub NPV_Beispiel()
    Dim IntZins As Single
    Dim NB As Single
    Static Werte(5) As Double

    IntZins = 0.05
    Werte(0) = -50000
    Werte(1) = 12000: Werte(2) = 18000
    Werte(3) = 23000: Werte(4) = 29000
    NB = NPV(IntZins, Werte())

    MsgBox "Der Netto-Barwert lautet " & _
        Format(NB, "#,##0,00")
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, Pmt, PPmt, PV, Rate, SLN, SYG

Pmt-Funktion

Die Funktion `Pmt` errechnet den Auszahlungswert für eine Annuität bei festen Zeiträumen, konstanten Zahlungen und festem Zinssatz.

► Syntax

`Pmt(rate, nper, pv[, fv[, type]])`

Die `Pmt`-Funktion hat die folgenden benannten Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Zinssatz pro Zeitraum angibt.
Nper	Erforderlich. Ein Wert vom Typ <code>Integer</code> , der die Gesamtanzahl der Zahlungszeiträume für die Annuität angibt.
Pv	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Barwert oder heutigen Wert einer Folge zukünftiger Aus- oder Einzahlungen angibt.
Fv	Optional. Ein Wert vom Typ <code>Variant</code> , der den Endwert oder Kontostand angibt, der nach der letzten Zahlung erreicht sein soll.
Type	Optional. Ein Wert vom Typ <code>Variant</code> , der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende eines Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.

Tabelle 3.65: Argumente der Funktion `Pmt`

► Beispiel

Im folgenden Beispiel aus Listing 3.94 wird ein Kredit von € 25.000 zu einem Zinssatz von 5% aufgenommen. Die Laufzeit ist mit 36 Monaten vorgegeben. Wie hoch ist der monatlich zu zahlende Betrag?

Listing 3.94: Über die Funktion Pmt die Monatsrate für einen Kredit errechnen

```

Sub Pmt_Beispiel()
Dim EndWert As Currency
Dim anfangWert As Currency
Dim ZINS As Single
Dim Monat As Integer
Dim ZahlTyp As Integer
Dim Zahlung As Currency

EndWert = 0
anfangWert = 25000
ZINS = 5 / 100
Monat = 36
ZahlTyp = 0
Zahlung = Pmt(ZINS / 12, Monat, -anfangWert, _
               EndWert, ZahlTyp)
MsgBox "Sie zahlen " & Format(Zahlung, _
                              "#,##0.00") & " im Monat."

End Sub

```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, PPmt, PV, Rate, SLN, SYG

PPmt-Funktion

Die Funktion **PPmt** errechnet den Kapitalanteil eines Zeitraums für eine Annuität bei einer festen Anzahl von Zeiträumen, konstanten Zahlungen und festem Zinssatz.

► Syntax

PPmt(rate, per, nper, pv[, fv[, type]])

Die PPmt-Funktion hat folgende benannte Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ Double, der den Zinssatz pro Zeitraum angibt.
Per	Erforderlich. Ein Wert vom Typ Integer, der den Zahlungszeitraum im Bereich von 1 bis nper angibt.
Nper	Erforderlich. Ein Wert vom Typ Integer, der die Gesamtanzahl der Zahlungszeiträume für die Annuität angibt.
Pv	Erforderlich. Ein Wert vom Typ Double, der den Barwert oder heutigen Wert einer Folge zukünftiger Aus- oder Einzahlungen angibt.
Fv	Optional. Ein Wert vom Typ Variant, der den Endwert oder Kontostand angibt, der nach der letzten Zahlung erreicht sein soll.
Type	Optional. Ein Wert vom Typ Variant, der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende eines Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.

Tabelle 3.66: Argumente der Funktion PPmt

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PV, Rate, SLN, SYG

PV-Funktion

Die Funktion PV errechnet den Barwert für eine Annuität bei konstanter Zahlung, festem Zinssatz und einer festen Laufzeit.

► Syntax

PV(rate, nper, pmt[, fv[, type]])

Die PV-Funktion hat folgende benannte Argumente:

Teil	Beschreibung
Rate	Erforderlich. Ein Wert vom Typ Double, der den Zinssatz pro Zeitraum angibt.
Nper	Erforderlich. Ein Wert vom Typ Integer, der die Gesamtanzahl der Zahlungszeiträume für die Annuität angibt.

