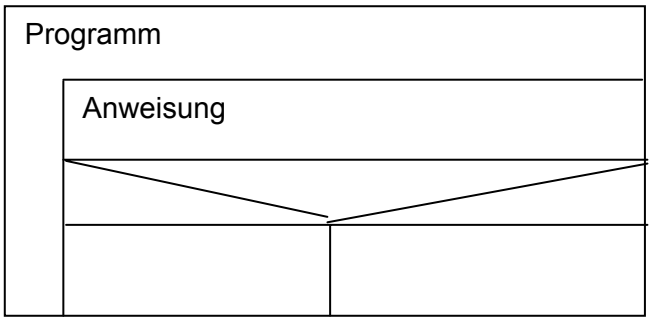
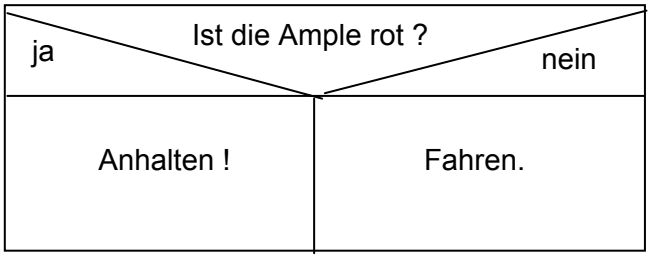
Inhaltsverzeichnis

Programmablaufplan-Technik nach Nassi Schneidermann

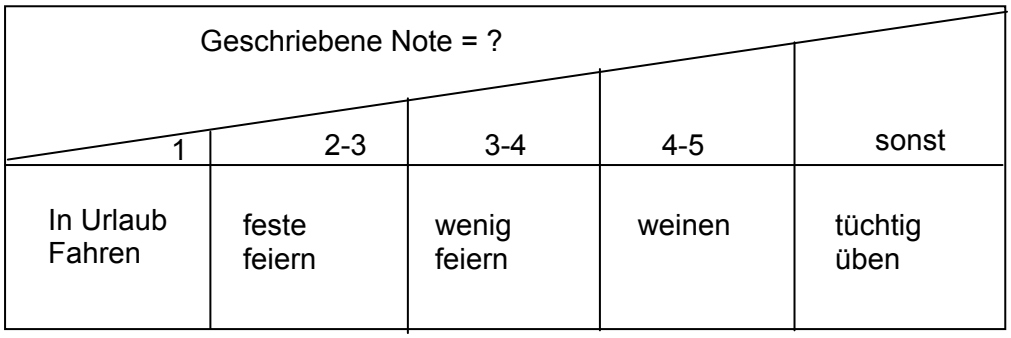
1. Das Programm-Modul



2. Die Bedingte Anweisung (if-Abfrage)

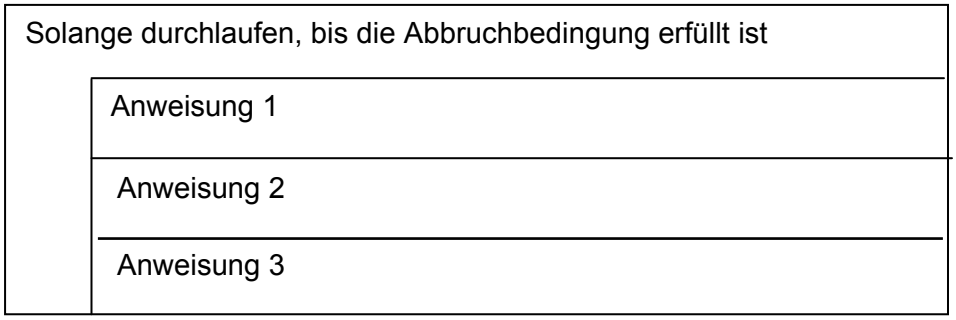


3. Die Case-Anweisung



4. Die Schleifen

4.1 die kopfgesteuerte Schleife
in die kopfgesteuerte wird nur verzweigt, wenn die Laufbedingung erfüllt ist oder die Abbruchbedingung nicht erfüllt ist



4.2 Die fussgesteuerte Schleife
in die fussgesteuerte Schleife wird immer verzweigt, und solange durchlaufen, wie die Laufbedingung erfüllt ist oder die Abbruchbedingung nicht erfüllt ist

Anweisungungen
Solange durchlaufen, bis die Abbruchbedingung erfüllt ist

Beispiel: Bestellung bearbeiten

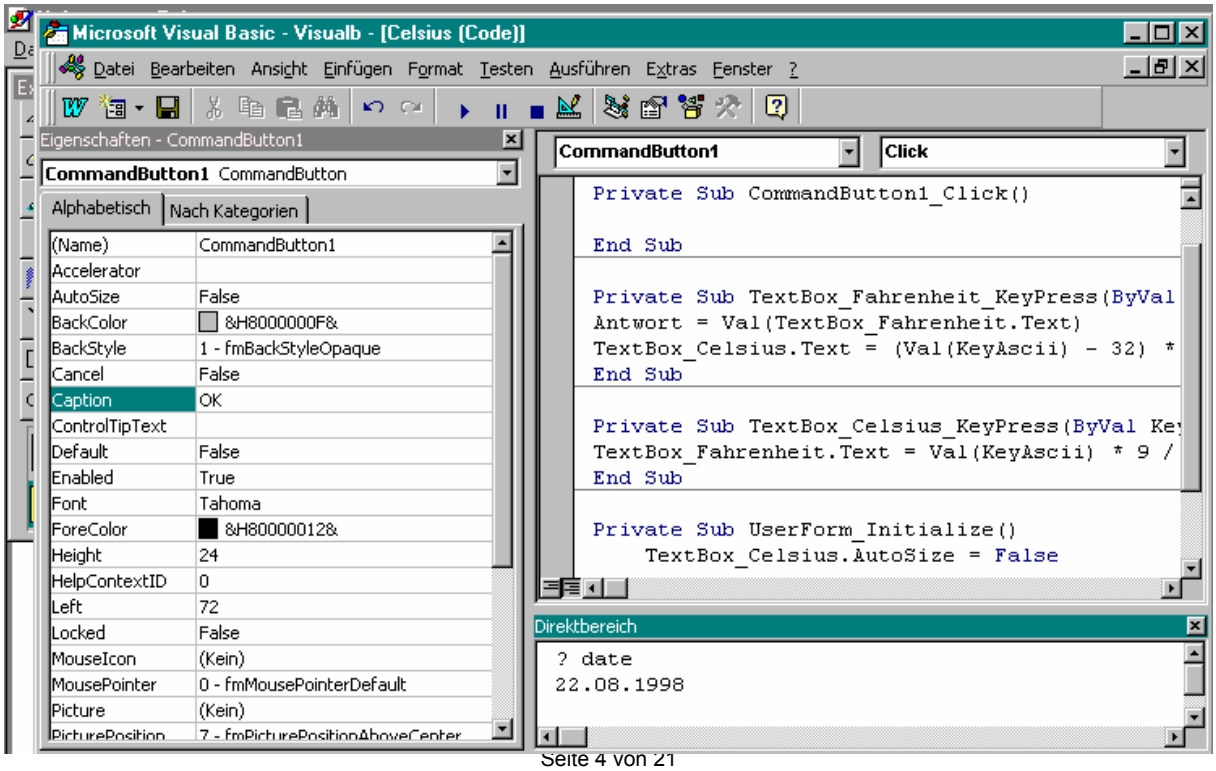
Bestellung		
Bestell-Anruf entgegennehmen		
Ist Kundenadresse vorhanden ?		
ja	**nein**	
	Kunden-Adress-Daten eingeben	
Nachfrage Artikel und Anzahl		
Artikel in Anzahl auf Lager ?		
ja	**nein**	
Artikel in Auftragsliste übernehmen	**Kann Artikel in 10 Tagen zugekauft werden ?**	
	ja	**nein**
	Artikel in Auftragsliste übernehmen	**Artikel ablehnen**
solange bis letzter Artikel aufgenommen		
Lieferadresse gleich Kundenadresse ?		
ja	**nein**	
	Vermerk über Lieferadresse	
Auftragsbestätigung veranlassen		

Einführung VBE

In allen MS-Office-Anwendungen gibt es einen Makrorekorder, der immer wiederkehrende Aktionen als Makro speichert. Dabei übersetzt der Makrorekorder die Maus- und Tastaturaktivitäten in die Officeinterne Programmiersprache Visual Basic für Anwendungen **(VBA)**.
Solchen Makros fehlt es jedoch fast immer an der nötigen Flexibilität. Wer daraus richtige Programme mit einer gehörigen Portion Intelligenz und Leistungsfähigkeit machen will, der muß die aufgezeichneten Quelltexte überarbeiten, oder noch besser schreibt sie von vornherein selbst.

