Quelle: Einstieg in Visual Basic 2008 von Thomas Theis
Galileo Computing / <openbook> / Einstieg Visual Basic

1 Einführung

Visual Basic 2008 ist der Nachfolger von Visual Basic 2005 und damit eine weitere Version von Visual Basic .NET. Hierbei handelt es sich um eine objektorientierte Programmiersprache.

1.1 Aufbau dieses Buches

Beispiele

Dieses Buch vermittelt zunächst einen einfachen Einstieg in die Programmierung mit Visual Basic 2008. Die Bearbeitung der Beispiele und das selbstständige Lösen der vorliegenden Übungsaufgaben helfen dabei. Der Leser hat dadurch schnell Erfolgserlebnisse, die ihn zum Weitermachen motivieren. In späteren Kapiteln werden auch die komplexen Themen vermittelt.

Grundlagen

Von Anfang an wird mit anschaulichen Windows-Anwendungen gearbeitet. Die Grundlagen der Programmiersprache und die Standardelemente einer Windows-Anwendung, wie sie der Leser schon von anderen Windows-Programmen kennt, werden gemeinsam vermittelt. Die Anschaulichkeit einer Windows-Anwendung hilft dabei, den eher theoretischen Hintergrund der Programmiersprache leichter zu verstehen.

Express Edition

Es wird die Visual Basic 2008 Express Edition eingesetzt. Diese freie Version von Visual Basic 2008 liegt dem Buch bei. Man kann sie auch bei Microsoft herunterladen. Sie muss dann innerhalb von dreißig Tagen bei Microsoft registriert werden.

Die Visual Basic 2008 Express Edition bietet eine komfortable Entwicklungsumgebung. Sie umfasst

- einen Editor zur Erstellung des Programmcodes,
- einen Compiler zur Erstellung der ausführbaren Programme,
- einen Debugger zur Fehlersuche und vieles mehr.

1.2 Mein erstes Windows-Programm

Anhand eines ersten Projekts werden die Schritte durchlaufen, die zur Erstellung eines einfachen Programms mithilfe von Visual Basic 2008 notwendig sind. Das Programm soll nach dem Aufruf zunächst wie folgt aussehen:

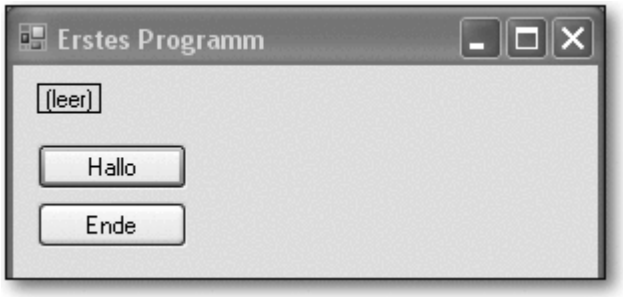

Abbildung 1.1 Erstes Programm, nach dem Aufruf

Nach Betätigung des Buttons Hallo soll sich der Text in der obersten Zeile verändern:

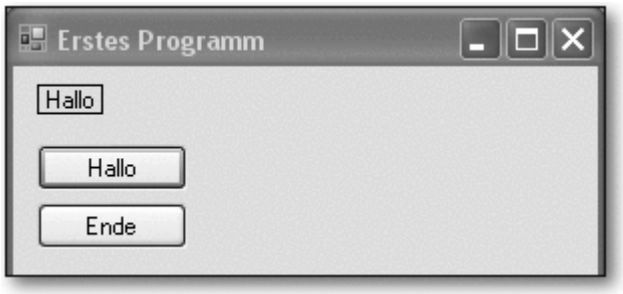

Abbildung 1.2 Nach Betätigung des Buttons »Hallo«

1.3 Visual Basic 2008-Entwicklungsumgebung

1.3.1 Ein neues Projekt

Nach dem Aufruf des Programms Visual Basic 2008 Express Edition muss zur Erstellung eines neuen Projekts der Menüpunkt Datei • Neu • Projekt • Windows Forms-Anwendung ausgewählt werden. Es erscheinen einige Elemente der Entwicklungsumgebung. Folgende Elemente sind besonders wichtig:

Form

- Das Benutzerformular (Form) enthält die Oberfläche für den Benutzer des Programms (siehe Abbildung 1.3).

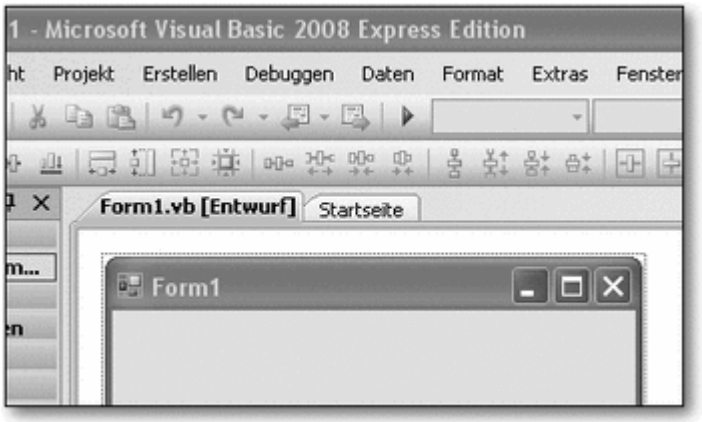

Abbildung 1.3 Benutzerformular

Toolbox

- Die Werkzeugsammlung (Toolbox) enthält die Steuerelemente für den Benutzer, mit denen er den Ablauf des Programms steuern kann. Sie werden vom Programm-Entwickler in das Formular eingefügt (siehe Abbildung 1.4).

Abbildung 1.4 Toolbox, Alle Steuerelemente

Eigenschaftenfenster

- Das Eigenschaftenfenster (Properties Window) dient zum Anzeigen und Ändern der Eigenschaften von Steuerelementen innerhalb des Formulars durch den Programm-Entwickler (siehe Abbildung 1.5).

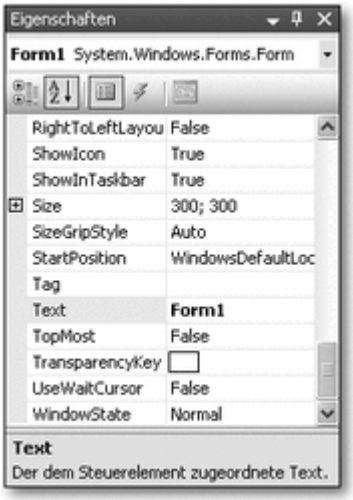

Abbildung 1.5 Eigenschaftenfenster

Projektmappen-Explorer

- Der Projektmappen-Explorer (Solution Explorer) zeigt das geöffnete Projekt und die darin vorhandenen Elemente (siehe Abbildung 1.6).

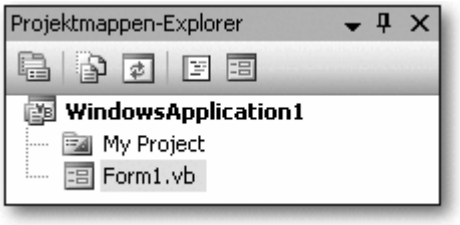

Abbildung 1.6 Projektmappen-Explorer

Sollte eines der letzten drei Elemente einmal nicht sichtbar sein, so kann es über das Menü Ansicht eingeblendet werden.

Zunächst werden nur einfache Programme mit wenigen Elementen geschrieben, daher wird der Projektmappen-Explorer noch nicht benötigt. Es empfiehlt sich, das Eigenschaftenfenster nach oben zu vergrößern.

Englische Terminologie

Im vorliegenden Buch werden sowohl die deutschen als auch die englischen Begriffe für die Elemente der Entwicklungsumgebung genutzt. Dies erleichtert den Umgang mit den Begriffen, mit den verschiedenen Versionen und mit der Visual Basic-Hilfe.

1.3.2 Einfügen von Steuerelementen

Label, Button

Zunächst sollen drei Steuerelemente in das Formular eingefügt werden: ein Bezeichnungsfeld (Label) und zwei Befehlsschaltflächen (Buttons). Ein Bezeichnungsfeld dient im Allgemeinen dazu, feste oder veränderliche Texte auf der Benutzeroberfläche anzuzeigen. In diesem Programm soll das Label einen Text anzeigen. Ein Button dient zum Starten bestimmter Programmteile oder, allgemeiner ausgedrückt, zum Auslösen von Ereignissen. In diesem Programm sollen die Buttons dazu dienen, den Text anzuzeigen bzw. das Programm zu beenden.

Allgemeine Steuerelemente

Um ein Steuerelement einzufügen, zieht man es mithilfe der Maus von der Toolbox an die gewünschte Stelle im Formular. Alle Steuerelemente finden sich in der Toolbox unter Alle Windows Forms. Übersichtlicher ist der Zugriff über Allgemeine Steuerelemente (Common Controls):

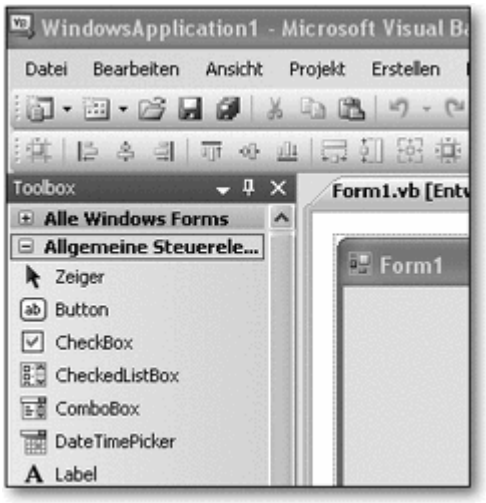

Abbildung 1.7 Toolbox, Allgemeine Steuerelemente

Steuerelement auswählen

Ein Doppelklick auf ein Steuerelement in der Toolbox fügt es ebenfalls in die Form ein. Anschließend können Ort und Größe noch verändert werden. Dazu muss das betreffende Steuerelement vorher durch Anklicken ausgewählt werden. Ein überflüssiges Steuerelement kann durch Auswählen und Drücken der Taste Entf entfernt werden.

Die Größe und andere Eigenschaften des Formulars selbst können auch verändert werden. Dazu muss man es vorher durch Anklicken auf einer freien Stelle auswählen.

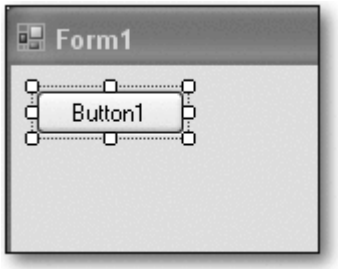

Abbildung 1.8 Ausgewählter Button

1.3.3 Arbeiten mit dem Eigenschaftenfenster

Die eingefügten Steuerelemente haben zunächst einheitliche Namen und Aufschriften, diese sollten allerdings zur einfacheren Programmentwicklung geändert werden. Es haben sich bestimmte Namenskonventionen eingebürgert, die die Lesbarkeit erleichtern. Diese Namen beinhalten den Typ (mit drei Buchstaben abgekürzt) und die Aufgabe des Steuerelements (mit großem Anfangsbuchstaben).

cmd, txt, lbl, ...

Ein Button (eigentlich: Command Button), der die Anzeige der Zeit auslösen soll, wird beispielsweise mit cmdZeit bezeichnet. Weitere Vorsilben sind txt (Textfeld/Text Box), lbl (Bezeichnungsfeld/Label), opt (Optionsschaltfläche/Option Button), frm (Formular/Form) und chk (Kontrollkästchen/Check Box).

Zur Änderung des Namens eines Steuerelementes muss es zunächst ausgewählt werden. Die Auswahl kann entweder durch Anklicken des Steuerelements auf dem Formular oder durch Auswahl aus der Liste am oberen Ende des Eigenschaftenfensters geschehen.

Eigenschaftenfenster

Im Eigenschaftenfenster werden alle Eigenschaften des ausgewählten Steuerelements angezeigt. Die Liste ist zweispaltig: In der linken Spalte steht der Name der Eigenschaft, in der rechten Spalte ihr aktueller Wert. Die Eigenschaft »(Name)« steht relativ am Anfang der Liste der Eigenschaften. Die betreffende Zeile wird durch Anklicken ausgewählt und der neue Name wird eingegeben. Nach Bestätigung mit der Taste ↵ ist die Eigenschaft geändert.

Abbildung 1.9 Button, nach der Namensänderung

Eigenschaft »Text«

Die Aufschrift von Buttons, Labels und Formularen ist in der Eigenschaft Text angegeben. Sobald diese Eigenschaft verändert wird, erscheint die veränderte Aufschrift in dem betreffenden Steuerelement. Auch der Name und die Aufschrift des Formulars sollten geändert werden. Im Folgenden sind die gewünschten Eigenschaften für die Steuerelemente dieses Programms in Tabellenform angegeben:

Typ	Eigenschaft	Einstellung
Formular	Name	frm0101
	Text	Erstes Programm
Button	Name	cmdHallo
	Text	Hallo
Button	Name	cmdEnde
	Text	Ende
Label	Name	lblAnzeige
	Text	(leer)
	BorderStyle	Fixed Single

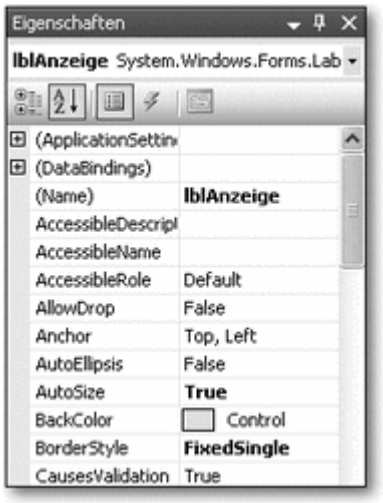

Abbildung 1.10 Label, nach der Änderung von Name und BorderStyle

Startzustand

Zu diesem Zeitpunkt legt man den Startzustand fest, also die Eigenschaften, die die Steuerelemente zu Beginn des Programms bzw. eventuell während des gesamten Programms haben sollen. Viele Eigenschaften kann man auch während der Laufzeit des Programms durch den Programmcode verändern lassen.

Bei einem Label ergibt die Einstellung der Eigenschaft Border Style auf Fixed Single einen Rahmen. Zur Änderung auf Fixed Single muss die Liste bei der Eigenschaft aufgeklappt und der betreffende Eintrag ausgewählt werden. Zur Änderung einiger Eigenschaften muss ein Dialogfeld aufgerufen werden.

Im Label soll zunächst der Text »(leer)« erscheinen. Hierzu muss der vorhandene Text durch Anklicken ausgewählt und geändert werden.

Liste der Steuerelemente

Man findet alle in diesem Formular vorhandenen Steuerelemente in der Liste, die sich am oberen Ende des Eigenschaftenfensters öffnen lässt. Dabei zeigt sich ein Vorteil der einheitlichen Namensvergabe: Die Steuerelemente des gleichen Typs stehen direkt untereinander.