Tabelle 3.67: Argumente der Funktion PV

Teil	Beschreibung
Pmt	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Zahlung pro Zeitraum angibt. Die Zahlungen enthalten gewöhnlich Kapital und Zinsen und ändern sich während der Laufzeit einer Annuität nicht.
Fv	Optional. Ein Wert vom Typ <code>Variant</code> , der den Endwert oder Kontostand angibt, der nach der letzten Zahlung erreicht sein soll.
Type	Optional. Ein Wert vom Typ <code>Variant</code> , der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende eines Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.

Tabelle 3.67: Argumente der Funktion PV (Forts.)

► Beispiel

Beim folgenden Beispiel aus Listing 3.95 soll am Ende einer Laufzeit ein bestimmter Betrag von 850.000 Euro verfügbar sein. Dazu kann jedes Jahr über einen bestimmten Zeitraum hinweg (10 Jahre) ein bestimmter Betrag (65.000 Euro) eingezahlt und verzinst werden. Wie hoch muss nun der Ausgangswert sein, um das Ziel zu erreichen?

Listing 3.95: Über die Funktion PV den Barwert für eine Annuität ausrechnen

```
Sub Pv_Beispiel()
Dim ZINS As Single
Dim ZahlungJahr As Integer
Dim EinkommenJahr As Currency
Dim EndWert As Currency
Dim AnfangWert As Currency
Dim Zahltyp As Integer

ZINS = 0.045
ZahlungJahr = 10
EinkommenJahr = 65000
EndWert = 850000

Zahltyp = 0
AnfangWert = PV(ZINS, ZahlungJahr, -EinkommenJahr, _
                EndWert, Zahltyp)
```

```
MsgBox "Der Startwert lautet " & _
Format(AnfangWert, "#,##0.00")
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, Rate, SLN, SYG

Randomize-Anweisung

Die Anweisung `Randomize` initialisiert den Zufallsgenerator.

► Syntax

`Randomize [Zahl]`

Das optionale Argument `Zahl` ist ein Wert vom Typ `Variant` oder ein beliebiger zulässiger numerischer Ausdruck.

► Beispiel

Siehe Funktion `Rnd`

Rate-Funktion

Die Funktion `Rate` errechnet den Zinssatz zu einer Annuität bei fester Anzahl von Zeiträumen, konstanten Zahlungen und festem Zinssatz.

► Syntax

`Rate(nper, pmt, pv[, fv[, type[, guess]])`

Die `Rate`-Funktion hat folgende benannte Argumente:

Teil	Beschreibung
Nper	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Gesamtanzahl der Zahlungszeiträume für die Annuität angibt.
Pmt	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Zahlung pro Zeitraum angibt. Die Zahlungen enthalten gewöhnlich Kapital und Zinsen und ändern sich während der Laufzeit einer Annuität nicht.
Pv	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Barwert oder heutigen Wert einer Folge zukünftiger Aus- oder Einzahlungen angibt.

Tabelle 3.68: Argument der Funktion `Rate`

Teil	Beschreibung
Fv	Optional. Ein Wert vom Typ Variant, der den Endwert oder Kontostand angibt, der nach der letzten Zahlung erreicht sein soll.
Type	Optional. Ein Wert vom Typ Variant, der angibt, wann Zahlungen fällig sind. Bei 0 sind die Zahlungen am Ende eines Zahlungszeitraums fällig, bei 1 zu Beginn des Zahlungszeitraums. Wird der Wert nicht angegeben, so wird 0 angenommen.
Guess	Optional. Ein Wert vom Typ Variant, der einen von Ihnen geschätzten Wert enthält, der von Rate zurückgegeben wird. Wird der Wert nicht angegeben, so ist guess gleich 0,1 (10 Prozent).

Tabelle 3.68: Argument der Funktion Rate (Forts.)

► Beispiel

Im folgenden Beispiel aus Listing 3.96 wird ein Kredit von 150.000 Euro aufgenommen. Der Kredit soll nach 12 Jahren zurückbezahlt sein. Monatlich werden jeweils 650 Euro abbezahlt. Wie hoch ist nun der Zinssatz?

Listing 3.96: Den Zinssatz für einen Kredit über die Funktion Rate ausrechnen