Der Visual-Basic-Editor

Das Hauptwerkzeug des fortgeschrittenen Office-Entwicklers ist nicht mehr der Makrorekorder, sondernder der Visual-Basic-Editor (VBE).
Der läßt sich aus allen 97er Versionen von Word, Excel, Access und PowerPoint über den Befehl Extras/ Makro/VisualBasic-Editor und alternativ auch über das Tastenkürzel (Alt-F11) starten.
Die Abläufe zum Anlegen eines Makros weisen von Office-Anwendung zu Office-Anwendung geringfügige Unterschiede auf. Für Ihre ersten praktischen Experimente sollten Sie daher den Visual-Basic-Editor aus Word heraus aktivieren.
Bevor Sie einen Quelltext eingeben, entscheiden Sie zunächst, in welchem Dokument oder welcher Dokumentvorlage Word das künftige Makro speichert. Das Projekt-Fenster (Ansicht/Projektexplorer) des Visual-Basic-Editors ermöglicht Ihnen Sie den Speicherort eines Makromoduls festzulegen.
Der Projekt-Explorer listet die Namen aller geöffneten auf. Wenn Sie sich für eines der Dokumente entscheiden, müssen Sie es immer zuerst laden, bevor Sie ein darin gespeichertes Makro starten können. Eine Alternative dazu ist die globale Dokumentvorlage normal. dot: Diese Datei wird bei jedem Winword- Start geladen.
Der Eintrag Normal im Projektfenster repräsentiert die globale Dokumentvorlage, die Sie mit einem Mausklick markieren. Mit dem Menübefehl Einfügen/Modul legen Sie darin ein neues Modul an, das die virtuelle Heimat für ein oder mehrere Makros darstellt. Der Standardname des neuen Moduls lautet modul1.



Grundlemente von VBA

Objekte, deren Methoden und Eigenschaften

Objekte sind überall. Sie halten im Augenblick eine Buchobjekt in der Hand und sitzen auf einem Stuhl. Dann werden Sie vielleicht das Kühlschrankobjekt auf suchen, das voll mit leckeren Getränken gefüllt ist.

Objekte

Ein Objekt kann eine Person, ein Platz oder ein Ding sein. In Office-97 sind Objekte z.B. ein Word-Dokument, eine Excel-Mappe. Objekte sind also die Grundlage für VBA-Programme. Alle Objekte lassen sich beschreiben, das nennt man deren Eigenschaften. Bei Ihrem Buch sind es der Einband, die Größe und die Seitenzahl. Ein Word-Dukument besitzt eine Seitenzahl, einen Namen und ist schreibgeschützt.
Ein Dokument kann gespeichert und geöffnet werden oder es kann eine Rechtschreibung durchgführt werden. Das alles nennt man Methoden des Objektes.

Methoden des Objektes

Der Unterschied zwischen Methode und Eigenschaft besteht darin, das sie einem Objekt mit der Methode mitteilen können, das es etwas tun soll. Das Ereignis hingegen, beschreibt die Art und Weise wie es das tut, „Schau her ich mache etwas“.
Nun ein Beispiel: ***ActiveDocument.Close*** schließt das aktuelle Word-Dokument.

Methode und Objekt sind durch einen Punkt getrennt.
ActiveDocument.SaveAs Filename:=“meinword.doc“
Hier wird die Methode ***SaveAs*** des Objektes ***ActiveDocument*** aufgerufen und es wird gleichzeitig der Dateiname (***Filename=“...“***) mitgeteilt, sodass das aktive Dokument mit der Funktion „speichern unter“ gespeichert wird.

Die Eigenschaften eines Objektes

Nun ein Beispiel zu den Eigenschaften:
ThisDokument.ShowSpellingError=False

bedeutet „zeige im aktuellen Objekt die Rechtschreibfehler nicht mehr an“

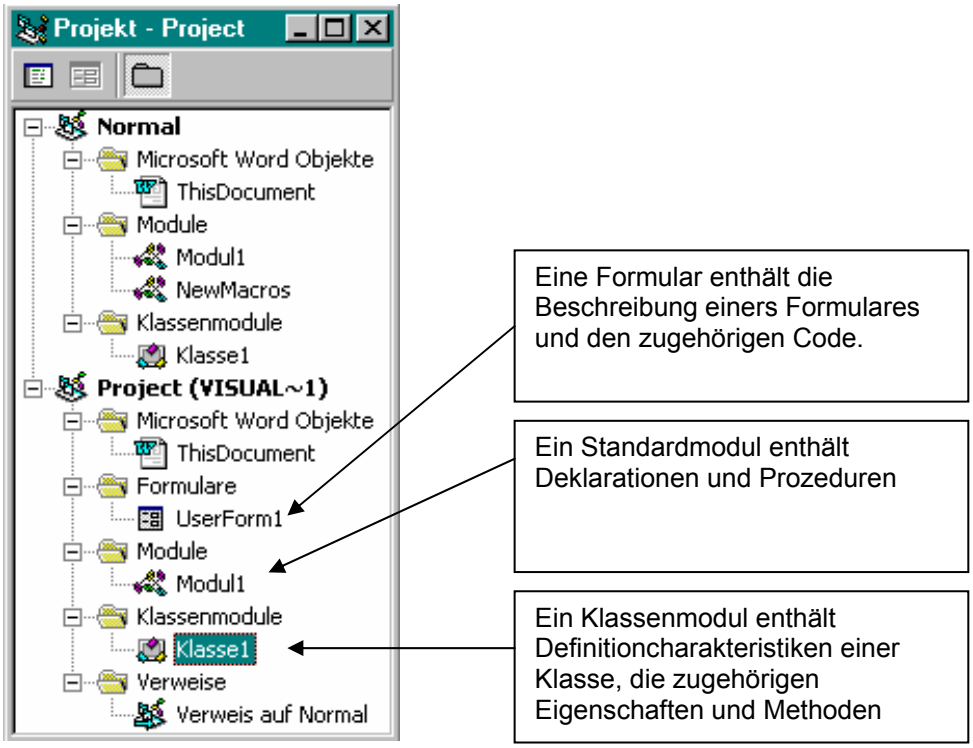
Visal Basic Werkzeuge

Visual Basic enthält eine Reihe von Werkzeugen, die Sie zur Gestaltung von grafischen Anwendungen einsetzen können. In dieser Lektion werden die folgenden Themen behandelt:

- Projekte
- Formen und Steuerelemente
- Module
- Die Visual Basic-Sprache
- Menüleisten
- Die Farbpalette

Ein Projekt

Ein Projekt ist eine Sammlung von Form-, Standard- und Klassenmodulen sowie Ressourcendateien, aus denen eine Anwendung zusammengesetzt ist. Das Projektfenster zeigt alle Dateien in einer Anwendung an.