1.3.4 Speichern eines Projekts

Alles speichern

Die Daten eines Visual Basic-Projekts werden in verschiedenen Dateien gespeichert. Zum Speichern des gesamten Projekts wird der Menüpunkt Datei • Alle Speichern verwendet. Im Dialogfeld Projekt speichern werden als Name und Projektmappenname WindowsApplication1 vorgeschlagen. Beide Namen sollten in p0101 geändert werden. Das gleiche Dialogfeld erscheint, wenn man das Projekt noch nicht gespeichert hat und den Menüpunkt Datei • Projekt schließen wählt.

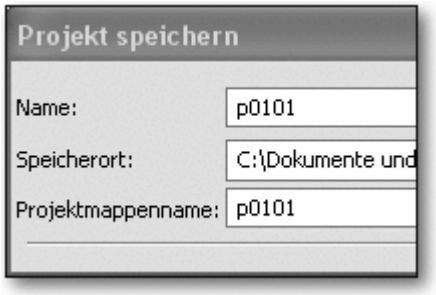

Abbildung 1.11 Projekt p0101 speichern

p0101

Die in diesem Skript angegebenen Namen dienen als Empfehlung, um die eindeutige Orientierung und das spätere Auffinden von alten Programmen zu erleichtern. Alle Namen beginnen mit »p« (für Projekt), es folgen zwei Ziffern für die Kapitelnummer (mit führender 0) und zwei Ziffern für die fortlaufende Nummerierung (mit führender 0) innerhalb des Kapitels. Das Gleiche gilt für die Formulare: Das zu diesem Projekt gehörende Formular trägt bereits den Namen frm0101.

1.3.5 Das Codefenster

Ereignis

Der Ablauf eines Visual Basic-Programms wird im Wesentlichen durch das Auslösen von Ereignissen durch den Benutzer gesteuert. Er löst z. B. die Anzeige des Texts »Hallo« aus, indem er auf den Button Hallo klickt. Der Entwickler muss dafür sorgen, dass aufgrund dieses Ereignisses der gewünschte Text angezeigt wird. Zu diesem Zweck schreibt er Programmcode und ordnet diesen Code dem Ereignis zu. Der Code wird in einer Ereignisprozedur abgelegt.

Ereignisprozedur

Zum Schreiben einer Ereignisprozedur führt man am besten einen Doppelklick auf das betreffende Steuerelement aus. Es erscheint das Codefenster. Zwischen der Formularansicht und der Codeansicht kann man anschließend über die Menüpunkte Ansicht • Code bzw. Ansicht • Designer hin- und herschalten. Dies ist auch über die Registerkarten oberhalb des Formulars bzw. des Codefensters möglich:

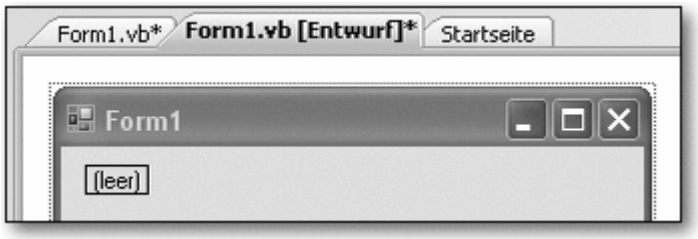

Abbildung 1.12 Registerkarten

Nach erfolgtem Doppelklick auf den Button Hallo erscheinen im Codefenster folgende Einträge:

```
Public Class frm0101
   Private Sub cmdHallo_Click(ByVal sender As System.Object, _
           ByVal e As System.EventArgs) Handles cmdHallo.Click
   End Sub
End Class
```

Innerhalb der Ereignisprozedur ist der Platz für den eigenen Programmcode.

Class

VB.NET ist eine objektorientierte Sprache. Ein wichtiges Element objektorientierter Sprachen sind Klassen. Alle Elemente des aktuellen Formulars frm0101 stehen innerhalb der Klasse frm0101 (zwischen Public Class und End Class). Auf die Einzelheiten der Objektorientierung wird zu einem späteren Zeitpunkt eingegangen, da dies hier noch nicht notwendig ist und eher verwirren würde.

Sub

Der Programmcode der Ereignisprozedur steht später zwischen Private Sub und End Sub. Der Name der Prozedur besteht aus den zwei Teilen »Name des Steuerelements« und »ausgelöstes Ereignis«.

Code über mehrere Zeilen

Das Zeichen _ (= Unterstrich) am Ende einer Zeile zeigt in Visual Basic an, dass die aktuelle Zeile in der nächsten Zeile fortgesetzt wird. Dies ist hier am Ende der Zeile Private Sub ... der Fall. Aus Platzgründen werden in diesem Buch häufig längere Codezeilen auf mehrere Zeilen verteilt. Diesem Beispiel muss der Leser beim Erstellen des eigenen Codes nicht notwendig folgen.

Am oberen Ende des Codefensters kann auf der linken Seite eine Liste der Steuerelemente aufgeklappt werden, um die Ereignisprozedur eines anderen Steuerelements anzusehen oder zu verändern.

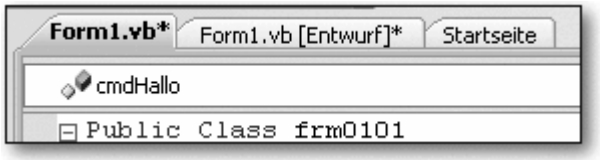

Abbildung 1.13 Liste der Steuerelemente, links

Auf der rechten Seite kann eine Liste der Ereignisse bzw. Ereignisprozeduren aufgeklappt werden, die mit diesem Steuerelement verbunden werden können. Der anfänglich ausgeführte Doppelklick führt immer zu dem Ereignis, das am häufigsten mit dem betreffenden Steuerelement verbunden wird. Dies ist beim Button natürlich das Ereignis Click.

Abbildung 1.14 Liste der Ereignisse, rechts

1.3.6 Schreiben von Programmcode

In der Prozedur cmdHallo_Click() soll eine Befehlszeile eingefügt werden, die wie folgt aussieht:

```
Private Sub cmdHallo_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdHallo.Click
   lblAnzeige.Text = "Hallo"
End Sub
```

Der Text muss in Anführungszeichen gesetzt werden, da Visual Basic sonst annimmt, dass es sich um eine Variable mit dem Namen Hallo handelt.

Anweisung

Der Inhalt einer Prozedur setzt sich aus einzelnen Anweisungen zusammen, die nacheinander ausgeführt werden. Die vorliegende Prozedur enthält nur eine Anweisung; in ihr wird mithilfe des Gleichheitszeichens eine Zuweisung durchgeführt.

Zuweisung

Bei einer Zuweisung wird der Ausdruck rechts vom Gleichheitszeichen ausgewertet und der Variablen, der Objekt-Eigenschaft oder der Steuerelement-Eigenschaft links vom Gleichheitszeichen zugewiesen. Die Zeichenkette »Hallo« wird der Eigenschaft Text des Steuerelements lblAnzeige mithilfe der Schreibweise Steuerelement.Eigenschaft = Wert zugewiesen. Dies führt zur Anzeige des Werts.

Nach dem Wechsel auf die Formularansicht kann das nächste Steuerelement ausgewählt werden, für das eine Ereignisprozedur geschrieben werden soll.

Code editieren

Innerhalb des Codefensters kann Text mit den gängigen Methoden der Textverarbeitung editiert, kopiert, verschoben und gelöscht werden.

Syntaxfehler

Sollte man bereits bei der Eingabe des Programmcodes Syntaxfehler gemacht haben, so wird dies angezeigt. Man sollte den Code unmittelbar entsprechend korrigieren.

In der Ereignisprozedur cmdEnde_Click() soll der folgende Code stehen:

```
Private Sub cmdEnde_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdEnde.Click
   Me.Close()
End Sub
```

Me

Die Methode Close() dient zum Schließen eines Formulars. Da es sich um das einzige Formular dieses Projekts handelt, wird dadurch das Programm beendet und die gesamte Windows-Anwendung geschlossen. Mit Me wird das aktuelle Objekt einer Klasse bezeichnet. Da wir uns innerhalb der Klasse für das aktuelle Formular befinden, bezieht sich Me auf dieses Formular.

Dies waren Beispiele zur Änderung der Eigenschaften eines Steuerelements zur Laufzeit des Programms durch Programmcode. Sie erinnern sich: Zu Beginn hatten wir die Start-Eigenschaften der Steuerelemente im Eigenschaftenfenster eingestellt.

1.3.7 Kommentare

Einfaches Hochkomma

Bei längeren Programmen mit vielen Anweisungen gehört es zum guten Programmierstil, Kommentarzeilen zu schreiben. In diesen Zeilen werden einzelne Anweisungen oder auch längere Blöcke von Anweisungen erläutert, damit man selber oder auch ein anderer Programmierer sie später leich-

ter versteht. Eine Kommentarzeile beginnt mit einem einfachen Hochkomma. Alle Zeichen bis zum Ende der Zeile werden als Kommentar angesehen und folglich nicht übersetzt oder ausgeführt.

Der folgende Programmcode wurde um eine Kommentarzeile ergänzt:

```
Private Sub cmdEnde_Click( ... ) Handles ...
   ' Schließt die Anwendung
   Me.Close()
End Sub
```

Code auskommentieren

Ein kleiner Trick: Sollen bestimmte Programmzeilen für einen Test des Programms kurzfristig nicht ausgeführt werden, so kann man sie »auskommentieren«, indem man das Hochkomma vor die betreffenden Zeilen setzt. Dies geht sehr schnell über das entsprechende Symbol in der Symbolleiste. Rechts daneben befindet sich das Symbol, das die Auskommentierung nach dem Test wieder rückgängig macht.

Abbildung 1.15 Kommentar ein/aus

Click(...) Handles ...

Ein Hinweis: Der Kopf der Prozedur cmdEnde_Click() wurde aus Gründen der Übersichtlichkeit in verkürzter Form abgebildet. Dies wird bei den meisten nachfolgenden Beispielen ebenfalls so sein, außer wenn es genau auf die Inhalte des Prozedurkopfs ankommt.

1.3.8 Starten, Ausführen und Beenden des Programms

Programm starten

Nach dem Einfügen der Steuerelemente und dem Erstellen der Ereignisprozeduren ist das Programm fertig und kann gestartet werden. Dazu wird der Start-Button in der Symbolleiste (dreieckiger Pfeil nach rechts) betätigt. Alternativ startet man das Programm über die Funktionstaste F5 oder den Menüpunkt Debuggen • Starten. Das Formular erscheint, das Betätigen der Buttons führt zum programmierten Ergebnis.

Programm beenden

Zur regulären Beendigung eines Programms ist der Button mit der Aufschrift Ende vorgesehen. Möchte man ein Programm während des Verlaufs abbrechen, kann man auch den End-Button in der Symbolleiste (Quadrat) betätigen. Außerdem kann man, sofern vorhanden, den aus vielen Windows-Anwendungen bekannten Button X oben rechts betätigen; dieser erscheint nur, wenn die Eigenschaft ControlBox des Formulars auf True steht.

Fehler

Tritt während der Ausführung eines Programms ein Fehler auf, so wird dies angezeigt und das Codefenster zeigt die entsprechende Ereignisprozedur sowie die fehlerhafte Zeile an. Man beendet das Programm, korrigiert den Code und startet das Programm wieder.

Programm testen

Es wird empfohlen, das Programm bereits während der Entwicklung mehrmals durch Aufruf zu testen und nicht erst, wenn das Programm vollständig erstellt worden ist. Geeignete Zeitpunkte sind zum Beispiel:

- nach dem Einfügen der Steuerelemente und dem Zuweisen der Eigenschaften, die sie zu Programmbeginn haben sollen
- nach dem Erstellen jeder Ereignisprozedur

1.3.9 Ausführbares Programm

.exe-Datei

Nach erfolgreichem Test des Programms könnte man auch die ausführbare Datei (.exe-Datei) außerhalb der Entwicklungsumgebung aufrufen. Hat man an den Grundeinstellungen nichts verändert und die vorgeschlagenen Namen verwendet, so findet sich die zugehörige .exe-Datei des aktuellen Projekts im Verzeichnis Eigene Dateien\Visual Studio 2008\Projects\p0101\p0101\bin\Debug. Das Programm kann also im Windows-Explorer direkt über Doppelklick gestartet werden.

Die Weitergabe eines eigenen Windows-Programms auf einen anderen PC ist etwas aufwendiger. Der Vorgang wird im Anhang beschrieben. Im Folgenden soll jedoch zunächst die Windows-Programmierung erlernt werden.

1.3.10 Projekt schließen, Projekt öffnen

Projekt schließen

Man kann ein Projekt schließen über den Menüpunkt Datei • Projekt schließen. Falls man Veränderungen vorgenommen hat, wird man gefragt, ob man diese Änderungen speichern möchte.

Möchte man die Projektdaten sicherheitshalber zwischendurch speichern, so ist dies über den Menüpunkt Datei • Alle speichern möglich.

Projekt öffnen

Zum Öffnen eines vorhandenen Projekts wählt man den Menüpunkt Datei • Projekt öffnen. Im darauf folgenden Dialogfeld Projekt öffnen wählt man zunächst das gewünschte Projektverzeichnis aus und anschließend die gleichnamige Datei mit der Endung .sln.

1.3.11 Übung

Übung p0102:

Erstellen Sie ein Windows-Programm, das zwei Buttons und ein Label beinhaltet. Bei Betätigung des ersten Buttons erscheint im Label Ihr Name wie im folgenden Satz: »Dieses Programm wurde erstellt von Bodo Basic«. Bei Betätigung des zweiten Buttons wird das Programm beendet. Namensvorschläge: Projektname p0102, Buttons cmdMyName und cmdEnde, Label lblMyName.

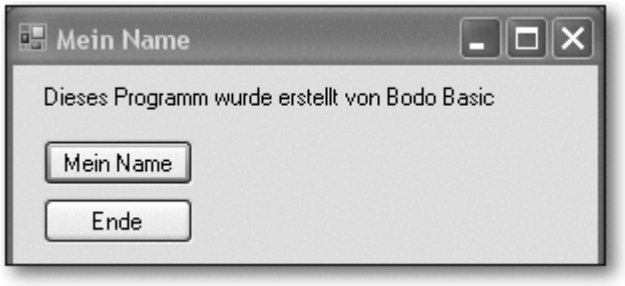

Abbildung 1.16 Übung p0102

1.4 Arbeiten mit Steuerelementen

1.4.1 Steuerelemente formatieren

Hilfslinien

Zur besseren Anordnung der Steuerelemente auf dem Formular können Sie sie mit der Maus nach Augenmaß anordnen. Beim Verschieben der Steuerelemente erscheinen automatisch Hilfslinien, falls das aktuelle Element horizontal oder vertikal parallel zu einem anderen Element steht.

Mehrere Steuerelemente markieren

Weitere Möglichkeiten bieten die Menüpunkte im Menü Format. In beiden Fällen müssen vorher mehrere Steuerelemente markiert werden. Dies geschieht

- entweder durch Umrahmung der Elemente mit einem Rechteck, nachdem man zuvor das Steuerelement Zeiger ausgewählt hat,

- oder durch Mehrfachauswahl, indem ab dem zweiten auszuwählenden Steuerelement die ⇧-Taste (wie für Großbuchstaben) oder die Strg-Taste gedrückt wird.