```
Sub Rate_Beispiel()
Dim EndWert As Currency
Dim Anfangwert As Currency
Dim Betrag As Currency
Dim Monate As Integer
Dim ZahlTyp As Integer
Dim ZINS As Single

EndWert = 0
Anfangwert = 150000
Betrag = 650
Monate = 12 * 12
ZahlTyp = 0
ZINS = (Rate(Monate, -Betrag, Anfangwert, EndWert, _
    ZahlTyp) * 12) * 100
MsgBox "Ihr Zinssatz beträgt " & _
    Format(ZINS, "#,##0.00")
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, PV, SLN, SYG

Rnd-Funktion

Die Funktion `Rnd` liefert eine Zufallszahl mit Dezimalstellen zwischen 0 und 1.

► Syntax

`Rnd[(Zahl)]`

Das optionale Argument `Zahl` ist ein Wert vom Typ `Single` oder ein beliebiger zulässiger numerischer Ausdruck.

► Beispiel

Beim folgenden Beispiel aus Listing 3.97 werden mithilfe der Funktion `Rnd` Zufallsbuchstaben gebildet.

Listing 3.97: Erzeugen von Zufallsbuchstaben über die Funktionen `Rnd` und `Chr`

```
Sub Rnd_Beiispiel()  
Dim i As Integer  
  
Randomize  
For i = 1 To 10  
    Debug.Print (Chr(Int((122 - 97 + 1) * Rnd + 97)))  
Next i  
End Sub
```



Abbildung 3.37: Buchstaben über den Zufallsgenerator erzeugen

Round-Funktion

Mithilfe der Funktion `Round` runden Sie einen Zahlenwert auf Basis einer angegebenen Dezimalstelle.

► Syntax

`Round(Ausdruck [,AnzahlAnDezimalpunktn])`

Die Syntax der `Round`-Funktion besteht aus folgenden Teilen:

Teil	Beschreibung
Ausdruck	Erforderlich. Numerischer Ausdruck, der gerundet wird.
AnzahlAnDezimalpunktn	Optional. Zahl, die angibt, wie viele Stellen rechts vom Dezimalpunkt beim Runden berücksichtigt werden. Wird dieser Wert ausgelassen, gibt die <code>Round</code> -Funktion Ganzzahlen zurück.

Tabelle 3.69: Argumente der Funktion `Round`

► Beispiel

Im folgenden Beispiel aus Listing 3.98 werden einige Zahlen auf verschiedene Art und Weise gerundet.

Listing 3.98: Zahlen runden über die Funktion Round

```

Sub Round_Beiispiel()
    Debug.Print Application.Round(12.456, 0)
    Debug.Print Application.Round(12.456, 1)
    Debug.Print Application.Round(12.456, 2)
    Debug.Print Application.Round(12.456, 3)
End Sub

```

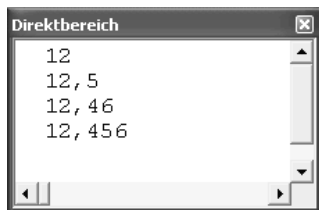


Abbildung 3.38: Die Ergebnisse des Rundens

► **Verwandte Funktionen**

Int, Fix

Sgn-Funktion

Die Funktion **Sgn** gibt das Vorzeichen eines Werts als Faktor (-10 oder 1) zurück.

► **Syntax**

Sgn(Zahl)

Das erforderliche Argument **Zahl** kann ein beliebiger numerischer Ausdruck sein.

Wert von Zahl	Rückgabewert von Sgn
Größer als Null	1
Gleich Null	0
Kleiner als Null	-1

Tabelle 3.70: Rückgabewerte der Funktion Sgn

► Beispiel

Im folgenden Beispiel aus Listing 3.99 werden einige Zahlen nach dem Vorzeichen ausgewertet.

Listing 3.99: Über die Funktion Sgn die Vorzeichen einer Zahl ermitteln

```
Sub Sgn_Beiispiel()  
    Debug.Print Sgn(-120)  
    Debug.Print Sgn(120)  
    Debug.Print Sgn(0)  
End Sub
```

► Verwandte Funktionen

ABS, Left

Sin-Funktion

Die Funktion Sin gibt den Sinus zu einem Winkel im Bogenmaß zurück.