Ein Formular

Eine Form beinhaltet die mit der Form verbundenen Steuerelemente und den mit der Form verbundenen Code. Sie können den Code im ganzen Projekt nutzen, indem er in ein Form- oder Standardmodul eingefügt und die Prozedur als Public deklariert wird.

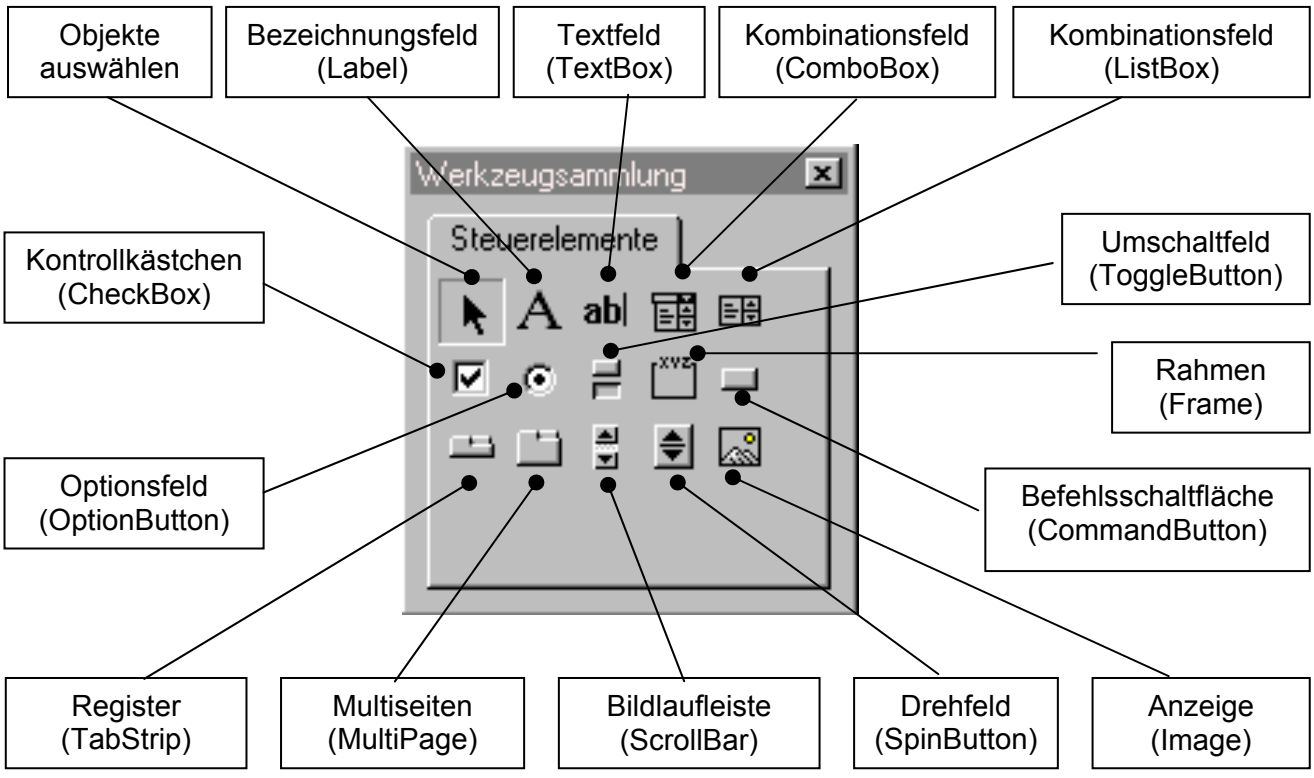
Sie erstellen Formen, die die Oberfläche Ihrer Anwendung bilden. Jede Form ist ein Fenster, das Steuerelemente, Grafiken oder andere Formen anzeigt.

Sie können Formen auf viele verschiedene Weisen verwenden:

Steuerelemente sind Werkzeuge
(z.B. Felder, Schaltflächen und Bezeichnungen)

Steuerelemente sind Werkzeuge, z.B. Felder, Schaltflächen und Bezeichnungen, die Sie auf einer Form zeichnen, um Eingaben zu ermöglichen und Ausgabedaten anzuzeigen. Sie tragen auch zur optischen Gestaltung der Anwendung bei.

Die Werkzeugsammlung
Mit der Werkzeugsammlung zeichnen Sie Steuerelemente auf einer Form.



Objekte auswählen
Der einzige Befehl in der Werkzeugsammlung, mit dem kein Steuerelement erstellt wird. Mit Objekte auswählen kann lediglich ein bereits in einem Formular erstelltes Steuerelement verschoben oder dessen Größe angepaßt werden.

Bezeichnungsfeld (Label)
Ermöglicht Ihnen, Text festzulegen, den die Benutzer nicht ändern können, wie z.B. Beschriftungen unter Abbildungen.

Textfeld (TextBox)
Enthält Text, den die Benutzer eingeben oder ändern können.

Kombinationsfeld (ComboBox)
Ermöglicht das Erstellen einer Kombination aus einem Listenfeld und einem Textfeld. Die Benutzer können ein Objekt aus der Liste auswählen oder im Textfeld eine Eingabe vornehmen.

Listenfeld (ListBox)
Verwenden Sie das Listenfeld zur Anzeige einer Liste mit Elementen, unter denen die Benutzer eine Auswahl treffen können. Enthält die Liste mehr Elemente als gleichzeitig dargestellt werden können, kann in der Liste ein Bildlauf durchgeführt werden.

Kontrollkästchen (CheckBox)
Erstellt ein Feld, das die Benutzer aktivieren können, um eine Wahr/Falschbedingung auszudrücken, oder das mehrere Wahlmöglichkeiten darstellt, aus denen die Benutzer mehr als eine Möglichkeit wählen können.

Optionsfeld (OptionButton)
Ermöglicht die Anzeige mehrerer Wahlmöglichkeiten, aus denen der Benutzer nur genau eine Option auswählen kann.

Umschaltfeld (ToggleButton)
Erstellt eine Schaltfläche, mit der ein Element ein- oder ausgeschaltet werde kann.

Rahmen (Frame)
Ermöglicht das Erstellen einer grafischen oder funktionalen Gruppierung für Steuerelemente. Erstellen Sie zum Gruppieren von Steuerelementen zunächst den Rahmen und dann die Steuerelemente innerhalb des Rahmens.

Befehlsschaltfläche (CommandButton)
Erstellt eine Schaltfläche, die die Benutzer zum Ausführen eines Befehls auswählen.

Register (TabStrip)
Ermöglicht die Definition mehrerer Seiten für einen Fenster- oder Dialogfeldbereich in der Anwendung.

Multiseiten (MultiPage)
Faßt mehrere Bildschirme mit Informationen innerhalb einer Gruppe zusammen.

Bildlaufleiste (ScrollBar)
Ein grafisches Hilfsmittel, um sich innerhalb einer Liste mit vielen Elementen oder Informationen zu bewegen, um die aktuelle Position auf eine Skala anzugeben oder um Geschwindigkeit oder Menge anzugeben oder anzuzeigen.