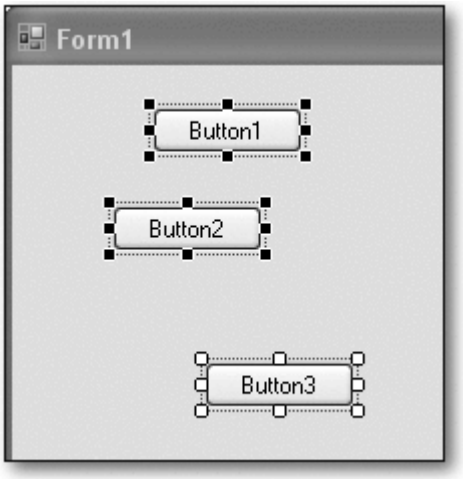

Abbildung 1.17 Mehrere markierte Elemente

Menü »Format«

Über das Menü Format hat man anschließend folgende Möglichkeiten zur Anpassung der Steuerelemente:

- Die ausgewählten Steuerelemente können horizontal oder vertikal zueinander ausgerichtet werden (Menü Format • Ausrichten).

- Die horizontalen und/oder vertikalen Dimensionen der ausgewählten Steuerelemente können angeglichen werden (Menü Format • Größe angleichen).

  Einheitliche Abstände

- Die horizontalen und vertikalen Abstände zwischen den ausgewählten Steuerelementen können angeglichen, vergrößert, verkleinert oder entfernt werden (Menü Format • Horizontaler Abstand/Vertikaler Abstand).

- Die Steuerelemente können horizontal oder vertikal innerhalb des Formulars zentriert werden (Menü Format • auf Formular zentrieren).

- Sollten sich die Steuerelemente teilweise überlappen, kann man einzelne Steuerelemente in den Vorder- bzw. Hintergrund schieben (Menü Format • Reihenfolge).

- Man kann alle Steuerelemente gleichzeitig gegen versehentliches Verschieben absichern (Menü Format • Sperren). Diese Sperrung gilt nur während der Entwicklung des Programms.

Abbildung 1.18 zeigt ein Formular mit drei Buttons, die alle links ausgerichtet sind und im gleichen vertikalen Abstand voneinander stehen:

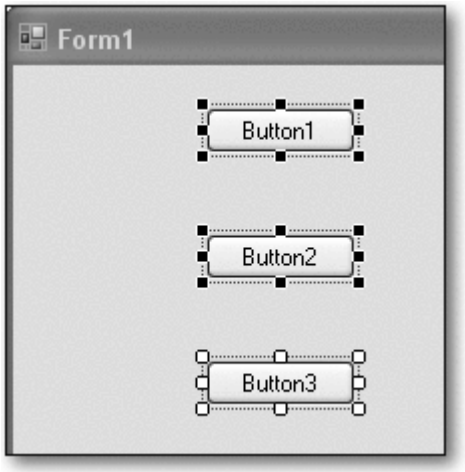

Abbildung 1.18 Nach der Formatierung

Übung:

Laden Sie das erste Projekt aus dem vorherigen Abschnitt (p0101), markieren Sie darin mehrere Steuerelemente, und testen Sie die einzelnen Möglichkeiten des Format-Menüs.

1.4.2 Steuerelemente kopieren

Steuerelemente kopieren

Zur schnelleren Erzeugung eines Projekts können vorhandene Steuerelemente einschließlich aller ihrer Eigenschaften kopiert werden. Markieren Sie hierzu die gewünschten Steuerelemente und kopieren Sie sie

- entweder über das Menü Bearbeiten • Kopieren und das Menü Bearbeiten • Einfügen (Menü Edit • Copy und Menü Edit • Paste)
- oder mit den Tasten Strg + C und Strg + V.

Anschließend sollten die neu erzeugten Steuerelemente direkt umbenannt und an der gewünschten Stelle angeordnet werden.

Übung:

Laden Sie das erste Projekt aus dem vorherigen Abschnitt (p0101), und kopieren Sie einzelne Steuerelemente. Kontrollieren Sie anschließend die Liste der vorhandenen Steuerelemente in der Eigenschaften-Tabelle auf einheitliche Namensgebung.

1.4.3 Eigenschaften zur Laufzeit ändern

Size, Location

Steuerelemente haben die Eigenschaften Size (mit den Komponenten Width und Height) und Location (mit den Komponenten X und Y) zur Angabe von Größe und Position. X und Y geben die Ko-

ordinaten der oberen linken Ecke des Steuerelements an, gemessen von der oberen linken Ecke des umgebenden Elements (meist das Formular). Alle Werte werden in Pixeln gemessen.

Alle diese Eigenschaften können sowohl während der Entwicklungszeit als auch während der Laufzeit eines Projekts verändert werden. Zur Änderung während der Entwicklungszeit kann die Eigenschaft wie gewohnt im Eigenschaftenfenster eingegeben werden. Als Beispiel für Änderungen während der Laufzeit soll das folgende Programm dienen:

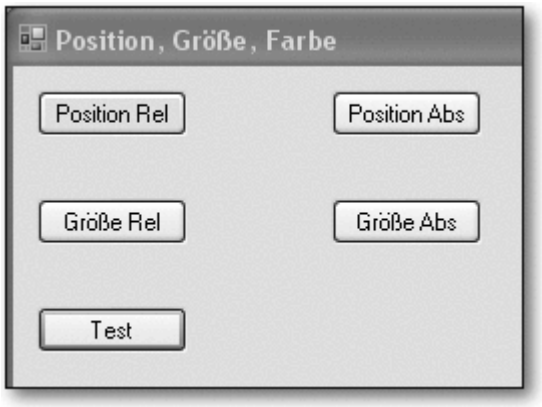

Abbildung 1.19 Position und Größe bestimmen

Der Programmcode:

```
Public Class frm0103
   Private Sub cmdPositionRel_Click( ... ) Handles ...
      cmdTest.Location = New Point(cmdTest.Location.X + 20, _
         cmdTest.Location.Y)
   End Sub

   Private Sub cmdPositionAbs_Click( ... ) Handles ...
      cmdTest.Location = New Point(100, 200)
   End Sub

   Private Sub cmdGrößeRel_Click( ... ) Handles ...
      cmdTest.Size = New Point(cmdTest.Size.Width + 20, _
         cmdTest.Size.Height)
   End Sub

   Private Sub cmdGrößeAbs_Click( ... ) Handles ...
      cmdTest.Size = New Point(50, 100)
   End Sub
End Class
```

Zur Erläuterung:

- Das Formular enthält fünf Buttons. Die oberen vier Buttons dienen zur Veränderung von Position und Größe des fünften Buttons.

- Die Position eines Elements kann relativ zur aktuellen Position oder auf absolute Werte eingestellt werden. Das Gleiche gilt für die Größe eines Elements.

- Bei beiden Angaben handelt es sich um Wertepaare (X/Y bzw. Breite/Höhe).

  New Point

- Zur Einstellung dient die Struktur Point. Ein Objekt dieser Struktur liefert ein Wertepaar. In diesem Programm wird mit New jeweils ein neues Objekt der Struktur Point erzeugt, um das Wertepaar bereitzustellen.

  X, Y

- Bei Betätigung des Buttons Position Abs wird die Position des fünften Buttons auf die Werte X=100 und Y=200 gestellt, gemessen von der linken oberen Ecke des Formulars.

- Bei Betätigung des Buttons Position Rel wird die Position des fünften Buttons auf die Werte X = cmdTest.Location.X + 20 und Y = cmdTest.Location.Y gestellt. Bei X wird also der alte Wert der Komponente X um 20 erhöht, das Element bewegt sich nach rechts. Bei Y wird der alte Wert der Komponente Y nicht verändert, das Element bewegt sich nicht nach oben oder unten.

  Width, Height

- Bei Betätigung des Buttons Größe Abs wird die Größe des fünften Buttons auf die Werte Width = 50 und Height = 100 gestellt.

- Bei Betätigung des Buttons Größe Rel wird die Größe des fünften Buttons auf die Werte Width = cmdTest.Size.Width + 20 und Height = cmdTest.Size.Height gestellt. Bei Width wird also der alte Wert der Komponente Width um 20 erhöht, das Element wird breiter. Bei Height wird der alte Wert der Komponente Height nicht verändert, das Element verändert seine Höhe nicht.

Nach einigen Klicks sieht das Formular wie folgt aus:

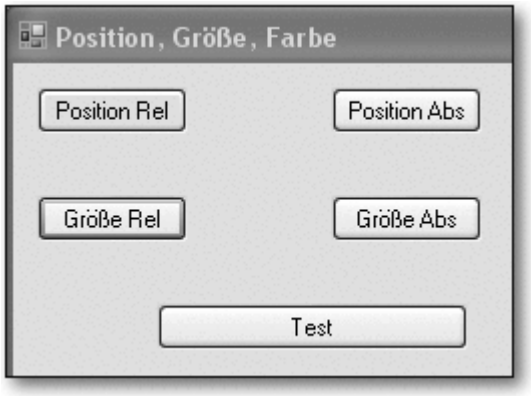

Abbildung 1.20 Veränderung von Eigenschaften zur Laufzeit

1.4.4 Vergabe und Verwendung von Namen

Beachten Sie in allen Programmen, dass jedes Steuerelement seinen eigenen, eindeutigen Namen hat und immer mit diesem Namen angesprochen werden muss. Es passiert erfahrungsgemäß besonders am Anfang häufig, dass ein Programm nicht zum gewünschten Erfolg führt, weil ein nicht vorhandener Name verwendet wurde. In diesem Zusammenhang weise ich noch einmal auf die Namenskonventionen hin:

- Buttons sollten Namen wie z. B. cmdEnde, cmdAnzeigen, cmdBerechnen usw. haben

- Labels sollten Namen wie z. B. lblAnzeige, lblName, lblUhrzeit, lblBeginnDatum haben

Diese Namen liefern eine eindeutige Information über Typ und Funktion des Steuerelements. Falls Sie beim Schreiben von Programmcode anschließend diese Namen vollständig in Kleinbuchstaben eingeben, werden sie nach Verlassen der Zeile automatisch korrigiert. Daran können Sie schnell erkennen, ob Sie ein vorhandenes Steuerelement verwendet haben.

1.4.5 Verknüpfung von Texten, mehrzeilige Texte

& und vbCrLf

Es können mehrere Texte in einer Ausgabe mithilfe des Zeichens & miteinander verknüpft werden. Falls man eine mehrzeilige Ausgabe wünscht, so muss zusätzlich ein Zeilenvorschub mithilfe der integrierten Visual Basic-Konstante vbCrLf eingegeben werden. Sollte eine Zeile bei der Eingabe im Codefenster zu lang werden, so kann man sie mithilfe des Zeichens _ am Ende der Zeile teilen, wie ich dies auch bei den Codebeispielen in diesem Buch handhabe.

Nachfolgend wird das Beispiel p0103 ergänzt um ein Label, in dem die aktuelle Position und Größe des Buttons angezeigt werden. Dies soll nach Betätigung des Buttons Anzeige geschehen:

```
Public Class frm0103
[ ... ]
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      lblAnzeige.Text = _
         "Position: X: " & cmdTest.Location.X _
         &  ", Y: " & cmdTest.Location.Y & vbCrLf _
         & "Größe: Breite: " & cmdTest.Size.Width _
         & ", Höhe: " & cmdTest.Size.Height
   End Sub
End Class
```

Nach einigen Klicks und der Betätigung des Buttons Anzeige sieht das Formular wie folgt aus:

Abbildung 1.21 Anzeige der Eigenschaften

1.4.6 Eigenschaft BackColor, Farben allgemein

BackColor

Die Hintergrundfarbe eines Steuerelements wird mit der Eigenschaft BackColor festgelegt. Dabei kann die Farbe zur Entwicklungszeit leicht mithilfe einer Farbpalette oder aus Systemfarben ausgewählt werden.

Color

Hintergrundfarben und andere Farben können auch zur Laufzeit eingestellt werden, dabei bedient man sich Farbwerten. Diese Farbwerte kann man über die Struktur Color auswählen.

Ein Beispiel, ebenfalls in p0103:

```
Public Class frm0103
[ ... ]
   Private Sub cmdFarbe_Click( ... ) Handles ...
      Me.BackColor = Color.Yellow
      lblAnzeige.BackColor = Color.FromArgb(192, 255, 0)
   End Sub
End Class
```

Zur Erläuterung:

- Diese Struktur bietet vordefinierte Farbnamen als Eigenschaften, zum Beispiel Yellow. Der Wert kann der Eigenschaft BackColor des Steuerelements zugewiesen werden, hier ist dies das Formular selbst (Me).

  FromArgb()

- Außerdem bietet die Struktur die Methode FromArgb(). Diese kann auf vier verschiedene Arten aufgerufen werden. Eine dieser vier Arten erwartet genau drei Parameter, nämlich die Werte für Rot, Grün und Blau, jeweils zwischen 0 und 255.

Abbildung 1.22 Nach Änderung der Eigenschaft »Farbe«

2 Grundlagen

2.1 Variablen und Datentypen

Variablen dienen zur vorübergehenden Speicherung von Daten, die sich zur Laufzeit eines Programms ändern können. Eine Variable besitzt einen eindeutigen Namen, unter dem sie angesprochen werden kann.

2.1.1 Namen, Werte

Namensregeln

Für die Namen von Variablen gelten in Visual Basic die folgenden Regeln:

- Sie beginnen mit einem Buchstaben.

- Sie können nur aus Buchstaben, Zahlen und einigen wenigen Sonderzeichen (z. B. dem Unterstrich _) bestehen.

- Innerhalb eines Gültigkeitsbereichs darf es keine zwei Variablen mit dem gleichen Namen geben (siehe Abschnitt 2.1.4, »Gültigkeitsbereich«).

Variablen erhalten ihre Werte durch Zuweisung per Gleichheitszeichen. Einer Variablen sollte vor ihrer ersten Benutzung ein Wert zugewiesen werden. Dadurch werden Programme eindeutiger, lesbarer und fehlerfreier.

2.1.2 Deklarationen

Neben dem Namen besitzt jede Variable einen Datentyp, der die Art der Information bestimmt, die gespeichert werden kann. Der Entwickler wählt den Datentyp danach aus, ob er Text, Zahlen ohne Nachkommastellen, Zahlen mit Nachkommastellen oder z. B. Datumsangaben speichern möchte.

Auswahl des Datentyps

Außerdem muss er sich noch Gedanken über die Größe des Bereichs machen, den die Zahl oder der Text annehmen könnte. Er sollte versuchen, den Datentyp mit dem geringsten Speicherbedarf zu verwenden; dieser gewährleistet gleichzeitig auch eine schnellere Verarbeitung. Im folgenden Abschnitt 2.1.3, »Datentypen«, finden Sie eine Liste der Datentypen mit Name, Speicherbedarf und Wertebereich.

Variablen sollten in Visual Basic immer mit einem Datentyp deklariert werden. Dies beugt Fehlern und unnötig hohem Speicherbedarf vor.