► Syntax

Sin(Zahl)

Das erforderliche Argument Zahl ist ein Wert vom Typ Double oder ein beliebiger zulässiger numerischer Ausdruck, der einen Winkel im Bogenmaß ausdrückt.

► Beispiel

Siehe Beispiel Cos

► Verwandte Funktionen

Cos, Tan, Atn

SLN-Funktion

Die Funktion `SLN` errechnet den periodischen Abschreibungswert bei linearer Abschreibung über einen bestimmten Zeitraum.

► Syntax

`SLN(cost, salvage, life)`

Die `SLN`-Funktion hat folgende benannte Argumente:

Teil	Beschreibung
Cost	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Anschaffungskosten des Vermögenswerts angibt.
Salvage	Erforderlich. Ein Wert vom Typ <code>Double</code> , der den Vermögenswert am Ende seiner Nutzungsdauer angibt.
Life	Erforderlich. Ein Wert vom Typ <code>Double</code> , der die Länge der Nutzungsdauer des Vermögenswerts angibt.

Tabelle 3.71: Argumente der Funktion `SLN`

► Beispiel

Im folgenden Beispiel aus Listing 3.100 wird eine Maschine nach der linearen Abschreibungsmethode abgeschrieben.

Listing 3.100: Lineare Abschreibung über die Funktion `SLN` durchführen

```
Sub SLN_Beispiel()
    Dim Wert As Currency
    Dim Endwert As Currency
    Dim Dauer As Integer
    Dim i As Integer

    Dauer = 5
    Wert = 2500
    Endwert = 0
    For i = 1 To Dauer - 1
        Debug.Print SLN(Wert, Endwert, Dauer)
    Next i
End Sub
```

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, PV, Rate, SYG

Sqr-Funktion

Die Funktion `Sqr` errechnet die Quadratwurzel einer Zahl.

► Syntax

`Sqr(Zahl)`

Das erforderliche Argument `Zahl` ist ein Wert vom Typ `Double` oder ein beliebiger zulässiger numerischer Ausdruck, der größer oder gleich Null ist.

► Beispiel

Im folgenden Beispiel aus Listing 3.101 wird die Wurzel aus ein paar Zahlenwerten gezogen.

Listing 3.101: Wurzelziehen über die Funktion `Sqr`

```
Sub Sqr_Beispiel()  
    Debug.Print Sqr(16)  
    Debug.Print Sqr(64)  
    Debug.Print Sqr(120)  
End Sub
```

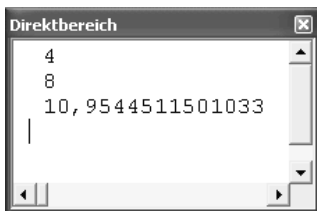


Abbildung 3.39: Die Ergebnisse aus der Wurzelrechnung

SYD-Funktion

Die Funktion SYD errechnet den Abschreibungswert für eine Periode nach dem Modell der Jahressummengewichtung.

► Syntax

SYD(cost, salvage, life, period)

Die SYD-Funktion hat folgende benannte Argumente:

Teil	Beschreibung
Cost	Erforderlich. Ein Wert vom Typ Double, der die Anschaffungskosten des Vermögenswerts angibt.
Salvage	Erforderlich. Ein Wert vom Typ Double, der den Vermögenswert am Ende seiner Nutzungsdauer angibt.
Life	Erforderlich. Ein Wert vom Typ Double, der die Länge der Nutzungsdauer des Vermögenswerts angibt.
Period	Erforderlich. Ein Wert vom Typ Double, der den Zeitraum angibt, für den die Abschreibung des Vermögenswerts berechnet wird.

Tabelle 3.72: Argumente der Funktion SYD

► Verwandte Funktionen

DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, PV, Rate, SLN

Tan-Funktion

Die Funktion Tan liefert den Tangens zu einem Winkel im Bogenmaß.

► Syntax

Tan(Zahl)

Das erforderliche Argument Zahl ist ein Wert vom Typ Double oder ein beliebiger zulässiger numerischer Ausdruck, der einen Winkel im Bogenmaß ausdrückt.

► Beispiel

Siehe Beispiel Cos

► Verwandte Funktionen

Atn, Cos, Sin