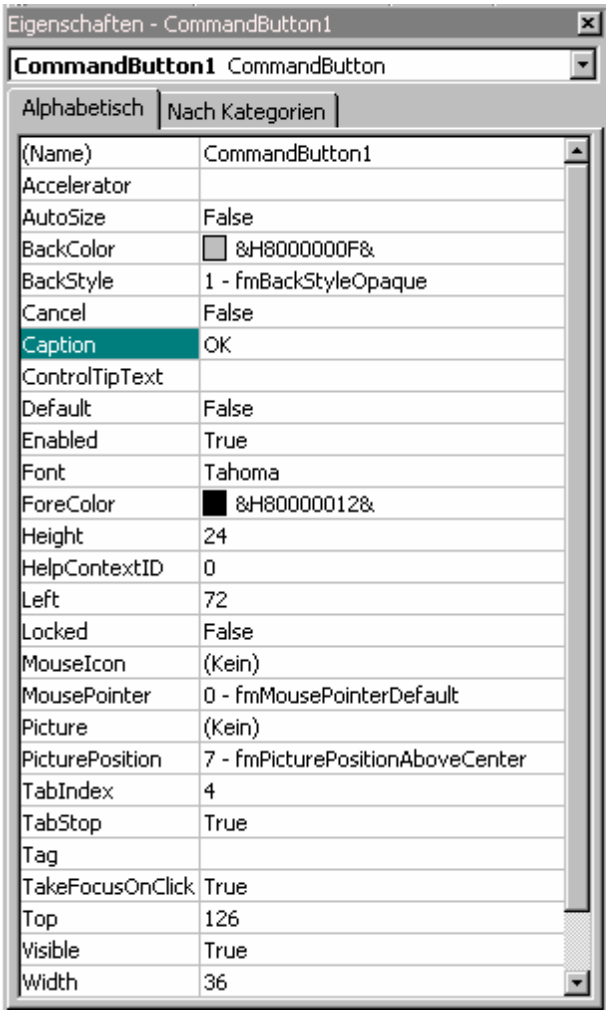
Drehfeld (SpinButton)
Dieses Steuerelement kann mit einem anderen Steuerelement zum Erhöhen und Reduzieren von Zahlen verwendet werden. Es kann auch zum Blättern in Werten oder Listen eingesetzt werden.

Anzeige (Image)
Zeigt eine grafische Darstellung aus einer Bitmap, einem Symbol oder einer Metafile-Datei in Ihrem Formular an. Bilder, die in einem Anzeige-Steuerelement (Image) angezeigt werden, dienen nur zur reinen Darstellung und erfordern weniger Speicherplatz als ein Bildfeld-Steuerelement (PictureBox).

Das Eigenschaftenfenster

Mit Hilfe des Eigenschaftenfensters legen Sie die Eigenschaften von Formen und Steuerelementen fest. Eigenschaften bestimmen die Startwerte für solche Merkmale wie Größe, Name und Position.

Das Eigenschaftenfenster enthält eine Liste aller Eigenschaften und ihre Einstellungen für das derzeit ausgewählte Modul oder Steuerelement. Eigenschaften können alphabetisch oder nach Kategorie (Bild, Darstellung, Schriftart, etc.) angezeigt werden.



In Anwendungen mit vielen Befehlen ermöglicht Visual Basic es Ihnen, die Befehle in einer Menüleiste zu gruppieren.

Von Zahlen und Zeichenketten

Die meisten Makrosprachen – und VBA bildet da keine Ausnahme – gehen mit unterschiedlichen Datentypen um. Grundsätzlich unterscheidet man zwischen Zahlen und Zeichenketten. Zahlen setzen sich ausschließlich aus dcn Ziffern 0 bis 9 zusammen sowie gegebenenfalls einem negativen Vorzeichen.

Um eine Konstante zu deklarieren verwenden Sie die Anweisung Const.
Die Anweisung:
Const PI As Double= 3,14159265
definiert eine Konstante namens PI.

Zeichenketten können jedes Zeichen enthalten, das der PC darstellen kann, also sämtliche Buchstaben, Ziffern, Satz- und Sonderzeichen in beliebigen Kombinationen.
Zeichenketten werden im Datentyp ***String*** aufgenommen.
Das Schlüsselwert ***Dim*** leitet immer die Definition einer Variablen ein.
Die Anweisung:
Dim meldetext As String
legt beispielsweise eine Variable meldetext zur Aufnahme einer Zeichenkette an.

Einen pauschalen Zahlentyp gibt es nicht. So dient der Zahlentyp ***Integer*** für die Darstellung ganzer Zahlen von -32 768 bis 32 767.
Die Anweisung:
Dim Jahreszahl As Integer
legt beispielsweise eine Variable ***Jahreszahl*** zur Aufnahme als ganze Zahl an.

Der Typ ***Long*** dagegen stellt ganze Zahlen von -2 147 483 646 bis 2 147 483 647 dar.
Allerdings wächst der interne Speicherbedarf damit von 2 auf 4 Byte pro Zahl.
Für die Darstellung von Gleitkommazahlen stehen die Datentypen ***Single*** und ***double*** zur Verfügung.
Die Anweisung:
Dim Betrag As double
legt beispielsweise eine Variable ***betrag*** zur Aufnahme als Gleitkommazahl an.

Der Datentyp ***date***, kann Kalenderdaten vom 1.01.100 bis 31.12.9999 repräsentieren.

Ein weiterer wichtiger Datentyp sind Datenfelder. Datenfelder dienen zur Aufnahme gleichartiger Daten – z.B. Teilnehmernamen . Der Zugriff auf die einzelnen Datenfeldelemente erfolgt über eine Indexzahl.
Die Anweisung:
Dim Teilnehmer (1 to 5) As String
Legt eine Feldvariable namens Teilnehmer fest mit 5 möglichen Teilnehmernamen.
Die Teilnehmer werden folgendermaßen zugewiesen:
Teilnehmer (1) = „Huber“
Teilnehmer (2) = „Müller“
*****
Teilnehmer (5) = „Schulze“

Datentypen

Schlüssel-wort	Datentyp
Byte	Ganzzahl (0 bis 255)
Boolean	Logischer Wert (True oder False)
Integer	Ganzzahl (-32 768 bis 32 767)
Long	32-Bit-Ganzzahl (-2147483 646 bis 2 147 483 647)
Decimal	96-Bit-Ganzzahl (+/ -79 228162 514 264 337 593 543.950 335)
Single	Gleitkommazahl einfacher Genauigkeit (+/-3,402823E38)
Double	Gleitkommazahl doppelter Genauigkeit (+/-1,79769313486232E308)
Currency	Gleitkommazahl für Währungsberechnungen (-922 337 203 685 477,5808 bis 922 337 203 685 477,5807)
Date	Datum (1.1.100 bis 31.12.9999)
String	Zeichenkette (ca. 2 Milliarden Zeichen)
Variant	Zeichenketten und numerische Werte im Bereich des Datentyps Long

