Bernd Held

Das Access-VBA Codebook

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Allgemeine VBA-Funktionen

In VBA gibt es Hunderte von Funktionen. Die meisten davon können im kompletten Office-Paket eingesetzt werden, andere sind Access-spezifisch. Funktionen lassen sich in verschiedene Gruppen einteilen. So gibt es beispielsweise Text-, Datums- und Zeitfunktionen, mathematische Funktionen und vieles mehr. Die folgenden Funktionen werden in diesem Kapitel vorgestellt:

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

1 Zeichenfolge in gültiges Datum umwandeln

Mithilfe der Funktion `CDate` können Zeichenfolgen in einen gültigen Datumswert konvertiert werden. Im folgenden Beispiel aus Listing 1 wird eine Zeichenfolge in ein Datumsformat umgewandelt.

```
'=====================================================
' Auf CD     Buchdaten\Beispiele\Kap01
' Dateiname  Funktionen.mdb
' Modul      MdlDate
'=====================================================

Sub DatumWandeln()
   Dim strDatum As String
   strDatum = "17. Mai 2004"
   strDatum = CDate(strDatum)
End Sub
```

Listing 1: Mit der Funktion CDate Zeichenfolgen in gültige Datumswerte umwandeln

Das Datum liegt in einer Variablen vom Typ `String` vor. Übergeben Sie diesen String der Funktion `CDate`, die daraus ein verwertbares Datum macht. Die Funktion `CDate` erkennt sowohl Datumsliterale und Zeitliterale als auch bestimmte Zahlen, die im zulässigen Bereich für ein Datum liegen. Beim Umwandeln einer Zahl in ein Datumsformat wird der ganzzahlige Teil für das Datum verwendet. Nachkommastellen der Zahl werden in eine Zeitangabe (beginnend bei 0:00) umgewandelt.

Selbstverständlich funktioniert dasselbe auch mit Zeitwerten:

```
'=================================================
' Auf CD     Buchdaten\Beispiele\Kap01
' Dateiname  Funktionen.mdb
' Modul      MdlDate
'=================================================

Sub ZeitWandeln()
   Dim strZeit As String
   strZeit = "18:45:00"
   strZeit = CDate(strZeit)
End Sub
```

Listing 2: Mit der Funktion CDate Zeichenfolgen in gültige Zeitwerte umwandeln

2 Systemdatum abrufen

Zugriff auf das Systemdatum, welches in Windows hinterlegt ist, erhalten Sie über die Funktion `Date`. Die Funktion `Time` gibt die aktuelle Systemzeit aus.

Ein typisches Beispiel für den Gebrauch dieser Funktionen ist das Anzeigen des aktuellen Tagesdatums sowie der Uhrzeit auf dem Bildschirm.

```
'=================================================
' Auf CD     Buchdaten\Beispiele\Kap01
' Dateiname  Funktionen.mdb
' Modul      MdlDate
'=================================================

Sub SystemdatumAbrufen()
   MsgBox "Heute ist der " & Date & vbLf & "um " & Time & " Uhr"
End Sub
```

Listing 3: Das aktuelle Tagesdatum anzeigen

Abbildung 1: Das aktuelle Datum sowie die Uhrzeit ermitteln

Hinweis

Sollten das Datum bzw. die Uhrzeit nicht stimmen, dann kontrollieren Sie diese Einstellungen in der Systemsteuerung von Windows.

3 Endtermine errechnen

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

```
DateAdd(interval, number, date)
```

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

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

Tabelle 1: Die Argumente der Funktion DateAdd

Im folgenden Makro aus Listing 4 wird ausgehend von einem Bestelldatum der Liefertermin errechnet und im Direktfenster der Entwicklungsumgebung ausgegeben.

```
'=====================================================
' Auf CD     Buchdaten\Beispiele\Kap01
' Dateiname  Funktionen.mdb
' Modul      MdlDate
'=====================================================

Sub EndTerminErrechnen()
   Dim DateBestellung As Date
   Dim strIntervall As String
   Dim intZahl As Integer

   DateBestellung = "20.07.2004"
   Debug.Print "Bestelldatum: " & DateBestellung
   strIntervall = "m"
   intZahl = 3
   Debug.Print "Lieferdatum: " & _
   DateAdd(strIntervall, intZahl, DateBestellung)
End Sub
```

Listing 4: Das Enddatum mit der Funktion DateAdd ausrechnen

Über die Variable strIntervall geben Sie bekannt, in welcher Einheit Sie rechnen möchten. Die dafür in Frage kommenden Einheiten können Sie der Tabelle 2 entnehmen.