2.1.3 Datentypen

Die folgende Liste enthält die wichtigsten von Visual Basic unterstützten Datentypen sowie deren Speicherbedarf und Wertebereich.

- Datentyp Boolean, Speicherbedarf plattformabhängig, Wertebereich True oder False (»Wahr« oder »Falsch«)

- Datentyp Byte, Speicherbedarf 1 Byte, Wertebereich von 0 bis 255

- Datentyp Char, Speicherbedarf 2 Bytes, Wertebereich von 0 bis 65536

- Datentyp Date, Speicherbedarf 8 Bytes, Wertebereich vom 1. Januar 1 bis zum 31. Dezember 9999

Double

- Datentyp Double, Speicherbedarf 8 Bytes, Gleitkommazahl mit doppelter Genauigkeit, Wertebereich von –1,79769313486231570E+308 bis 4,94065645841246544E-324 für negative Werte und von 4,94065645841246544E-324 bis 1,79769313486231570E+308 für positive Werte

Integer

- Datentyp Integer, Speicherbedarf 4 Bytes, Wertebereich von –2.147.483.648 bis 2.147.483.647

- Datentyp Long, Speicherbedarf 4 Bytes, lange Ganzzahl, Wertebereich von –9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807

- Datentyp Object, Speicherbedarf 4–8 Bytes, Wertebereich beliebig

- Datentyp Short, Speicherbedarf 2 Bytes, Wertebereich von –32,768 bis 32,767

- Datentyp Single, Speicherbedarf 4 Bytes, Gleitkommazahl mit einfacher Genauigkeit; Wertebereich: 3,4028235E+38 bis –1,401298E-45 für negative Werte und 1,401298E-45 bis 3,4028235E+38 für positive Werte

String

- Datentyp String, Speicherbedarf plattformabhängig, Zeichenkette mit variabler Länge

- benutzerdefinierte Struktur, Speicherbedarf abhängig von den Elementen, jedes Element hat seinen eigenen Datentyp und damit seinen eigenen Wertebereich

Im folgenden Beispiel werden Variablen der wichtigsten Typen deklariert, mit Werten versehen und in einem Label angezeigt (p0201).

```
Public Class frm0201
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim Bo As Boolean
      Dim By As Byte
      Dim Ch As Char
      Dim Dt As Date
      Dim Db As Double
      Dim It As Integer
      Dim Lg As Long
      Dim Sh As Short
      Dim Sg As Single
      Dim St As String

      Bo = True
      By = 200
      Ch = "a"
      Dt = "15.12.2007"
      Db = 1 / 7
      It = 2000000000
      Lg = 3000000000
      Sh = 30000
      Sg = 1 / 7
      St = "Zeichenkette"

      lblAnzeige.Text = _
         "Boolean: " & Bo & vbCrLf & _
         "Byte: " & By & vbCrLf & _
         "Char: " & Ch & vbCrLf & _
         "Double: " & Db & vbCrLf & _
         "Date: " & Dt & vbCrLf & _
         "Integer: " & It & vbCrLf & _
         "Long: " & Lg & vbCrLf & _
         "Short: " & Sh & vbCrLf & _
         "Single: " & Sg & vbCrLf & _
```

```
         "String: " & St
   End Sub
End Class
```

Das Programm hat nach Betätigung des Buttons die folgende Ausgabe:

Abbildung 2.1 Wichtige Datentypen

Zur Erläuterung:

- Variablen werden mit Dim ... As ... deklariert.

  Wertebereich

- Bei den Zahlen-Datentypen führt eine Über- oder Unterschreitung des Wertebereichs zu einer Fehlermeldung.

  Genauigkeit

- Die Datentypen Single und Double für Zahlen mit Nachkommastellen unterscheiden sich in ihrer Genauigkeit.

- Werte für Zeichen, Zeichenketten und Datumsvariablen müssen in doppelten Anführungszeichen angegeben werden.

Mehrere Variablen des gleichen Typs können, durch Kommata getrennt, in einer Zeile deklariert werden (z. B. Dim x, y As Integer).

Übung p0202:

Schreiben Sie ein Programm, in dem Ihr Nachname, Vorname, Ihre Adresse, Ihr Geburtsdatum und Ihr Alter jeweils in Variablen eines geeigneten Datentyps gespeichert und anschließend wie folgt ausgegeben werden:

Abbildung 2.2 Übung p0202

2.1.4 Gültigkeitsbereich

Lokal

Variablen, die innerhalb einer Prozedur vereinbart wurden, haben ihre Gültigkeit nur in der Prozedur. Außerhalb der Prozedur sind sowohl Name als auch Wert unbekannt. Solche Variablen bezeichnet man auch als lokale Variablen. Sobald die Prozedur abgearbeitet wurde, steht der Wert auch nicht mehr zur Verfügung. Beim nächsten Aufruf der gleichen Prozedur werden diese Variablen neu deklariert und erhalten neue Werte.

Statisch

Anders verhält es sich mit statischen Variablen. Diese behalten ihren Wert, solange das Programm läuft. Ein wiederholter Aufruf der gleichen Prozedur kann auf den letzten gespeicherten Wert einer Variablen zugreifen. Eine statische Variable vereinbart man z. B. wie folgt: Static Sx As Integer.

Modulweit

Variablen, die außerhalb von Prozeduren vereinbart werden, sind modulweit gültig. Ihr Wert kann in jeder Prozedur gesetzt oder abgerufen werden und bleibt innerhalb des Moduls auch erhalten. Eine solche Variable vereinbart man wie eine lokale Variable, nur an anderer Stelle, z. B. wie folgt: Dim Mx As Integer.

Gibt es mehrere Variablen mit dem gleichen Namen, gelten folgende Regeln:

- Lokale Variablen mit gleichem Namen in der gleichen Prozedur sind nicht zulässig.

Ausblenden

- Eine modulweite Variable wird innerhalb einer Prozedur von einer lokalen Variablen mit dem gleichen Namen ausgeblendet.

- Eine anwendungsweite Variable wird innerhalb eines Moduls von einer modulweiten oder lokalen Variablen mit dem gleichen Namen ausgeblendet.

Im folgenden Beispiel werden lokale, statische und modulweite Variablen deklariert, in verschiedenen Prozeduren erhöht und ausgegeben (p0203).

```
Public Class frm0203
   Dim Mx As Integer

   Private Sub cmdAnzeigen1_Click( ... ) Handles ...
```

```
      Static Sx As Integer
      Dim x As Integer
      Sx = Sx + 1
      Mx = Mx + 1
      x = x + 1
      lblAnzeige.Text = "Sx: " & Sx _
         & "  x: " & x & "  MX: " & Mx
   End Sub

   Private Sub cmdAnzeigen2_Click( ... ) Handles ...
      Dim Mx As Integer
      Mx = Mx + 1
      lblAnzeige.Text = "MX: " & Mx
   End Sub
End Class
```

Abbildung 2.3 Statische und modulweite Variable

In der ersten Prozedur wird der Wert der statischen Variablen Sx und der modulweiten Variablen Mx bei jedem Aufruf erhöht. Die lokale Variable x wird immer wieder auf 1 gesetzt.

Abbildung 2.4 Lokale Variable

In der zweiten Prozedur blendet die lokale Variable Mx die gleichnamige modulweite Variable aus. Die lokale Variable wird immer wieder auf 1 gesetzt.

2.1.5 Konstanten

Konstanten repräsentieren Werte

Konstanten sind vordefinierte Werte, die während der Laufzeit nicht verändert werden können. Man gibt Konstanten im Allgemeinen aussagekräftige Namen, dadurch sind sie leichter zu behalten als die Werte, die sie repräsentieren. Konstanten werden an einer zentralen Stelle definiert und können an verschiedenen Stellen des Programms genutzt werden. Somit muss eine eventuelle Änderung einer Konstanten zur Entwurfszeit nur an einer Stelle erfolgen. Der Gültigkeitsbereich von Konstanten ist analog zum Gültigkeitsbereich von Variablen.

Integrierte Konstanten

Zu den Konstanten zählen auch die integrierten Konstanten, wie z. B. vbCrLf. Auch sie repräsentieren Zahlen, die aber nicht so einprägsam sind wie die Namen der Konstanten.

Im folgenden Beispiel werden mehrere Konstanten vereinbart und genutzt (p0204).

```
Public Class frm0204
   Const MaxWert = 75
   Const Eintrag = "Picture"

   Private Sub cmdKonstanten_Click( ... ) Handles ...
      Const MaxWert = 55
      Const MinWert = 5
      lblAnzeige.Text = (MaxWert - MinWert) / 2 & _
         vbCrLf & Eintrag
    End Sub
End Class
```

Zur Erläuterung:

- Die Konstanten MaxWert und Eintrag werden modulweit festgelegt.
- Innerhalb der Prozedur werden die beiden lokalen Konstanten MaxWert und MinWert festgelegt. MaxWert blendet die modulweite Konstante gleichen Namens aus, wie man an der Ausgabe im Label ersehen kann.
- Außerdem kommt noch die integrierte Konstante vbCrLf zum Einsatz.

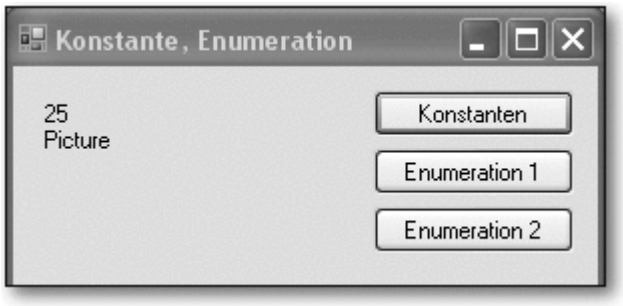

Abbildung 2.5 Konstanten

2.1.6 Enumerationen

Konstanten aufzählen

Enumerationen sind Aufzählungen von Konstanten, die thematisch zusammengehören. Alle Enumerationen haben den gleichen Datentyp, der ganzzahlig sein muss. Bei der Deklaration werden ihnen Werte zugewiesen, am besten explizit.

Innerhalb von Visual Basic gibt es zahlreiche vordefinierte Enumerationen. Ähnlich wie bei den integrierten Konstanten sind die Namen der Enumerationen und deren Elemente besser lesbar als die durch sie repräsentierten Zahlen.

Ein Beispiel: Die Enumeration MsgBoxResult ermöglicht es dem Programmierer, die zahlreichen möglichen Antworten des Benutzers beim Einsatz von Windows-Standard-Dialogfeldern (Ja, Nein, Abbrechen, Wiederholen, Ignorieren, ...) anschaulich einzusetzen.

Im folgenden Programm wird mit einer eigenen und einer vordefinierten Enumeration gearbeitet (ebenfalls in p0204).

```
Public Class frm0204
```

```
[ ... ]
   Enum Zahl As Integer
      Eins = 1
      Zwei = 2
      Drei = 3
      Vier = 4
   End Enum
[ ... ]

   Private Sub cmdEnumeration1_Click( ... ) Handles ...
      lblAnzeige.Text = Zahl.Zwei * Zahl.Drei
   End Sub

   Private Sub cmdEnumeration2_Click( ... ) Handles ...
      lblAnzeige.Text = "Montag: " _
         & FirstDayOfWeek.Sunday & _
         ", Samstag: " & FirstDayOfWeek.Saturday
   End Sub
End Class
```

Zur Erläuterung:

- Es wird die Enumeration Zahl vom Datentyp Integer vereinbart. Da es sich um einen Typ handelt und nicht um eine Variable oder Konstante, muss sie außerhalb von Prozeduren vereinbart werden. Damit ist sie automatisch modulweit gültig

- In der ersten Ereignisprozedur werden zwei Elemente der eigenen Enumeration Zahl verwendet. Die beiden Zahlen, die sie repräsentieren, werden miteinander multipliziert.

- In der zweiten Ereignisprozedur werden zwei Elemente der vordefinierten Enumeration FirstDayOfWeek verwendet.

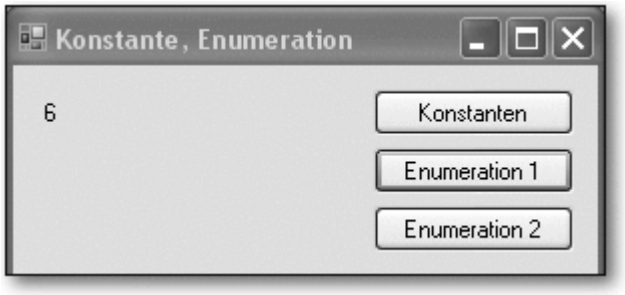

Abbildung 2.6 Erste Enumeration

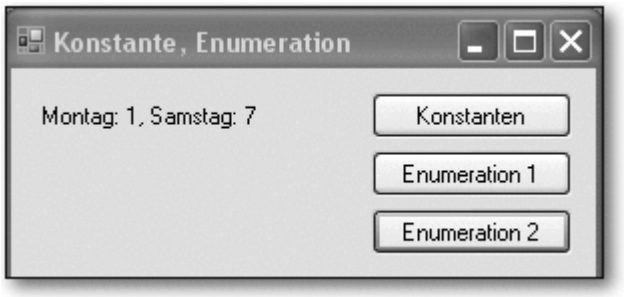

Abbildung 2.7 Zweite Enumeration

Übung p0205:

Erstellen Sie ein Programm, in dem zwei Buttons, ein Label und drei Variablen eines geeigneten Datentyps eingesetzt werden:

- die modulweite Variable x
- die Variable y, die nur lokal in der Prozedur zum Click-Ereignis des ersten Buttons gültig ist
- die Variable z, die nur lokal in der Prozedur zum Click-Ereignis des zweiten Buttons gültig ist
- In der ersten Prozedur werden x und y jeweils um 0,1 erhöht und angezeigt.
- In der zweiten Prozedur werden x und z jeweils um 0,1 erhöht und angezeigt.

Abbildung 2.8 Ausgabe der ersten Prozedur nach einigen Klicks

Abbildung 2.9 Ausgabe der zweiten Prozedur nach weiteren Klicks

2.2 Operatoren

Zum Zusammensetzen von Ausdrücken werden in Visual Basic, wie in jeder anderen Programmiersprache auch, Operatoren verwendet. In diesem Buch wurden schon die Operatoren = für Zuweisungen und & für Verkettungen verwendet.

Priorität

Es gibt verschiedene Kategorien von Operatoren. Vorrangregeln (Prioritäten) sind für die Reihenfolge der Abarbeitung zuständig, falls mehrere Operatoren innerhalb eines Ausdrucks verwendet werden. Diese Vorrangregeln sind weiter unten in diesem Abschnitt angegeben. Falls man sich bei der Verwendung dieser Regeln nicht sicher ist, so empfiehlt es sich, durch eigene Klammersetzung die Reihenfolge explizit festzulegen.