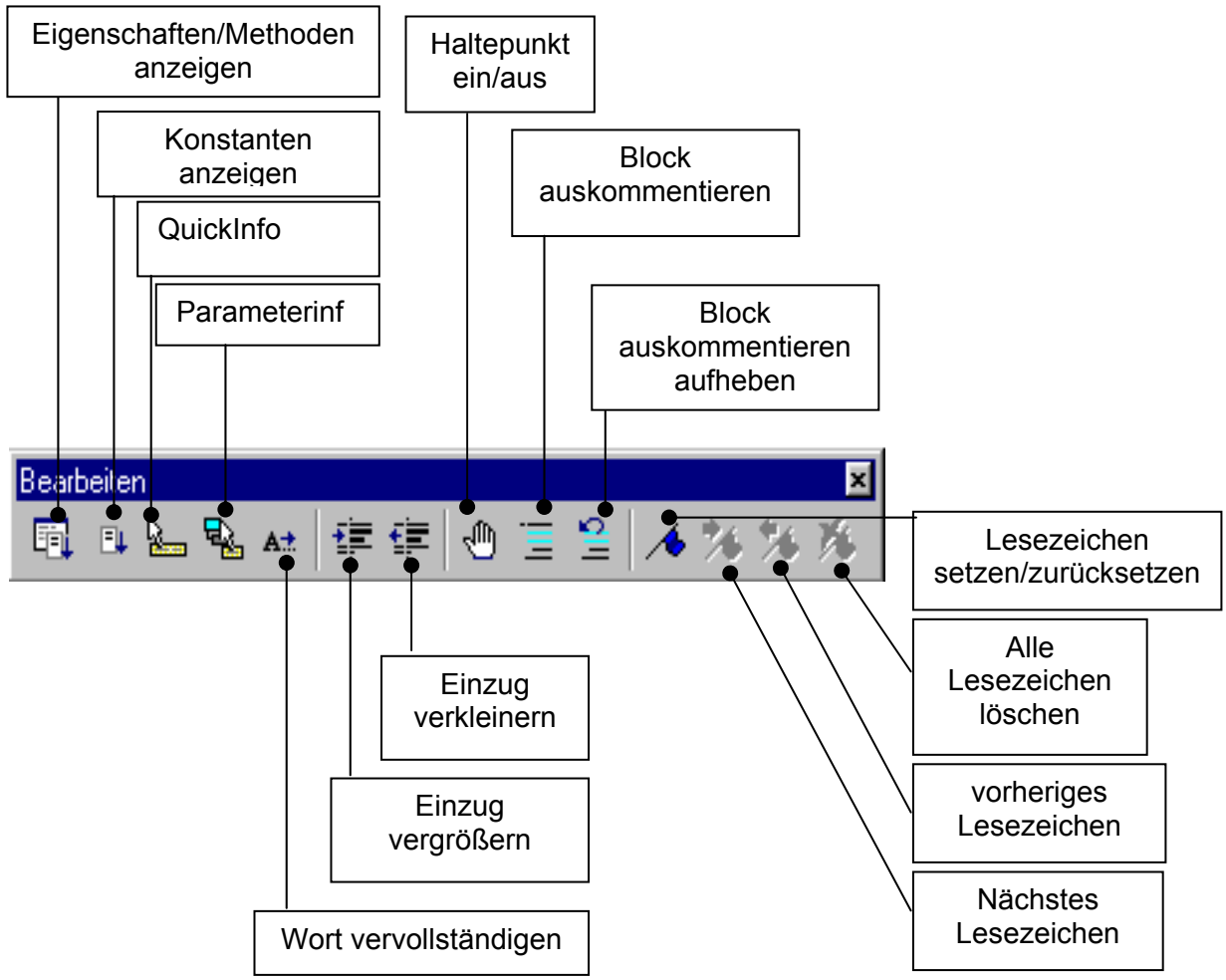
VBA Rechenoperationen

Operator	Anwendung	Beispiel	Ausgabe
+	Addition	Wert=6 + 4	10
-	Subtraktion	Wert =10 - 7	3
*	Multiplikation	Wert=2 * 4	8
/	Division	Wert=8 / 2	4
\	Division mit ganzzahligem Ergebnis	Wert = 5 \ 2	2
MOD	gibt den Rest einer ganzzahligen Division zweier Zahlen zurück	Wert = 1O Mod 3	1
^	Potenzieren	Wert = 2 ^4	16
	Verknüpfen von Zeichenketten	Wert = „PC“ & „Magazin“	PC Magazin

Vergleichoperatoren

Operator	Bedeutung
<	Kleiner als
>	Größer als
=	Gleich
<=	Kleiner oder gleich
>=	Größer oder gleich
<>	Ungleich

Die Symbolleistenschaltflächen



Eigenschaften/Methoden anzeigen

Öffnet ein Feld im Codefenster mit den Eigenschaften und Methoden, die für das Objekt vor dem Punkt (.) verfügbar sind.

Konstanten anzeigen
Öffnet ein Feld im Codefenster mit den Konstanten, die für die Eigenschaft zur Verfügung stehen, die Sie eingegeben haben und die vor dem Gleichheitszeichen (=) steht.

QuickInfo
Stellt Ihnen Informationen zur Syntax für eine Variable, Funktion, Methode oder Prozedur zur Verfügung, auf deren Namen sich der Zeiger befindet.

Parameterinfo
Zeigt im Codefenster ein Popup-Fenster mit Informationen zu den Parametern der Funktion an, in der sich der Zeiger befindet.

Wort vervollständigen
Übernimmt die Zeichen, die Visual Basic automatisch für das Wort, das Sie eingeben, ergänzt.

Einzug vergrößern
Verschiebt alle Zeilen des markierten Bereichs bis zum nächsten Tabstop.

Einzug verkleinern
Verschiebt alle Zeilen des markierten Bereichs bis zum vorherigen Tabstop.

Haltepunkt ein/aus
Setzt oder entfernt einen Haltepunkt in der aktuellen Zeile.

Block auskommentieren
Fügt Kommentarzeichen am Anfang der Zeilen eines markierten Textbereichs ein.
Auskommentierung des Blocks aufheben.
Entfernt die Kommentarzeichen am Anfang der Zeilen eines markierten Textbereichs.

Lesezeichen setzen/zurücksetzen
Aktiviert bzw. deaktiviert ein Lesezeichen für die aktive Zeile im Codefenster.

Nächstes Lesezeichen
Setzt den Fokus auf das nächste Lesezeichen in der Lesezeichenliste.
Vorheriges Lesezeichen.
Verschiebt den Fokus auf das vorherige Lesezeichen in der Lesezeichenliste.

Alle Lesezeichen löschen
Löscht alle Lesezeichen.

Funktionen von Visual Basic zu Code-Bearbeitung

Visual Basic bietet folgende Funktionen, damit Sie Ihren Code schnell überarbeiten können:
- Automatische Syntaxüberprüfung
- Testwerkzeuge
- Ein Testfenster

Die Visual Basic-Symbolleiste enthält Schaltflächen für viele häufig verwendete Gestaltungs- und Testbefehle.
Indem Sie die Visual Basic-Formen, -Werkzeuge und -Programmiersprache miteinander verbinden, können Sie leistungsfähige Anwendungen schnell und einfach erstellen.
Die Anwendung enthält Schaltflächen, die Tastenkombinationen für häufig verwendete Menübefehle zum Bearbeiten von Code darstellen.
Klicken Sie einmal auf eine Symbolleistenschaltfläche, um den durch diese Schaltfläche repräsentierten Vorgang auszuführen. Aktivieren Sie im Dialogfeld Optionen auf der Registerkarte Allgemein das Kontrollkästchen QuickInfo anzeigen, um QuickInfo für die Symbolleistenschaltflächen anzuzeigen.

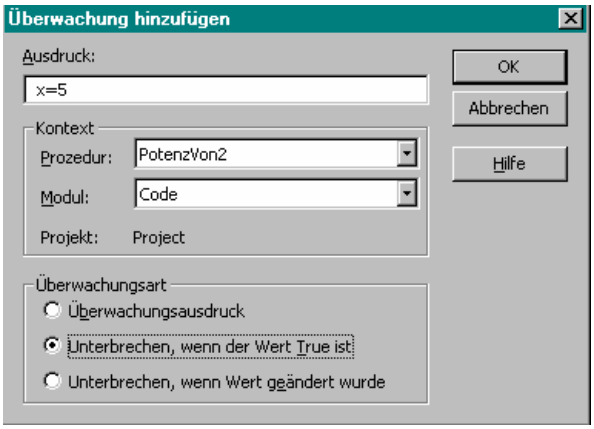
1. Beispiel:
Erstellen Sie ein Modul *RechneFuerMich*, dem Sie eine Ganzzahl übergeben wird, mit der dann in dieser Funktion gerechnet wird.
Sub *RechneFuerMich (y As Integer)*
Nach der Dim-Anweisung weisen Sie an, das bei Fehlern zur Marke *Fehler:* gesprungen werden soll:
On Error GoTo Fehler.
Setzen Sie die Vatiable x auf 10 und durcken Sie folgendes aus: *Debug.Print x / y*
Im Falle eine Fehlers (y=0) geben Sie in einer if-Anweisung (*Err.Number = 11*; d.h, Division durch 0) eine neue Zahl y durch eine inputbox ein (*y = InputBox("Bitte eine neu Zahl eingeben <> 0")*
Resume, bewirkt, das der nächste Befehl nach der Fehlerzeile angearbeitet wird.