Einheit	Beschreibung
yyyy	Jahr
q	Quartal
m	Monat
y	Tag des Jahres
d	Tag
w	Wochentag
ww	Woche
h	Stunde
n	Minute
s	Sekunde

Tabelle 2: Die verfügbaren Intervalle für das Argument Interval

Über die Variable `intZahl` geben Sie den Datumsversatz bekannt, der entweder aus einer positiven (Zukunft) oder einer negativen Zahl (Vergangenheit) bestehen kann.

Als letzte Variable übergeben Sie der Funktion das Ausgangsdatum, von dem Sie die Berechnung starten möchten.

Das Ergebnis wird über die Anweisung `Debug.Print` im Direktfenster der Entwicklungsumgebung ausgegeben.

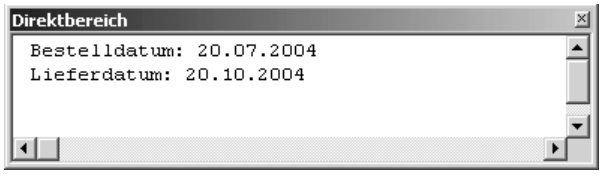

Abbildung 2: Ein Lieferdatum in der Zukunft errechnen

4 Kalenderwoche ermitteln

Mithilfe der Funktion `DatePart` können Sie einen bestimmten Teil des Datums extrahieren, indem Sie auf die Konstanten dieser Funktion zurückgreifen. Unter anderem können Sie dabei auch die Kalenderwoche zu einem Datum ermitteln.

Die Syntax der Funktion `DatePart` lautet:

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

Im Argument `Interval` können Sie angeben, welchen Teil des Datums Sie extrahieren möchten. Sehen Sie sich dazu die Tabelle 2 an.

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 ersten 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 4.

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

Tabelle 3: Die 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 4: Die FirstWeekOfYear-Konstanten der Funktion DatePart

Im nächsten Beispiel aus Listing 5 soll anhand des aktuellen Tagesdatums die dazugehörige Wochennummer ermittelt und im Direktfenster der Entwicklungsumgebung ausgegeben werden.

```
'====================================================
' Auf CD     Buchdaten\Beispiele\Kap01
' Dateiname  Funktionen.mdb
' Modul      MdlDate
'====================================================

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

Listing 5: Die Kalenderwoche ermitteln

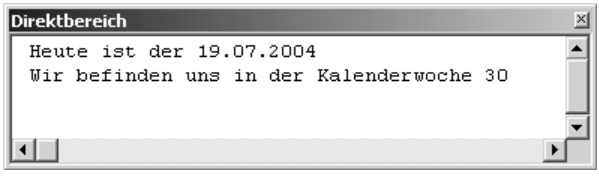

Abbildung 3: Aus einem Datum die dazugehörige KW errechnen

Möchten Sie diese Lösung über eine Funktion aufrufen, dann sehen Sie sich einmal das Listing 6 an.

```
'====================================================
' Auf CD     Buchdaten\Beispiele\Kap01
' Dateiname  Funktionen.mdb
' Modul      MdlDate
'====================================================

Function DINKw(Datum)
   DINKw = DatePart("ww", Datum, vbMonday, vbFirstFourDays)
End Function

Sub KW()
   MsgBox "Wir gefinden uns in der KW " & DINKw(Date)
End Sub
```

Listing 6: Die Kalenderwoche über eine Funktion abrufen

Übergeben Sie der Funktion `DINKw` das aktuelle Datum, welches Sie über die Funktion `Date` abrufen. In der Funktion `DINKw` wird die Funktion `DatePart` eingesetzt, die das übergebene Datum untersucht. Den Rückgabewert geben Sie über die Methode `Msgbox` am Bildschirm aus.