2.2.1 Arithmetische Operatoren

Rechen-Operatoren

Arithmetische Operatoren dienen zur Durchführung von Berechnungen. Die folgende Tabelle listet die arithmetischen Operatoren auf:

Operator	Beschreibung

+	Addition
-	Subtraktion oder Negation
*	Multiplikation
/	Division
\	Ganzzahl-Division
^	Potenzierung
Mod	Modulo

Die Ganzzahl-Division wird in zwei Schritten durchgeführt. Im ersten Schritt werden Dividend und Divisor einzeln gerundet. Im zweiten Schritt werden die beiden verbliebenen Zahlen geteilt, anschließend werden die Ziffern nach dem Komma abgeschnitten. Beispiele:

Ausdruck	Ergebnis
19 / 4	4.75
19 \ 4	4
19 \ 4.6	3
19.5 \ 4.2	5

Modulo

Der Modulo-Operator Mod berechnet den Rest einer Division. Beispiele:

Ausdruck	Ergebnis
19 Mod 4	3
19.5 Mod 4.2	2.7

Zur Potenzierung einer Zahl dient der Operator ^ (hoch). Beispiele:

Ausdruck	Ergebnis
2 ^ 5	32
3 ^ 2 ^ 3	729
2 ^ 5.4	-42.2242531447326
(-2) ^ 5	-32

Multiplikation und Division innerhalb eines Ausdrucks sind gleichrangig und werden von links nach rechts in der Reihenfolge ihres Auftretens ausgewertet. Dasselbe gilt für Additionen und Subtraktionen, die zusammen in einem Ausdruck auftreten.

Klammern

Mit Klammern kann diese Rangfolge außer Kraft gesetzt werden, damit bestimmte Teilausdrücke vor anderen Teilausdrücken ausgewertet werden. In Klammern gesetzte Operationen haben grundsätzlich Vorrang. Innerhalb der Klammern gilt jedoch wieder die normale Rangfolge der Operatoren.

Übung p0206:

Berechnen Sie die beiden folgenden Ausdrücke, speichern Sie das Ergebnis in einer Variablen eines geeigneten Datentyps und zeigen Sie es an:

- 1. Ausdruck: 3 * -2.5 + 4 * 2
- 2. Ausdruck: 3 * (-2.5 + 4) * 2

2.2.2 Vergleichsoperatoren

Vergleich

Vergleichsoperatoren dienen dazu festzustellen, ob bestimmte Bedingungen zutreffen oder nicht. Das Ergebnis nutzt man unter anderem zur Ablaufsteuerung von Programmen. Der Abschnitt 2.4, »Verzweigung« geht hierauf genauer ein. Die folgende Tabelle führt die Vergleichsoperatoren auf:

Operator	Beschreibung
<	kleiner als
<=	kleiner als oder gleich
>	größer als
>=	größer als oder gleich
=	gleich
<>	ungleich

Einige Beispiele:

Ausdruck	Ergebnis
5 > 3	True
3 = 3.2	False
5 + 3 * 2 >= 12	False

Like

Darüber hinaus gibt es noch den Operator Like, der zum Mustervergleich dient. Dabei kann man unter anderem die Platzhalter * (eines oder mehrere Zeichen) und ? (genau ein Zeichen) einsetzen. Beispiele:

Ausdruck	Ergebnis
"abxba" Like "a*a"	True
"abxba" Like "a?a"	False
"aba" Like "a?a"	True
"asdlfigc" Like "a?d?f*c"	True

Übung p0207:

Ermitteln Sie das Ergebnis der beiden folgenden Ausdrücke, speichern Sie es in einer Variablen eines geeigneten Datentyps und zeigen Sie es an:

- 1. Ausdruck: 12 – 3 >= 4 * 2.5
- 2. Ausdruck: "Maier" Like "M??er"

2.2.3 Logische Operatoren

Logik

Logische Operatoren dienen dazu, mehrere Bedingungen zusammenzufassen. Das Ergebnis nutzt man ebenfalls u. a. zur Ablaufsteuerung von Programmen (siehe hierzu auch den Abschnitt 2.4, »Verzweigungen«). Die logischen Operatoren listet die folgende Tabelle auf:

Operator	Beschreibung	Das Ergebnis ist True, wenn ...
Not	Nicht	... der Ausdruck False ist
And	Und	... beide Ausdrücke True sind
Or	Inklusives Oder	... mindestens ein Ausdruck True ist
Xor	Exklusives Oder	... genau ein Ausdruck True ist

Es seien die Variablen A = 1, B = 3 und C = 5 gesetzt. Die folgenden Ausdrücke ergeben dann jeweils:

Ausdruck	Ergebnis
Not (A < B)	False
(B > A) And (C > B)	True
(B < A) Or (C < B)	False
(B < A) Xor (C > B)	True

Eine Zusammenstellung der Funktionsweise der logischen Operatoren liefert auch die folgende Wahrheitstabelle:

Ausdruck 1	Ausdruck 2	Not	And	Or	Xor
True	True	False	True	True	False
True	False	False	False	True	True
False	True	True	False	True	True
False	False	True	False	False	False

Übung p0208:

Ermitteln Sie das Ergebnis der beiden folgenden Ausdrücke, speichern Sie es in einer Variablen eines geeigneten Datentyps und zeigen Sie es an:

- 1. Ausdruck: 4 > 3 And -4 > -3
- 2. Ausdruck: 4 > 3 Or -4 > -3

2.2.4 Verkettungsoperator

Umwandlung in String

Der Operator & dient zur Verkettung von Zeichenfolgen. Ist einer der Ausdrücke keine Zeichenfolge, sondern eine Zahl oder Datumsangabe, so wird er in einen String verwandelt. Das Gesamtergebnis ist dann wiederum eine Zeichenfolge. Beispiel:

```
Public Class frm0209
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim a As String
      Dim s As Single
      Dim d As Date
      d = "5.11.2007"
      s = 4.6
      a = "t " & s & " " & d
      lblAnzeige.Text = a
   End Sub
End Class
```

Das Ergebnis:

Abbildung 2.10 Verkettung

2.2.5 Zuweisungsoperatoren

Zeichen =

Der einfachste Zuweisungsoperator, das Gleichheitszeichen, wurde bereits genutzt. Es gibt zur Verkürzung von Anweisungen noch einige weitere Zuweisungsoperatoren:

Operator	Beispiel	Ergebnis
=	x = 7	x erhält den Wert 7
+=	x += 5	der Wert von x wird um 5 erhöht
-=	x -= 5	der Wert von x wird um 5 verringert
*=	x *= 3	der Wert von x wird auf das Dreifache erhöht
/=	x /= 3	der Wert von x wird auf ein Drittel verringert
\=	x \= 3	der Wert von x wird auf ein Drittel verringert, Nachkommastellen werden abgeschnitten
^=	x ^= 3	der Wert von x wird auf x hoch 3 erhöht
&=	z &= "abc"	die Zeichenkette z wird um den Text »abc« verlängert

2.2.6 Rangfolge der Operatoren

Priorität

Enthält ein Ausdruck mehrere Operationen, so werden die einzelnen Teilausdrücke in einer bestimmten Rangfolge ausgewertet und aufgelöst, die als Rangfolge bzw. Priorität der Operatoren bezeichnet wird. Es gilt die folgende Rangfolge:

Operator	Beschreibung
^	Exponentialoperator
-	negatives Vorzeichen
*, /	Multiplikation, Division
\	Ganzzahl-Division
Mod	Modulo
+, -	Addition, Subtraktion
&	Verkettung
=, <>, <, >, <=, >=, Like	Vergleichsoperatoren (das Zeichen = steht für den Vergleich, nicht für die Zuweisung
Not	logisches NICHT
And	logisches UND
Or	logisches ODER

Die Operatoren, die in der Tabelle weiter oben stehen, haben die höchste Priorität.

Klammern

Wie schon bei den arithmetischen Operatoren erwähnt: Mit Klammern kann diese Rangfolge außer Kraft gesetzt werden, damit bestimmte Teilausdrücke vor anderen Teilausdrücken ausgewertet werden. In Klammern gesetzte Operationen haben grundsätzlich Vorrang. Innerhalb der Klammern gilt jedoch wieder die normale Rangfolge der Operatoren.

Übung p0210:

Sind die folgenden Bedingungen wahr oder falsch? Lösen Sie die Aufgabe möglichst ohne PC.

Nr.	Werte	Bedingung
1	a=5 b=10	a>0 And b<>10
2	a=5 b=10	a>0 Or b<>10
3	z=10 w=100	z<>0 Or z>w Or w-z=90
4	z=10 w=100	z=11 And z>w Or w-z=90
5	x=1.0 y=5.7	x>=.9 And y<=5.8
6	x=1.0 y=5.7	x>=.9 And Not(y<=5.8)
7	n1=1 n2=17	n1>0 And n2>0 Or n1>n2 And n2<>17
8	n1=1 n2=17	n1>0 And (n2>0 Or n1>n2) And n2<>17

2.3 Panel, Zeitgeber, Textfeld, Zahlenauswahlfeld

Windows-Programmierung mit Visual Basic besteht aus zwei Teilen: der Arbeit mit visuellen Steuerelementen und der Programmierung mit der Sprache. Beides soll in diesem Buch parallel vermittelt werden, damit die eher theoretischen Abschnitte zur Programmiersprache durch eine anschauliche Praxis vertieft werden können.

Daher wird in diesem Abschnitt mit vier weiteren Steuerelementen gearbeitet, bevor im nächsten Abschnitt die Verzweigungen zur Programmsteuerung vorgestellt werden.

2.3.1 Panel

Container

Ein Panel dient normalerweise als Container für andere Steuerelemente. In diesem Abschnitt wird es zur visuellen Darstellung eines Rechtecks und für eine kleine Animation genutzt.

Die Eigenschaften BackColor (Hintergrundfarbe), Location (Position) und Size (Größe) sind schon von anderen Steuerelementen bekannt.

Mithilfe des nachfolgenden Programms p0211 wird ein Panel durch Betätigung von vier Buttons um 10 Pixel nach oben, unten, links oder rechts verschoben. Es hat die Größe 100 x 100 Pixel, die Startposition X=100 und Y=100 sowie eine eigene Hintergrundfarbe. Die Bewegung wird mithilfe der Struktur Point durchgeführt.

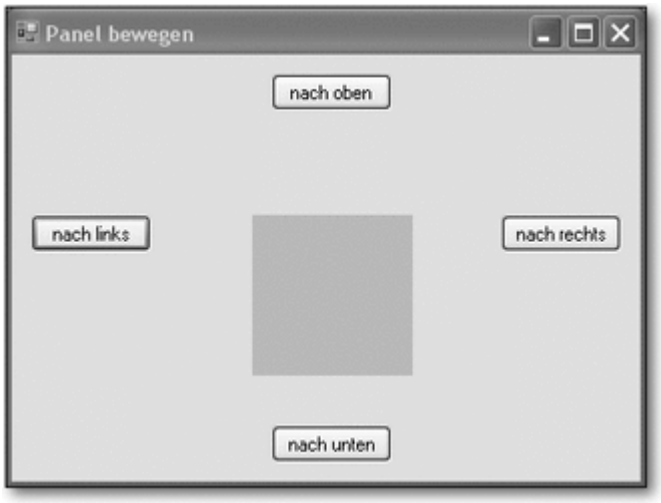

Abbildung 2.11 Panel, Startzustand

```
Public Class frm0211
   Private Sub cmdOben_Click( ... ) Handles ...
      panMove.Location = New Point(_
         panMove.Location.X, _
         panMove.Location.Y - 10)
   End Sub

   Private Sub cmdUnten_Click( ... ) ...
      panMove.Location = New Point(_
         panMove.Location.X, _
         panMove.Location.Y + 10)
   End Sub
   Private Sub cmdLinks_Click( ... ) Handles ...
      panMove.Location = New Point(_
         panMove.Location.X - 10, _
         panMove.Location.Y)
   End Sub

   Private Sub cmdRechts_Click( ... ) Handles ...
      panMove.Location = New Point(_
         panMove.Location.X + 10, _
         panMove.Location.Y)
   End Sub
End Class
```

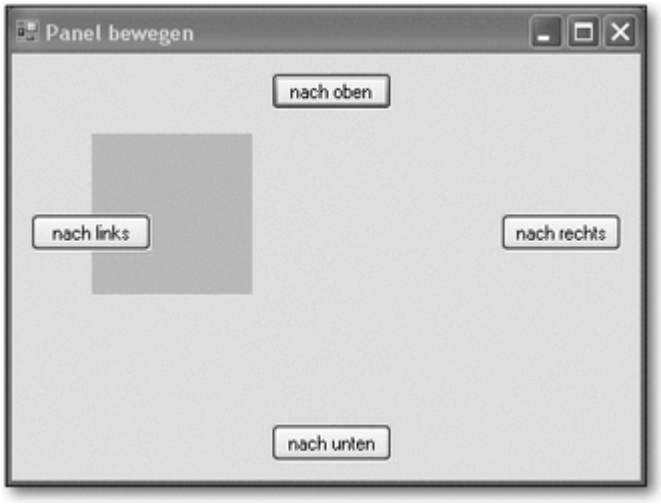

Abbildung 2.12 Panel nach Verschiebung

2.3.2 Zeitgeber

Timer

Ein Zeitgeber (Timer) erzeugt in festgelegten Abständen Zeittakte. Diese Zeittakte sind Ereignisse, die der Entwickler mit Aktionen verbinden kann. Ein Zeitgeber kann wie jedes andere Steuerelement zum Formular hinzugefügt werden. Da es sich aber um ein nicht sichtbares Steuerelement handelt, wird er unterhalb des Formulars angezeigt. Auch zur Laufzeit ist er nicht sichtbar.

Intervall

Seine wichtigste Eigenschaft ist das Zeitintervall, in dem das Ereignis auftreten soll. Dieses Zeitintervall wird in Millisekunden angegeben.

Enabled

Die Eigenschaft Enabled dient zur Aktivierung bzw. Deaktivierung des Zeitgebers. Sie kann zur Entwicklungszeit oder zur Laufzeit auf True oder False gestellt werden.

Im nachfolgenden Programm p0212 erscheint zunächst ein Formular mit zwei Buttons. Betätigt man den Start-Button, so erscheint ein »x« in einem Bezeichnungsfeld. Alle 0,5 Sekunden erscheint automatisch ein weiteres »x«. Dies wird durch den Timer gesteuert, bei dem der Wert für die Eigenschaft Interval auf 500 gesetzt wurde. Nach Betätigung des Stop-Buttons kommt kein weiteres »x« hinzu.

Abbildung 2.13 Nach einigen Sekunden

```
Public Class frm0212
   Private Sub timAnzeige_Tick( ... ) Handles ...
      lblAnzeige.Text &= "x"
   End Sub

   Private Sub cmdStart_Click( ... ) Handles ...
      timAnzeige.Enabled = True
   End Sub

   Private Sub cmdStop_Click( ... ) Handles ...
      timAnzeige.Enabled = False
   End Sub
End Class
```

Übung p0213:

Erstellen Sie eine Windows-Anwendung. In der Mitte eines Formulars sollen zu Beginn vier Panels verschiedener Farbe der Größe 20 × 20 Pixel platziert werden. Sobald ein Start-Button betätigt wurde, sollen diese vier Panels sich diagonal in ca. 5–10 Sekunden zu den Ecken des Formulars bewegen, jedes Panel in eine andere Ecke.