Haltepunkte setzten:

Klicken Sie dort in den Code, wo der Haltepunkt wirken soll und drücken Sie ***„Testen/Haltepunktsetzen“ oder F9.*** Sobald die Programmablauf an diese Stelle gerät, wird unterbrochen.

Überwachen von Variablen:
Öffnen Sie das Lokal-Fenster durck Klick auf ***„Ansicht/Lokal-Fenster“*** und schauen Sie sich an, wie die Variablen- hier x und y - verhalten.

Überwachungsausdruck definieren und aktivieren
Wenn Sie möchten, dass der Programmablauf beim erreichen eines bestimmten Zustandes einer Variablen unterbrochen wird, rufen Sie das Überwachungsfenster durch Klick auf „Testen/Überwachnungsaudruck“ auf und geben einen Ausdruck – hier x=5 und die Option „Unterbrechen, wenn der Wert wahr“ ein.
Die andere Option „Unterbrechen, wenn der Wert geändert wurde, stoppt das prgramm, wenn der Wert nicht mehr mit dem im Fenster „Ausdruck“ übereinstimmt.



Geben Sie nun folgenden Code von *„PotenzVon“* in Ihr Code-Fenster ein und testen Sie

Variablen müssen deklariert werden, bevor Sie benutzt werden können.

```
Option Explicit
Dim Y As Integer
On Error GoTo Fehler
Sub PotenzVon2()
   Dim x As Integer
   For x = 1 To 10
      Y = 2 ^ x
        if (Err.Number = 11)    goto Fehler:
        else
                Debug.Print "Die " & x & ". Potenz von 2 ist " & Y
        endif

   Next
Fehler:
        Resume next
End Sub
Sub CodeAusführen()
PotenzVon2
End Sub
Sub RechneFuerMich(y As Integer)
      Dim x As Integer
      x = 10
      y = 2 ^ x
      Debug.Print x / y
   Next
End Sub
```

Geben Sie im Direkfenster ein:
1. *RechneFuerMich(5)*
2. *RechneFuerMich(0)*

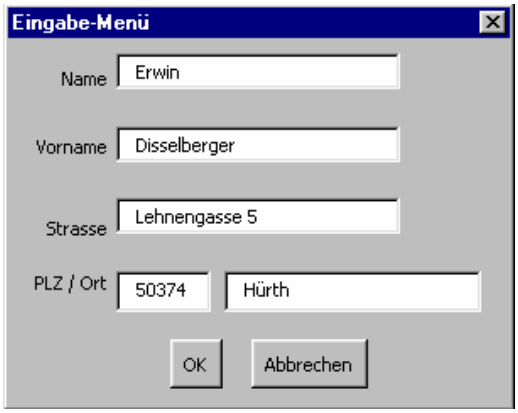
Haltepunkte setzten:

Klicken Sie dort in den Code, wo der Haltepunkt wirken soll und drücken Sie ***„Testen/Haltepunktsetzen“ oder F9.*** Sobald die Programmablauf an diese Stelle gerät, wird unterbrochen.

Überwachen von Variablen:

Öffnen Sie das Lokal-Fenster durck Klick auf ***„Ansicht/Lokal-Fenster“*** und schauen Sie sich an, wie die Variablen- hier x und y - verhalten.

Sie erstellen eine Anwendung zur Eingabe von Adress-Daten



Eine Anwendung wird in drei Schritten erstellt:

1. Erstellen der Benutzeroberfläche
2. Festlegen der Eigenschaften
3. Schreiben des Codes

Erstellen eine Anwendung, die auf Benutzeraktionen oder Systemereignisse reagiert, indem Sie Code für Ihre Formen und Steuerelemente schreiben.
Die Anwendung soll ein Eingabe für einen Adress-Datensatz ermöglichen.

Fügen sie als ersten mit dem ***Menüpunkt Einfügen/Userform*** ein Formular ein.
Dann plazieren Sie durch klicken und ziehen 5 Bezeichnungsfelder (Name...PLZ/Ort).
Im Eigenschaftsfenster tragen sie bitte unter ***Caption*** den richtigen Text ein.
Die Caption-Eigenschaft legt den Text fest, der im Steuerelement angezeigt wird. Die Caption-Eigenschaft der Bezeichnungsfelder wird in "Name“, “Vorname", „Strasse“, PLZ“ und „Ort“ geändert.
Nun erzeugen sie die 5 Textfelder ***Textbox1... Textbox5*** und die beiden Befehlsschaltflächen mit der Bezeichnung ok und Abbrechen.
Ändern sie nun auch Name-Eigenschaft der Befehlsschaltflächen (CommandButton) in ***txtName, txtVorName, txtStrasse, txtOrt, txtPLZ, OKBotton*** und ***AbbrechenBotton***.
Die Caption-Eigenschaft in ***OK*** und ***Abbrechen***.

Als erstes muss nun das Fomular initialisert werden mit folgendem Code:

```
Private Sub UserForm_Initialize()
        TextName.MultiLine = True
        TextName.WordWrap = True
        TextName.EnterKeyBehavior = False
        TextVorname.MultiLine = True
        TextVorname.WordWrap = True
        TextStrasse.MultiLine = True
        TextStrasse.WordWrap = True
        TextPLZ.MultiLine = True
        TextPLZ.WordWrap = True
        TextOrt.MultiLine = True
        TextOrt.WordWrap = True
 End Sub
```

Die Eigenschaft MultiLine gibt an, ob ein Steuerelement mehrere Textzeilen akzeptieren und anzeigen kann.
Die Eigenschaft ***WordWrap*** gibt an, ob der Inhalt eines Steuerelements am Zeilenende automatisch umbrochen wird.
Die Eigenschaft ***EnterKeyBehavior*** definiert, welche Wirkung das Drücken der *EINGABETASTE* in einem Textfeld-Steuerelement (*TextBox*) hat.
Dann brauchen wir nach folgende Funktionen:

```
Private Sub TextName_Change()
        Name = TextName.Text
End Sub
Private Sub TextVorName_Change()

End Sub
Private Sub TextStrasse_Change()

End Sub
Private Sub TextPLZ_Change()

End Sub
Private Sub TextNOrt_Change()

End Sub
```

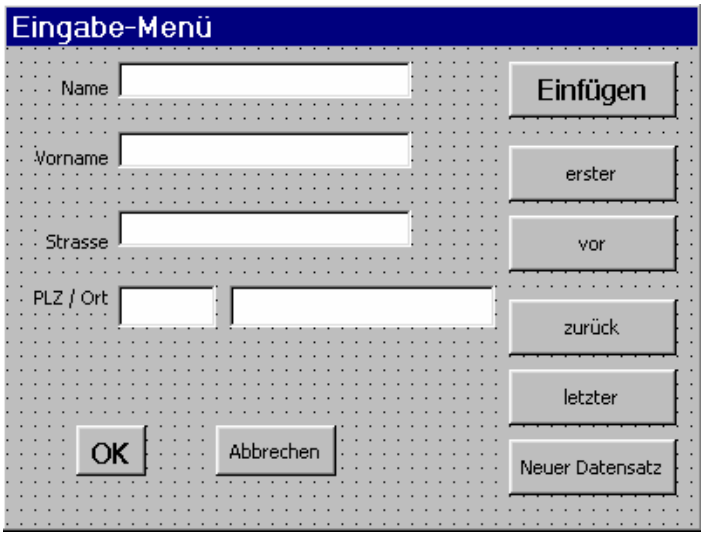
Das Ereignis ***Change()*** tritt ein, wenn sich die ***Value***-Eigenschaft (Inhalt der Box) ändert.

```
Private Sub Abbrechen_bottom_Click()
        Unload Me
End Sub
```