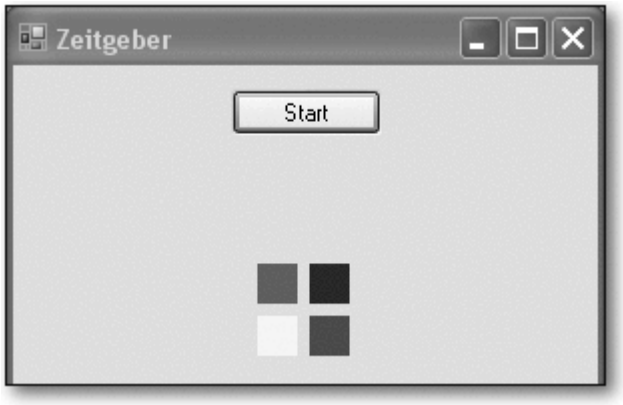

Abbildung 2.14 Start-Zustand

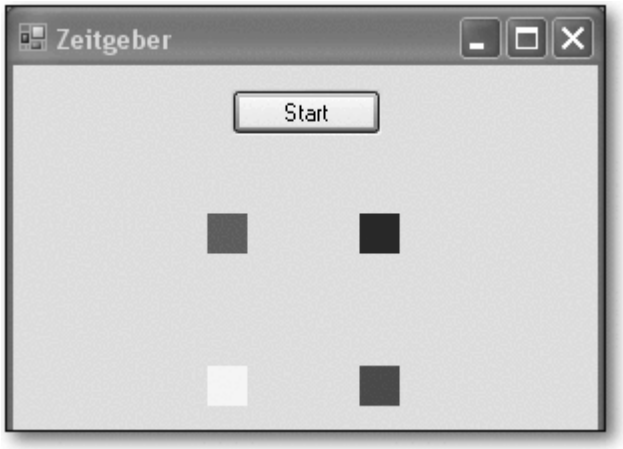

Abbildung 2.15 Nach einigen Sekunden

Übung p0214:

Diese Übung gehört nicht zum »Pflichtprogramm«. Sie ist etwas umfangreicher, verdeutlicht aber die Möglichkeiten einer schnellen Visualisierung von Prozessen durch Visual Basic mit wenigen Programmzeilen.

Konstruieren Sie aus mehreren Panels einen Kran (Fundament, senkrechtes Hauptelement, waagrechter Ausleger, senkrechter Haken am Ausleger). Der Benutzer soll die Möglichkeit haben, über Buttons die folgenden Aktionen auszulösen:

- Haken um 10 Pixel ausfahren
- Haken um 10 Pixel einfahren
- Ausleger um 10 Pixel ausfahren
- Ausleger um 10 Pixel einfahren
- Kran um 10 Pixel nach rechts fahren

- Kran um 10 Pixel nach links fahren
- Kran um 10 Pixel in der Höhe ausfahren
- Kran um 10 Pixel in der Höhe einfahren

Denken Sie daran, dass bei vielen Bewegungen mehrere Steuerelemente bewegt werden müssen, da der Kran sonst seinen »Zusammenhalt« verliert. Manche Aktionen resultieren nur aus Größenveränderungen (Eigenschaften Width und Height), manche nur aus Ortsveränderungen, manche aus beidem.

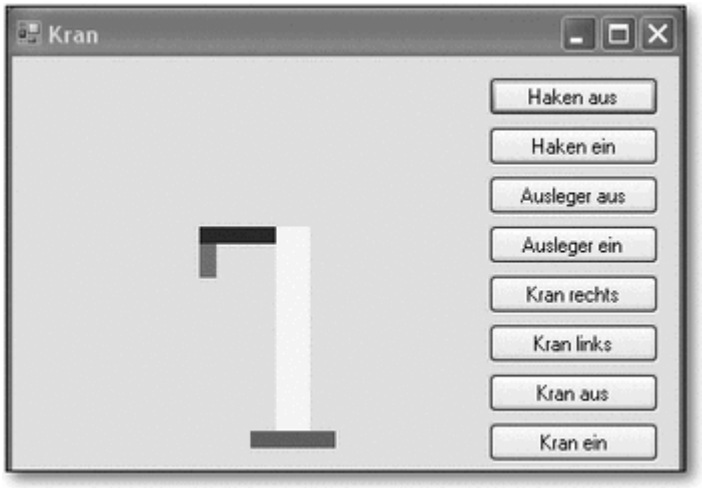

Abbildung 2.16 Start-Zustand

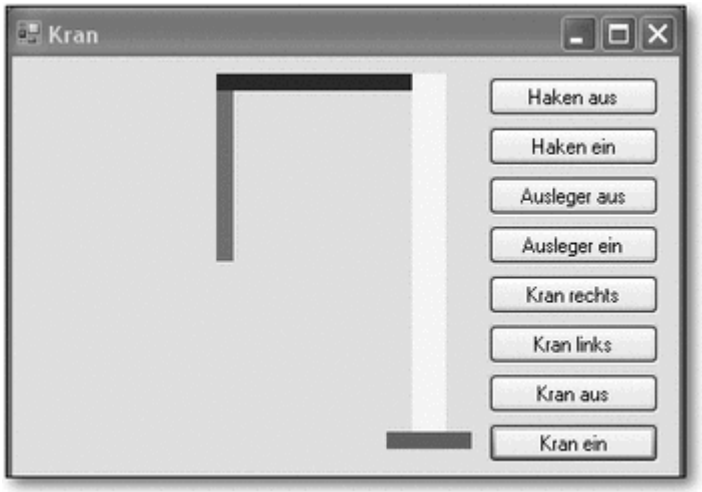

Abbildung 2.17 Nach einigen Aktionen

2.3.3 Textfelder

Eingabefeld

Ein Textfeld dient in erster Linie dazu, die Eingabe von Text oder Zahlen vom Benutzer entgegenzunehmen. Diese Eingaben werden in der Eigenschaft Text des Textfelds gespeichert. Das Aussehen und das Verhalten eines Textfelds werden unter anderem durch folgende Eigenschaften gekennzeichnet:

- MultiLine: Steht MultiLine auf True, so kann bei der Eingabe und bei der Anzeige mit mehreren Textzeilen gearbeitet werden.

- ScrollBars: Man kann ein Textfeld mit vertikalen und/oder horizontalen Bildlaufleisten zur Eingabe und Anzeige längerer Texte versehen.

- MaxLength: Mit dieser Eigenschaft lässt sich die Anzahl der Zeichen des Textfelds beschränken. Ist keine Beschränkung vorgesehen, kann das Textfeld 32K Zeichen aufnehmen.

Passwort

- PasswordChar: Falls für diese Eigenschaft im Entwurfsmodus ein Platzhalter-Zeichen eingegeben wurde, wird während der Laufzeit für jedes eingegebene Zeichen nur dieser Platzhalter angezeigt. Diese Eigenschaft wird vor allem bei Passwort-Abfragen verwendet.

Im nachfolgenden Programm p0215 kann der Benutzer in einem Textfeld einen Text eingeben. Nach Betätigung des Buttons Ausgabe wird der eingegebene Text in einem zusammenhängenden Satz ausgegeben.

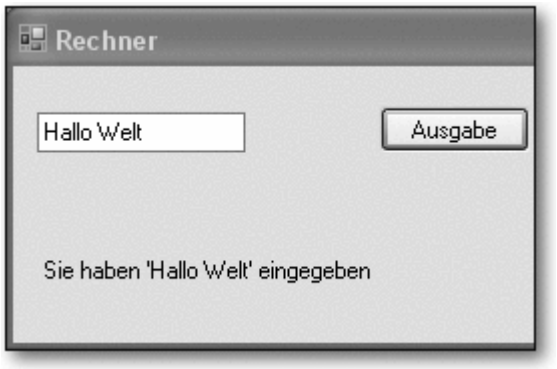

Abbildung 2.18 Eingabe in Textfeld

Der Code lautet wie folgt:

```
Public Class frm0215
   Private Sub cmdAusgabe_Click( ... ) Handles ...
      lblAusgabe.Text = "Sie haben '" _
         & txtEingabe.Text & "' eingegeben"
   End Sub
End Class
```

Zur Erläuterung:

- In der Eigenschaft Text des Textfelds wird die Eingabe gespeichert. Die Eigenschaft wird in einen längeren Ausgabetext eingebettet.

Zahlen eingeben

Bei der Eingabe und Auswertung von Zahlen sind einige Besonderheiten zu beachten. Im nachfolgenden Programm, ebenfalls in Projekt p0215, kann der Benutzer in einem Textfeld eine Zahl eingeben. Nach Betätigung des Buttons Rechnen 1 wird der Wert dieser Zahl verdoppelt, das Ergebnis wird in einem Label darunter ausgegeben.

```
Public Class frm0215
[ ... ]
```

```
   Private Sub cmdRechnen1_Click( ... ) Handles ...
      lblAusgabe.Text = txtEingabe.Text * 2
   End Sub
End Class
```

Zur Erläuterung:

- Wenn eine Zeichenkette eingegeben wurde, die eine Zahl darstellt, dann wird sie implizit, d. h. automatisch, in eine Zahl umgewandelt, mit der dann gerechnet werden kann.

- Stellt die eingegebene Zeichenkette keine Zahl dar, kommt es zu einem Laufzeitfehler. Diese Situation sollte natürlich vermieden werden:

- Man kann vorher überprüfen, ob es sich bei der Zeichenkette um eine gültige Zahl handelt und entsprechend reagieren. Dies wird möglich, sobald man Verzweigungen zur Programmsteuerung beherrscht.

  Ausnahmebehandlung

- Allgemein kann man Programme so schreiben, dass ein Programmabbruch abgefangen werden kann. Dies wird möglich, sobald man die Ausnahmebehandlung (siehe hierzu Kapitel 3, »Fehlerbehandlung«) beherrscht.

Einige Beispiele:

Abbildung 2.19 zeigt die Eingabe einer Zahl mit Nachkommastellen.

Abbildung 2.19 Eingabe einer Zahl mit Nachkommastellen

Die Eingabe einer Zeichenkette, z. B. »abc«, führt zu einer Ausnahme. Die Zeile, in der der Fehler auftritt, wird im Code markiert, damit der Fehler beseitigt werden kann (siehe Abbildung 2.20).

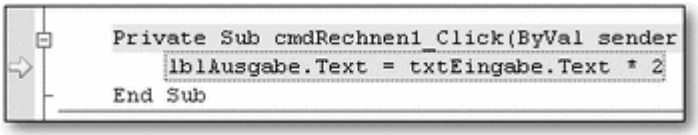

Abbildung 2.20 Markierung der Fehlerzeile nach Eingabe des Textes »abc«

Die Eingabe einer Zahl, bei der ein Punkt statt einem Komma zur Abtrennung von Nachkommastellen eingegeben wird, führt zu einem ganz anderen Rechenergebnis:

Abbildung 2.21 Eine Zahl mit einem Punkt vor den Nachkommastellen

Der Punkt wird ignoriert, die Zahl wird als 352 angesehen und führt so zu dem Ergebnis 704.

Val()

Eine Verbesserung, gleichzeitig aber eine Umstellung ergibt sich, wenn man die Funktion Val() verwendet. Die Eingabe einer Zeichenkette (z. B. »abc«) führt nicht mehr zu einer Ausnahme, da die Funktion Val() aus einer Zeichenkette den Wert 0 ermittelt. Mit 0 kann gerechnet werden.

Allerdings sollten die Zahlen nun in englischer Schreibweise eingegeben werden, also mit einem Punkt vor den Nachkommastellen.

Abbildung 2.22 Funktion Val()

```
Public Class frm0215
[ ... ]
   Private Sub cmdRechnen2_Click( ... ) Handles ...
      lblAusgabe.Text = Val(txtEingabe.Text) * 2
   End Sub
End Class
```

Zur Erläuterung:

- Der eingegebene Text wird mit der Funktion Val() in eine Zahl umgewandelt. Falls keine Zahl eingegeben wurde, ergibt sich als Wert 0.

2.3.4 Zahlenauswahlfeld

NumericUpDown

Das Steuerelement Zahlenauswahlfeld (NumericUpDown) bietet eine andere Möglichkeit, Werte an ein Programm zu übermitteln. Die Werte können innerhalb selbst definierter Grenzen und in selbst definierten Schritten über zwei kleine Pfeiltasten ausgewählt werden.

Wichtige Eigenschaften des Steuerelements sind:

Value

- Value: Bezeichnet zur Entwicklungszeit den Startwert und zur Laufzeit den vom Benutzer aktuell eingestellten Wert.

- Maximum, Minimum: Bestimmt den größtmöglichen Wert und den kleinstmöglichen Wert der Eigenschaft Value. Es handelt sich also um die Werte, die durch die Auswahl mit den Pfeiltasten ganz oben und ganz unten erreicht werden können.

- Increment: Mit Increment wird die Schrittweite eingestellt, mit der sich der Wert (Eigenschaft Value) ändert, wenn der Benutzer eine der kleinen Pfeiltasten betätigt.

- DecimalPlaces: Bestimmt die Anzahl der Nachkommastellen in der Anzeige des Zahlenauswahlfelds.

ValueChanged

Das wichtigste Ereignis dieses Steuerelements ist ValueChanged. Es tritt bei der Veränderung der Eigenschaft Value ein und sollte anschließend zur Programmsteuerung verwendet werden.

Im nachfolgenden Programm p0216 werden alle diese Eigenschaften und das genannte Ereignis genutzt. Der Benutzer kann Zahlenwerte zwischen -5,0 und +5,0 in Schritten von 0,1 über ein Zahlenauswahlfeld einstellen. Der ausgewählte Wert wird unmittelbar in einem Label angezeigt.

Abbildung 2.23 Zahlenauswahlfeld

Die Eigenschaften wurden zur Entwicklungszeit wie folgt eingestellt:

- Value: Wert 2, die Anwendung startet bei dem Wert 2,0 für das Zahlenauswahlfeld

- Maximum, Minimum: Wert -5 und +5

- Increment: Wert 0,1

- DecimalPlaces: Wert 1, zur Anzeige einer einzelnen Nachkommastelle

Der Code lautet:

```
Public Class frm0216
   Private Sub numEingabe_ValueChanged( ... ) Handles ...
      lblAusgabe.Text = numEingabe.Value
   End Sub
End Class
```

2.4 Verzweigungen

Der Programmcode wurde bisher rein sequentiell abgearbeitet, d. h. eine Anweisung nach der anderen. Kontrollstrukturen ermöglichen eine Steuerung dieser Reihenfolge. Die Kontrollstrukturen un-

terteilen sich in Verzweigungen und Schleifen. Verzweigungen gestatten dem Programm, in verschiedene alternative Anweisungsblöcke zu verzweigen.

Es gibt die beiden Verzweigungsstrukturen If...Then...Else und Select Case.... Diese Auswahlmöglichkeiten übergeben aufgrund von Bedingungen die Programmausführung an einen bestimmten Anweisungsblock. Bedingungen werden mithilfe der bereits vorgestellten Vergleichsoperatoren erstellt.

Seltener genutzt werden außerdem noch die Auswahlfunktionen IIf() und Choose().

2.4.1 Einzeiliges If…Then…Else

Das einzeilige If...Then...Else hat folgenden Aufbau:

```
If Bedingung Then Anweisungen1 [ Else Anweisungen2 ]
```

If…Then…Else

Die Bedingung wird ausgewertet, sie ist entweder wahr oder falsch (True oder False). Ist das Ergebnis der Auswertung True, so wird der Then-Teil mit den Anweisungen1 ausgeführt. Ist das Ergebnis der Auswertung False und gibt es einen Else-Teil, so wird der Else-Teil mit den Anweisungen2 ausgeführt.

Dabei kann es sich sowohl um eine einzelne Anweisung als auch um mehrere Anweisungen handeln, die dann durch einen Doppelpunkt (:) voneinander getrennt sind. In jedem Fall muss der gesamte Block in einer Zeile untergebracht werden. If-Strukturen können auch ineinander verschachtelt werden.

Einzeilig

Im nachfolgenden Programm p0217 wird das einzeilige If in vier verschiedenen Beispielen genutzt. Dabei ist zu beachten, dass die einzelnen Zeilen zu lang für den Druck in diesem Buch sind. Es handelt sich aber in jedem Fall um einzeilige If-Verzweigungen.

```
Public Class frm0217
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim x As Integer
      x = -1
      If x > 0 Then lblAnzeige.Text = "Positiv"
      If x > 0 Then lblAnzeige.Text = "Positiv" Else _
         lblAnzeige.Text = "Negativ oder Null"
      If x > 0 Then lblAnzeige.Text = "Positiv" Else _
         If x = 0 Then lblAnzeige.Text = "Null" _
         Else lblAnzeige.Text = "Negativ"
      If x > 0 Then _
         x = x + 1 : lblAnzeige.Text = "Positiv " & x _
         Else x = x - 1 : lblAnzeige.Text = _
         "Negativ oder Null " & x
   End Sub
End Class
```

Zur Erläuterung:

- Die Integer-Variable x erhält den Wert 1. Für die Tests in den einzelnen Beispielen muss dieser Wert natürlich auch einmal auf 0 oder einen anderen Wert (positiv oder negativ) gesetzt werden.

- Beim ersten Beispiel wird nur etwas angezeigt, falls die Variable x positiv ist.

- Beim zweiten Beispiel wird in jedem Falle etwas angezeigt.

- Beim dritten Beispiel wird für den Fall, dass die Variable x nicht positiv ist, eine weitere Verzweigung durchlaufen. Man nennt diese Verzweigung auch eine innere Verzweigung, im Gegensatz zu einer äußeren Verzweigung. Ist die Variable x = 0, so wird wegen der inneren Verzweigung »Null« angezeigt, andernfalls wird »Negativ« angezeigt.

- Beim vierten Beispiel werden für beide möglichen Fälle jeweils zwei Anweisungen durchlaufen, die durch : (Doppelpunkt) voneinander getrennt sind.

Natürlich sieht man die Ergebnisse der einzelnen Beispiele nur dann im Label, wenn man die Beispiele, die danach kommen, auskommentiert.

2.4.2 If…Then…Else-Block

Mehrzeilig, Block

Bei einfachen Entscheidungen und einzelnen Anweisungen ist das einzeilige If geeignet. Sobald mehrere Anweisungen auszuführen sind, wird der Programmcode schnell unübersichtlich. Für diese Zwecke ist ein If...Then...Else -Block wesentlich besser geeignet. Der Block hat folgenden Aufbau:

```
If Bedingung1 Then
   Anweisungen1
[ ElseIf Bedingung2
   Anweisungen2 ] ...
[ Else
   AnweisungenX ]
End If
```

Das Programm verzweigt zu den Anweisungen hinter der ersten zutreffenden Bedingung. Falls keine Bedingung zutrifft, werden die Anweisungen hinter dem Else ausgeführt, sofern es diesen Else-Zweig gibt. Andernfalls wird keine Anweisung durchgeführt. Ein If...Then...Else-Block endet immer mit einem End If.

Im nachfolgenden Programm p0218 werden vier verschiedene Fälle geprüft. Trifft keiner dieser Fälle zu, so wird der Else-Zweig ausgeführt:

```
Public Class frm0218
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim x As Integer
      Dim y As Integer
      x = 0
      y = 0
      If x >= 0 And y >= 0 Then
         lblAnzeige.Text = _
            "Beide größer oder gleich Null"
      ElseIf x >= 0 And y < 0 Then
         lblAnzeige.Text = _
            "Nur X größer oder gleich Null"
      ElseIf x < 0 And y >= 0 Then
```

```
         lblAnzeige.Text = _
            "Nur Y größer oder gleich Null"
      ElseIf x >= 0 Then
         lblAnzeige.Text = "Wird nie angezeigt"
      Else
         lblAnzeige.Text = "Beide kleiner Null"
      End If
   End Sub
End Class
```

2.4.3 Select Case

Mehrfachauswahl

Die Anweisung Select Case ... kann als Alternative zum If...Then ...Else -Block gelten. Sie vereinfacht eine Mehrfachauswahl, wenn nur ein Wert untersucht werden muss, und ist wie folgt aufgebaut:

```
Select Case Testausdruck
   [ Case Ausdrucksliste1
      Anweisungen1 ]
   [ Case Ausdrucksliste2
      Anweisungen2 ] ...
   [ Case Else
      AnweisungenX ]
End Select
```

To, Is

Die Anweisung Select Case ... verwendet nur einen Testausdruck, der am Beginn der Struktur ausgewertet wird. Sein Wert wird anschließend der Reihe nach mit den Werten der Ausdrucksliste verglichen. Eine Ausdrucksliste kann aus mehreren Ausdrücken oder einer Bereichsangabe mit dem Schlüsselwort To bestehen. Ein Ausdruck kann aus einem Wert oder einer Bedingung mit dem Schlüsselwort Is bestehen.

End Select

Bei der ersten Übereinstimmung wird der zugehörige Anweisungsblock ausgeführt und dann mit der nächsten Anweisung hinter dem End Select fortgefahren.

Case Else

Der optionale Anweisungsblock hinter dem Case Else wird ausgeführt, falls vorher keine Übereinstimmung gefunden wurde.

Im nachfolgenden Programm p0219 werden ebenfalls vier verschiedene Fälle geprüft. Trifft keiner dieser Fälle zu, wird der Case Else-Zweig ausgeführt:

```
Public Class frm0219
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim x As Integer
      x = 16
      Select Case x
         Case 1, 3, 5, 7, 9
            lblAnzeige.Text = "Ungerade, Einstellig"
         Case 2, 4, 6, 8
            lblAnzeige.Text = "Gerade, Einstellig"
```

```
         Case Is < 1, Is > 20
            lblAnzeige.Text = _
               "Kleiner Eins oder größer 20"
         Case 11 To 15
            lblAnzeige.Text = "Größer gleich 11 " _
               & "und kleiner gleich 15"
         Case Else
            lblAnzeige.Text = _
               "Größer 15 und kleiner 21"
      End Select
   End Sub
End Class
```

In diesem Beispiel sind nur die Zahlen größer 15 und kleiner 21 in keiner Ausdrucksliste enthalten. Der entsprechende Text ist also im Case Else-Zweig zu finden.

2.4.4 Funktion IIf

Liefert Wert

Die Funktion IIf() ähnelt dem einzeiligen If...Then...Else, liefert allerdings im Unterschied zu diesem direkt einen Wert zurück. Ihre Syntax lautet:

```
IIf(Bedingung, True-Ausdruck, False-Ausdruck)
```

Sowohl True-Ausdruck als auch False-Ausdruck müssen angegeben werden. Im nachfolgenden Programm wird das Maximum der beiden Zahlen x und y ermittelt und ausgegeben:

```
Public Class frm0220
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim x As Integer
      Dim y As Integer
      x = 5
      y = 3
      lblAnzeige.Text = IIf(x > y, x, y)
   End Sub
End Class
```

2.4.5 Funktion Choose

Wert aus Liste

Die Funktion Choose() gibt den Wert aus einer Liste zurück, dessen Position dem Indexwert entspricht. Die Positionen in der Liste beginnen allerdings bei 1, nicht bei 0. Die Syntax lautet:

```
Choose(Index, Ausdruck1, [Ausdruck2] ...)
```

Im nachfolgenden Programm wird eine Währung aus einer Liste von Währungen ausgewählt und ausgegeben:

```
Public Class frm0221
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      Dim x As Integer
      x = 2
      lblAnzeige.Text = Choose(x, "US Dollar", _
         "Pfund", "Euro")
```

```
   End Sub
End Class
```

Zur Erläuterung:

- Der Wert x = 2 führt zur Ausgabe von »Pfund«.

2.4.6 Übungen

Übung p0222:

Schreiben Sie ein Programm, das zu einem eingegebenen Gehalt den Steuerbetrag berechnet und ausgibt. In der folgenden Tabelle sind die Steuersätze angegeben.

Gehalt	Steuersatz
bis einschl. 12.000 €	12 %
von 12.000 bis einschl. 20.000 €	15 %
von 20.000 bis einschl. 30.000 €	20 %
über 30.000 €	25 %

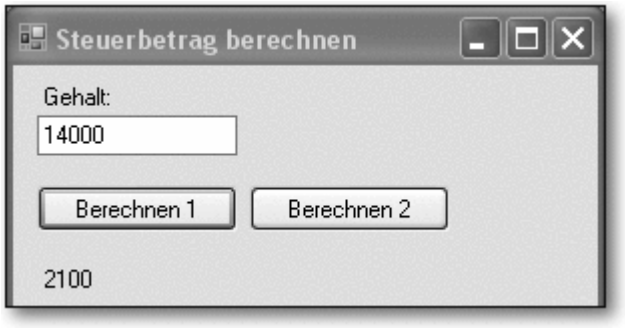

Abbildung 2.24 Übung p0222

Übung p0223:

Erweitern Sie die Übung p0214. Die Bewegung des Krans soll kontrolliert werden. Kein Teil des Krans darf zu groß oder zu klein werden. Der Kran darf sich nicht über die sinnvollen Begrenzungen hinaus bewegen. Nutzen Sie Bedingungen und Verzweigungen, um dies zu verhindern.

2.5 Kontrollkästchen, Optionsschaltfläche, Gruppe

Mithilfe der beiden Steuerelemente Kontrollkästchen und Optionsschaltfläche können Zustände unterschieden bzw. Eigenschaften eingestellt werden. Dazu werden Verzweigungen benötigt, die Gegenstand des vorherigen Abschnitts waren.

2.5.1 Kontrollkästchen

Checkbox

Das Kontrollkästchen (Checkbox) bietet dem Benutzer die Möglichkeit, zwischen zwei Zuständen zu wählen, z. B. An oder Aus, wie bei einem Schalter. Man kann damit auch kennzeichnen, ob man eine bestimmte optionale Erweiterung wünscht oder nicht. Der Benutzer bedient ein Kontrollkästchen, indem er ein Häkchen setzt oder entfernt.

CheckedChanged

Das wichtigste Ereignis ist beim Kontrollkästchen nicht der Click, sondern das Ereignis CheckedChanged. Dieses Ereignis zeigt nicht nur an, dass das Kontrollkästchen vom Benutzer bedient wurde, sondern auch, dass es seinen Zustand geändert hat. Dies kann beispielsweise auch durch Programmcode geschehen. Eine Ereignisprozedur zu CheckedChanged löst in jedem Fall etwas aus, sobald das Kontrollkästchen (vom Benutzer oder vom Programmcode) geändert wurde.

An/Aus

Allerdings wird der Programmablauf meist so gestaltet, dass bei einem anderen Ereignis der aktuelle Zustand des Kontrollkästchens (An/Aus) abgefragt und anschließend je nach Zustand unterschiedlich reagiert wird.

Die wichtigen Eigenschaften des Kontrollkästchens sind:

Checked

- Checked - der Zustand der Checkbox, mit den Werten True und False
- Text – die Beschriftung neben dem Kontrollkästchen

Im nachfolgenden Programm p0224 werden alle oben genannten Möglichkeiten genutzt:

Abbildung 2.25 Zustand nach Klick auf Kontrollkästchen

```
Public Class frm0224
   Private Sub cmdEreignis_Click( ... ) Handles ...
      If chkSchalter.Checked Then
         lblTest1.Text = "An"
      Else
         lblTest1.Text = "Aus"
      End If
   End Sub

   Private Sub chkSchalter_CheckedChanged( ... ) _
         Handles ...
      If chkSchalter.Checked Then
         lblTest2.Text = "An"
      Else
```

```
         lblTest2.Text = "Aus"
      End If
   End Sub

   Private Sub cmdAnschalten_Click( ... ) Handles ...
      chkSchalter.Checked = True
   End Sub
End Class
```

Zur Erläuterung:

- Der Zustand eines Kontrollkästchens (Häkchen gesetzt oder nicht) kann im Programm mithilfe einer einfachen Verzweigung ausgewertet werden.

  Wahrheitswert

- Normalerweise werden bei einer Bedingung in einer Verzweigung zwei Werte durch Vergleichsoperatoren miteinander verglichen und eines der beiden Ergebnisse True oder False ermittelt. Da die Eigenschaft Checked aber bereits einem solchen Wahrheitswert entspricht, kann die Bedingung auch verkürzt formuliert werden. If chkSchalter.Checked = True ... hätte also das gleiche Ergebnis erzeugt.

- Die Prozedur cmdEreignis_Click() wird aufgerufen, wenn der Benutzer den Button Ereignis betätigt. Erst in diesem Moment wird der Zustand des Kontrollkästchens (Eigenschaft Checked = True oder False) abgefragt und im ersten Label ausgegeben. Es kann also sein, dass das Kontrollkästchen vor längerer Zeit oder noch nie benutzt wurde.

- Dagegen wird die Prozedur chkSchalter_CheckedChanged() sofort aufgerufen, wenn der Benutzer das Kontrollkästchen benutzt, also ein Häkchen setzt oder entfernt. Die Prozedur wird auch dann aufgerufen, wenn der Benutzer den Zustand des Kontrollkästchens durch Programmcode ändert. Hier wird der Zustand des Kontrollkästchens also unmittelbar nach der Änderung ausgegeben (im zweiten Label).

- Die Prozedur cmdAnschalten_Click() dient zum Setzen des Häkchens beim Kontrollkästchen per Programmcode. Dies kommt in Windows-Anwendungen häufig vor, wenn es logische Zusammenhänge zwischen mehreren Steuerelementen gibt. Die Eigenschaft Checked wird auf True gesetzt. Dies führt wiederum zum Ereignis chkSchalter_CheckedChanged und dem Ablauf der zugehörigen, oben erläuterten Ereignisprozedur.