Das Ereignis ***bottom_Click()*** führt zum Schliessen des Formulares.
Unsere Adress-Datensätze sollen nun in einer Access-Datenbank abgelegt werden (C:\euroschulen\ibw97\adressen.mdb).
Die Tabelle ***Privat-adressen*** der Datenbank hat folgende Struktur:

Nachname	Vorname	Strasse	PLZ	Ort
Hoffman	Peter	Siegesstr.5	51045	Köln
Hubert	Josef	Frankfurterstr.3	83954	München
Meister	Werner	Lindenstr.12	14536	Berlin
Schmitt	Herbert	Arnoldstr.67	83954	München
Kuhn	Marta	Berlinerstr.32	43855	Neuss

Alle Felder (Nachname bis Ort) sind vom Typ Text und sollen 30 Zeichen aufnehmen (ausser PLZ = 8 Zeichen) können.
Unser Formular muss wie folgt neu gestaltet werden:



- Die Button vor und zurück bewegt uns von einen Datensatz zum nächsten und zurück.
- Die Button erster und leztzer bewegt uns zum 1. Datensatz und zum letzten Datensatz.
- Der Button Neuer Datensatz speichert einen neuen Datensatz in der Datenbank
- Der Button Einfügen überrägt den aktuellen Datensatz in das Word-Dokument

Der Code muss wie folgend beschrieben ergänzt werden:

1. Als erstes müssen sie mit dem Menü-Punkt (Extras/Verweise) einen Verweis auf eine Bibliothek (Microsoft DDAO 3.5 Object Library) herstellen, damit die Funktionen zur Bearbeitung von Accsedatenbanken zur Verfügung stehen.
2. Dann müssen zwei neue Variablen (db für Datenbank und Rs für den jeweils zu bearbeitenden Datensatz) deklariert werden:
   Dim db As Object
   Dim Rs As Recordset
3. In der Initialisierungs-SUB für das Formular kann bereits der Zugriff auf die Datenbank ermöglicht werden:
   Set db = OpenDatabase(Name:="C:\eigene dateien\adressen.mdb")
   Set Rs = db.OpenRecordset(Name:="privat_adressen")
4. Beim Abspeichern eines neuen Datensatzes müssen wir einen neuen Datensatz (Record) erzeugen (.AddNew), *Namen* bis *ort* an die Felder der Datenbank *(.Fields.Value)* übergeben und updaten.

```
With Rs
    .AddNew
    .Fields(0).Value = Name
    .Fields(1).Value = VorName
    .Fields(2).Value = strasse
    .Fields(3).Value = plz
    .Fields(4).Value = ort
    .Update
End With
```

1. Unseren Datensatz-Zeiger bewegen wir mit ***MoveNext, MovePrevious, MoveFirst*** und ***MoveLast***:

```
Private Sub erster_Click()
With Rs
   .MoveFirst
   TxtName.Text = .Fields(0).Value
   TxtVorname.Text = .Fields(1).Value
   TxtStrasse.Text = .Fields(2).Value
   TxtPLZ.Text = .Fields(3).Value
   TxtOrt.Text = .Fields(4).Value
End With
End Sub
```

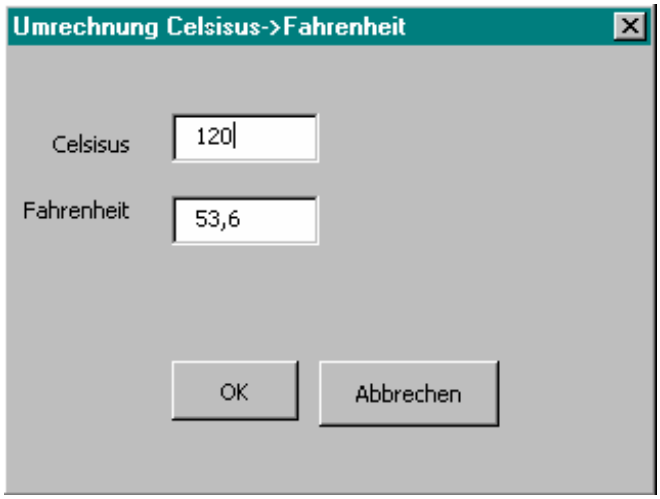
2. Das Übertragen des Adress-Datensatzes in unser Dokument erfolgt der einfachheit halber in eine bestehende Tabelle (4 Zeilen, 1 Spalte), wobei der cursur sich in der ersten Zeile befindet:

```
Selection.InsertAfter (VorName)
Selection.InsertAfter (" ")
Selection.InsertAfter (Name)
Selection.MoveDown Unit:=wdLine, Count:=1
Selection.InsertAfter (strasse)
Selection.MoveDown Unit:=wdLine, Count:=1
Selection.MoveDown Unit:=wdLine, Count:=1
Selection.InsertAfter (RTrim(plz))
Selection.InsertAfter (" ")
Selection.InsertAfter (ort)
```

3. Beim Verlassen des Formulares schliessen wir die Datenbank
   Rs.Close
   db.Close
   unload me

Anwendung zur Umwandlung von Temperaturen

Die Anwendung zur Umwandlung von Temperaturen besteht aus diesen Elementen:



Zum Erstellen einer Anwendung müssen Sie zunächst ein neues Projekt öffnen, falls Sie nicht schon in einem arbeiten.

Sie öffnen ein Projekt, indem Sie aus dem Menü Datei den Befehl Neues Projekt wählen. Jedes neue Projekt enthält eine Form. Sie können so viele zusätzliche Formen hinzufügen, wie Sie für Ihre Anwendung benötigen. Für die Anwendung zur Umrechnung von Temperaturen wird nur eine Form benötigt.

Als nächstes können Sie die benötigten Werkzeuge aus der Werkzeugsammlung auswählen, um die Steuerelemente zu zeichnen, die Sie in der Form benötigen.

In dieser Anwendung akzeptieren die Textfelder Benutzereingaben und zeigen Text an.
Zum Erstellen eines Steuerelements wählen Sie das Werkzeug aus der Werkzeugsammlung aus und ziehen anschließend den für das Steuerelement gewünschten Bereich in der Form, wobei Sie die linke Maustaste gedrückt halten.

Sie können ein Steuerelement in der Standardgröße erstellen, indem Sie auf das Werkzeug doppelklicken oder ein Werkzeug auswählen und die EINGABETASTE drücken.
Es werden zwei Textfelder erstellt, eines für Temperaturen in Fahrenheit, eines für Temperaturen in Celsius.

Mit Hilfe von Bezeichnungen wird der Inhalt von jedem Textfeld beschrieben. Bezeichnungen können vom Benutzer nicht verändert werden.

Sie legen die Eigenschaften eines Steuerelements im Fenster Eigenschaften fest.
Wenn Sie eine Form oder ein Steuerelement auswählen, werden dessen Eigenschaften und die entsprechenden Einstellungen im Fenster Eigenschaften angezeigt.

Sie verwenden die Name-Eigenschaft, um auf ein Steuerelement mit Code zu verweisen. In diesem Fall erhält das Textfelds den Namen txtCels.

Die Caption-Eigenschaft legt den Text fest, der im Steuerelement angezeigt wird. Die Caption-Eigenschaft des Bezeichnungsfeldes wird in "Celsius" geändert.
In diesem Schritt wird die Name-Eigenschaft des anderen Textfeldes in txtFahr und die Caption-Eigenschaft der Bezeichnung in "Fahrenheit" geändert.

Als nächstes muß der Code geschrieben werden, so daß die Anwendung auf die Aktionen des Benutzers reagiert.
Wenn Sie auf ein Steuerelement doppelklicken, springt der Fokus in das Codefenster. Die Codeschablone für die Standard-Ereignisprozedur des ausgewählten Steuerelements wird angezeigt.

Der Name des Steuerelements wird im Feld Objekt angezeigt.
Das Prozedurfeld enthält eine Liste von Ereignissen für das Steuerelement.

Da der Code in dieser Anwendung ausgeführt werden soll, sobald der Benutzer mit einer Taste nach unten bzw. oben geht, wählen Sie das ***KeyDown***- bzw. ***KeyUp***-Ereignis im Prozedurfeld aus.
Die Eingabewerte sollen in zwei Variablen ***Antwort_fahr*** und ***Antwort_Cel*** zwischengespeichert werden.
Visual Basic zeigt dann die Schablone für die Ereignisprozedur an, die geschrieben werden soll.
Die folgenden Formeln werden Sie zur Umrechnung von Temperaturen verwenden:

Cels = (Fahr - 32) * 5/9	Fahr = (Cels * 9/5) + 32

Als nächstes wird der Code für die Ereignisprozedur eingegeben.
In diesem Schritt wird dem anderen Textfeld eine ähnliche Ereignisprozedur zugewiesen.
Wenn Sie die Arbeit an Ihrer Anwendung beenden, soll das Projekt gespeichert werden.

Nachdem Sie aus dem Menü Datei den Befehl Projekt speichern gewählt haben, werden Sie zur Eingabe eines Namens für das Projekt aufgefordert.

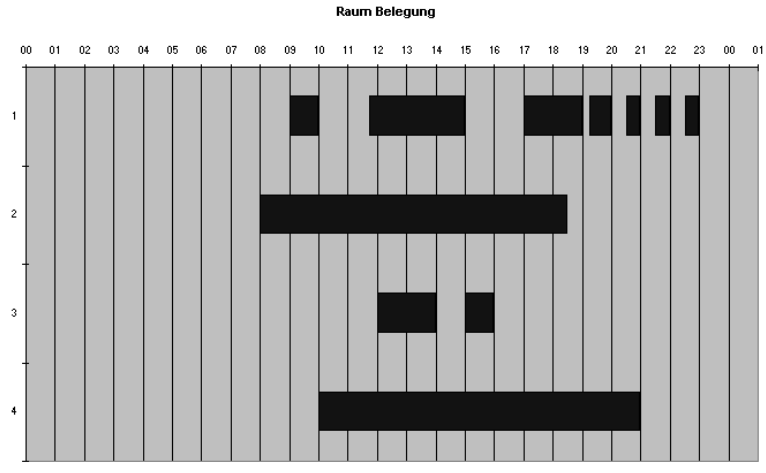
Sie können die Anwendung testen, indem Sie aus dem Menü Ausführen den Befehl Starten wählen oder auf der Symbolleiste auf die Schaltfläche "Starten" klicken.

Sie können auch die Testwerkzeuge verwenden, um Fehler in Ihrer Anwendung zu finden und Probleme zu beseitigen.
Wenn Sie die Anwendung ausführen, wird Celsius in Fahrenheit umgerechnet oder umgekehrt, sobald Sie eine Zahl eingeben und die EINGABETASTE drücken.

Excel-Anwendung: GANT-Diagramm zur Raumbelegung

Die Excel-Mappe besteht aus drei Tabellen

1. TimeChart



2. ChartData

Room	Start Time	Used	Not Used	Used	Not Used	Used	Not Used	Used	Not Used	Used	Not Used	Used	Not Used	Used
1	09:00	01:00	01:45	03:15	02:00	02:00	00:15	00:45	00:30	00:30	00:30	00:30	00:30	00:30
2	08:00	10:30												
3	12:00	02:00	01:00	01:00										

4. TimeData

Room	Start Time	End Time
1	9:00	10:00
1	11:45	15:00
1	17:00	19:00
1	19:15	20:00
1	20:30	21:00
1	21:30	22:00
1	22:30	23:00
2	8:00	18:30
3	12:00	14:00
3	15:00	16:00

Es werden die Raum-Nummer, die Start- und Endzeit jedes Raumes angegeben.

Das VBA-Programm trägt dann in die Tabelle ChartData für jeden Raum den Beginn der Raumbelegung ein,die benutzte und unbenutzte Zeit in Stunden des Tages
VBA-Code:

```
1.  Option Explicit
2.
3.  Sub CreateTimeChartData()
4.
5.      Dim vTimeData As Variant
6.      Dim i As Integer
7.      Dim sRoom As String
8.      Dim vLastEndTime As Variant
9.      Dim oSeries As Series                           '\ set up
10.     Application.ScreenUpdating = False
11.     Application.DisplayAlerts = False               '\ create chart data worksheet
12.     With Worksheets("TimeData").Range("TimeList").CurrentRegion
13.         .Sort Key1:="Room", Key2:="Start Time", Header:=xlYes
14.         vTimeData = .Value
15.         Worksheets.Add
16.         On Error Resume Next
17.             Worksheets("ChartData").Delete
18.             Charts("TimeChart").Delete
19.         On Error GoTo 0
20.         ActiveSheet.Name = "ChartData"
21.         .Columns(1).AdvancedFilter Action:=xlFilterCopy, _
22.             CopyToRange:=Range("A1"), Unique:=True
23.     End With
24.     Range("a1").Select
25.     For i = 2 To UBound(vTimeData)
26.         If vTimeData(i, 1) <> Selection.EntireRow.Cells(1) Then
27.             Selection.Offset(1).EntireRow.Cells(1).Select
```

```
28.             Selection.Value = vTimeData(i, 1)
29.             vLastEndTime = 0
30.         End If
31.         Selection.Offset(0, 1).Select
32.         Selection.Value = vTimeData(i, 2) - vLastEndTime
33.         Selection.Offset(0, 1).Select
34.         Selection.Value = vTimeData(i, 3) - vTimeData(i, 2)
35.         vLastEndTime = vTimeData(i, 3)
36.     Next i
37.     With Selection.CurrentRegion
38.         .Offset(0, 1).NumberFormat = "h:mm"
39.         .Columns(2).Cells(1) = "Start Time"
40.         For i = 3 To .Columns.Count
41.             If i Mod 2 <> 0 Then
42.                 .Columns(i).Cells(1) = "Used"
43.             Else
44.                 .Columns(i).Cells(1) = "Not Used"
45.             End If
46.         Next i
47.     End With
48.     Charts.Add
49.     ActiveChart.Name = "TimeChart"
50.     ActiveChart.ChartWizard Source:=Sheets("ChartData").Range("A1").CurrentRegion, _
51.             Gallery:=xlBar, format:=3, PlotBy:=xlColumns, CategoryLabels :=1, SeriesLabels:=1,
                HasLegend:=2
52.
53.
54.     For Each oSeries In ActiveChart.SeriesCollection
55.         If oSeries.PlotOrder Mod 2 <> 0 Then
56.             oSeries.Border.LineStyle = xlNone
57.             oSeries.Interior.ColorIndex = xlNone
58.         Else
59.             oSeries.Interior.ColorIndex = 5
60.         End If
61.     Next oSeries
62.
63.     With ActiveChart.Axes(xlValue)
64.         .MajorUnit = 0.0416666666
65.         .TickLabels.NumberFormat = "hh"
66.         .HasMajorGridlines = True
67.     End With
68.
69.     With ActiveChart.Axes(xlCategory)
70.         .ReversePlotOrder = True
71.     End With
72.
73.     With ActiveChart
74.         .HasTitle = True
75.         .ChartTitle.Characters.Text = "Raum Belegung"
76.     End With
77. End Sub
```