2.5.2 Optionsschaltfläche

Radio Button

Optionsschaltflächen (Radio Buttons) treten immer in Gruppen auf und bieten dem Benutzer die Möglichkeit, zwischen zwei oder mehr Möglichkeiten zu wählen, etwa zwischen den Farben Rot, Grün oder Blau. Bei zusammengehörigen Optionsschaltflächen kann der Benutzer genau eine per Klick auswählen. Alle anderen werden dann unmittelbar als »Nicht ausgewählt« gekennzeichnet.

CheckedChanged

Analog zum Kontrollkästchen ist das wichtigste Ereignis bei einer Optionsschaltfläche CheckedChanged. Dieses Ereignis zeigt nicht nur an, dass die betreffende Optionsschaltfläche vom

Benutzer ausgewählt wurde, sondern auch, dass sie ihren Zustand geändert hat. Dies kann auch durch Programmcode geschehen.

Der Programmablauf wird auch hier meist so gestaltet, dass bei einem anderen Ereignis der aktuelle Zustand der Gruppe abgefragt wird und anschließend je nach Zustand unterschiedlich reagiert wird.

Es ist guter Programmierstil und verringert Folgefehler, wenn man eine der Optionsschaltflächen der Gruppe bereits zur Entwicklungszeit auf True setzt. Dies muss nicht notwendigerweise die erste Optionsschaltfläche der Gruppe sein.

Checked

Die wichtigen Eigenschaften der Optionsschaltflächen sind Checked (mit den Werten True und False) und Text (zur Beschriftung). Im nachfolgenden Programm p0225 werden alle genannten Möglichkeiten genutzt. Es wird der Zustand angezeigt, nachdem der Benutzer

- Blau gewählt,
- den Button Ereignis betätigt,
- Grün gewählt hat (siehe Abbildung 2.26).

```
Public Class frm0225
   Private Sub cmdEreignis_Click( ... ) Handles ...
      If optFarbeRot.Checked Then
         lblAnzeige1.Text = "Rot"
      ElseIf optFarbeGrün.Checked Then
         lblAnzeige1.Text = "Grün"
      Else
         lblAnzeige1.Text = "Blau"
      End If
   End Sub

   Private Sub optFarbeRot_CheckedChanged( ... ) _
         Handles ...
      If optFarbeRot.Checked Then
         lblAnzeige2.Text = "Rot"
      End If
   End Sub

   Private Sub optFarbeGrün_CheckedChanged( ... ) _
         Handles ...
      If optFarbeGrün.Checked Then
         lblAnzeige2.Text = "Grün"
      End If
   End Sub

   Private Sub optFarbeBlau_CheckedChanged( ... ) _
         Handles ...
      If optFarbeBlau.Checked Then
         lblAnzeige2.Text = "Blau"
      End If
   End Sub

   Private Sub cmdSchalter_Click( ... ) Handles ...
      optFarbeRot.Checked = True
   End Sub
End Class
```

Abbildung 2.26 Zustand nach zwei Klicks

Zur Erläuterung:

- Der Zustand einer einzelnen Optionsschaltfläche kann im Programm mithilfe einer einfachen Verzweigung ausgewertet werden.

- Der Zustand einer Gruppe von Optionsschaltflächen kann im Programm mithilfe einer mehrfachen Verzweigung ausgewertet werden.

- Die Prozedur cmdEreignis_Click() wird aufgerufen, wenn der Benutzer den Button Ereignis betätigt. Erst in diesem Moment wird der Zustand der Gruppe abgefragt und im ersten Label ausgegeben.

- Dagegen wird eine der Prozeduren optFarbeRot_CheckedChanged() (bzw. ...Grün... oder ...Blau...) sofort aufgerufen, wenn der Benutzer eine der Optionsschaltflächen auswählt. Diese Prozeduren werden jeweils auch dann aufgerufen, wenn der Benutzer den Zustand der zugehörigen Optionsschaltfläche durch Programmcode ändert. Hier wird der Zustand der Gruppe also unmittelbar nach der Änderung ausgegeben (im zweiten Label).

- Die Prozedur cmdAnschalten_Click() dient zur Auswahl einer bestimmten Optionsschaltfläche per Programmcode. Dies kommt in Windows-Anwendungen häufig vor, wenn es logische Zusammenhänge zwischen mehreren Steuerelementen gibt. Die Eigenschaft Checked wird auf True gesetzt. Dies führt wiederum zum Ereignis CheckedChanged der jeweiligen Optionsschaltfläche und zum Ablauf der zugehörigen, oben erläuterten Ereignisprozedur.

Innerhalb eines Formulars oder einer GroupBox (siehe übernächster Abschnitt) kann immer nur bei einer Optionsschaltfläche die Eigenschaft Checked den Wert True haben. Sobald eine andere Optionsschaltfläche angeklickt wird, ändert sich der Wert der Eigenschaft bei der bisher gültigen Optionsschaltfläche.

2.5.3 Mehrere Ereignisse in einer Prozedur behandeln

Im Folgenden wird eine häufig verwendete Technik vorgestellt. Gibt es mehrere Ereignisse, die auf die gleiche oder auf ähnliche Weise behandelt werden sollen, ist es vorteilhaft, diese Ereignisse mit einer gemeinsamen Ereignisprozedur aufzurufen.

Handles

Dies ist möglich, da nach dem Schlüsselwort Handles zu Beginn der Ereignisprozedur mehrere Ereignisse genannt werden können. Im nachfolgenden Programm wird diese Technik verwendet, um

den Zustand einer Gruppe von Optionsschaltflächen sofort anzuzeigen, wenn der Benutzer eine der Optionsschaltflächen auswählt.

```
Public Class frm0226
   Private Sub optFarbeRot_CheckedChanged( ... ) _
            Handles optFarbeRot.CheckedChanged, _
            optFarbeGrün.CheckedChanged, _
            optFarbeBlau.CheckedChanged
      If optFarbeRot.Checked Then
         lblAnzeige.Text = "Rot"
      ElseIf optFarbeGrün.Checked Then
         lblAnzeige.Text = "Grün"
      Else
         lblAnzeige.Text = "Blau"
      End If
   End Sub
End Class
```

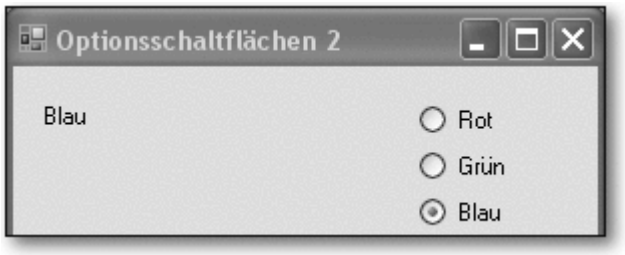

Abbildung 2.27 Mehrere Ereignisse in einer Prozedur

Zur Erläuterung:

- Die Prozedur optFarbeRot_CheckedChanged() wird durch alle drei CheckedChanged-Ereignisse aufgerufen: ... Handles optFarbeRot. CheckedChanged, optFarbeGrün.CheckedChanged, optFarbeBlau. CheckedChanged

2.5.4 Mehrere Gruppen von Optionsschaltflächen

Falls im vorherigen Programm weitere Optionsschaltflächen hinzugefügt wurden, so gilt nach wie vor: Nur eine der Optionsschaltflächen ist ausgewählt.

Container

Benötigt man aber innerhalb eines Formulars mehrere voneinander unabhängige Gruppen von Optionsschaltflächen, wobei in jeder der Gruppen jeweils nur eine Optionsschaltfläche ausgewählt sein soll, so muss man jede Gruppe einzeln in einen Container setzen. Ein Formular ist bereits ein Container, wir benötigen also einen weiteren Container.

Group Box

Als ein solcher Container kann beispielsweise das Steuerelement Gruppe (GroupBox) dienen. Die Optionsschaltfläche, die zu einem (und damit auch in einen) Container gehören soll, muss zuerst erzeugt werden. Anschließend wird sie in die gewünschte GroupBox verschoben. Mit der Zuweisung der Eigenschaft Text der GroupBox gibt man eine Beschriftung an.

Abbildung 2.28 Gruppen von Optionsschaltflächen

```
Public Class frm0227
   Dim AusgabeUrlaubsort As String
   Dim AusgabeUnterkunft As String

   Private Sub optBerlin_CheckedChanged( ... ) Handles _
         optBerlin.CheckedChanged, _
         optParis.CheckedChanged, _
         optRom.CheckedChanged
      ' Urlaubsort
      If optBerlin.Checked Then
         AusgabeUrlaubsort = "Berlin"
      ElseIf optParis.Checked Then
         AusgabeUrlaubsort = "Paris"
      Else
         AusgabeUrlaubsort = "Rom"
      End If
      lblAnzeige.Text = AusgabeUrlaubsort _
         & ", " & AusgabeUnterkunft
   End Sub

   Private Sub optAppartment_CheckedChanged( ... ) _
         Handles optAppartment.CheckedChanged, _
         optPension.CheckedChanged, _
         optHotel.CheckedChanged
      ' Unterkunft
      If optAppartment.Checked Then
         AusgabeUnterkunft = "Appartment"
      ElseIf optPension.Checked Then
         AusgabeUnterkunft = "Pension"
      Else
         AusgabeUnterkunft = "Hotel"
      End If
      lblAnzeige.Text = AusgabeUrlaubsort _
         & ", " & AusgabeUnterkunft
   End Sub
End Class
```

Zur Erläuterung:

- Bei einer Urlaubsbuchung können Zielort und Art der Unterkunft unabhängig voneinander gewählt werden. Es gibt also zwei Gruppen von Optionsschaltflächen, jede in einer eigenen GroupBox.

- Bei Auswahl einer der drei Optionsschaltflächen in einer Gruppe wird jeweils die gleiche Prozedur aufgerufen. In den Prozeduren wird der modulweiten Variablen AusgabeUrlaubs-

ort bzw. AusgabeUnterkunft ein Wert zugewiesen. Anschließend werden die beiden Variablen ausgegeben.

- Die Variablen mussten modulweit deklariert werden, damit sie in der jeweils anderen Prozedur zur Verfügung stehen.

Übung p0228:

Erweitern Sie die Übung p0223. Die Bewegung des Krans soll per Zeitgeber (Timer) gesteuert werden. Der Benutzer wählt zunächst über eine Gruppe von Optionsschaltflächen aus, welche Bewegung der Kran ausführen soll. Anschließend betätigt er den Start-Button. Die Bewegung wird so lange ausgeführt, bis er den Stop-Button drückt oder eine Begrenzung erreicht wurde.

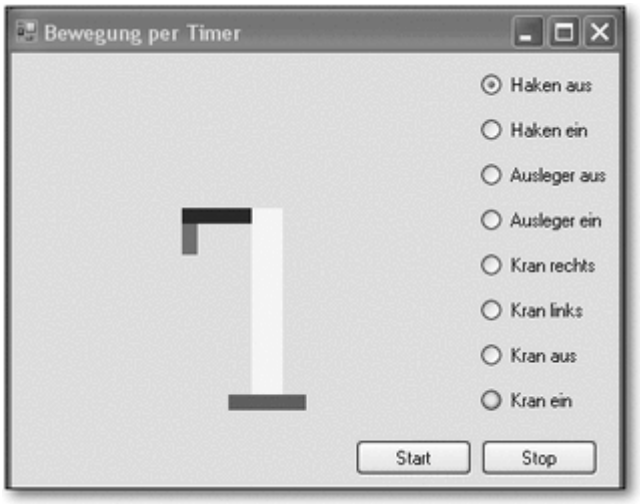

Abbildung 2.29 Übung p0228

2.5.5 Prozedur ohne Ereignis, Modularisierung

Allgemeine Prozedur

Bisher wurden nur Prozeduren behandelt, die mit einem Ereignis zusammenhingen. Darüber hinaus kann man aber auch unabhängige, allgemeine Prozeduren schreiben, die von anderen Stellen des Programms aus aufgerufen werden. Diese Prozeduren können direkt im Codefenster eingegeben werden.

Nachfolgend eine geänderte Version des Programms p0227:

```
...
Private Sub optAppartment_CheckedChanged( ... ) _
      Handles optAppartment.CheckedChanged, _
      optPension.CheckedChanged, _
      optHotel.CheckedChanged
   ' Unterkunft
   If optAppartment.Checked Then
      AusgabeUnterkunft = "Appartment"
   ElseIf optPension.Checked Then
      AusgabeUnterkunft = "Pension"
   Else
      AusgabeUnterkunft = "Hotel"
   End If
   Anzeigen()
End Sub
```

```
   Private Sub Anzeigen()
      lblAnzeige.Text = AusgabeUrlaubsort _
         & ", " & AusgabeUnterkunft
   End Sub
End Class
```

Zur Erläuterung:

- Abgebildet wird nur der zweite Teil der Klasse.
- Am Ende der beiden Ereignisprozeduren optAppartment_Checked Changed() und optBerlin_CheckedChanged() steht jeweils die Anweisung Anzeigen(). Dabei handelt es sich um einen Aufruf der Prozedur Anzeigen().
- Diese Prozedur steht weiter unten. Sie ist nicht direkt an ein Ereignis gekoppelt.

Vorteil dieser Vorgehensweise: Gemeinsam genutzte Programmteile können ausgelagert werden und müssen nur einmal geschrieben werden. Man nennt diesen Vorgang bei der Programmierung auch Modularisierung. In Abschnitt 4.6, »Prozeduren und Funktionen«, wird dieses Thema noch genauer behandelt.

2.6 Schleifen

Schleifen werden in Programmen häufig benötigt. Sie ermöglichen den mehrfachen Durchlauf von Anweisungen. Darin liegt eine besondere Stärke der Programmierung allgemein: die schnelle wiederholte Bearbeitung ähnlicher Vorgänge.

Es gibt die Schleifenstrukturen: Do...Loop, For...Next, For Each... In... und With.

Mithilfe der ersten beiden Strukturen steuert man die Wiederholungen eines Anweisungsblocks (die Anzahl der Schleifendurchläufe). Dabei wird der Wahrheitswert eines Ausdrucks (der Schleifenbedingung) oder der Wert eines numerischen Ausdrucks (Wert des Schleifenzählers) benötigt.

Die Schleife For Each...In... wird meist bei Feldern oder Collections (Auflistungen) eingesetzt. Die Anweisung With dient zur Steuerung einer besonderen Schleife mit nur einem Durchlauf.

2.6.1 For ... Next

Falls die Anzahl der Schleifendurchläufe bekannt oder vor Beginn der Schleife berechenbar ist, sollte man die For... Next-Schleife verwenden. Ihr Aufbau sieht wie folgt aus:

```
For Zähler = Anfang To Ende [ Step = Schritt ]
   [ Anweisungen ]
   [ Exit For ]
   [ Anweisungen ]
Next [ Zähler ]
```

Step

Die Zahlen-Variable Zähler wird zunächst auf den Wert von Anfang gesetzt. Nach jedem Durchlauf wird sie um den Wert von Schritt verändert, also vergrößert oder verkleinert. Falls Step = Schritt nicht angegeben wurde, wird die Variable um 1 vergrößert. Der neue Wert von Zähler wird mit dem Wert von Ende verglichen.

- Falls die Schrittweite positiv ist und der Wert von Zähler nicht größer als der Wert von Ende ist, wird die Schleife wiederum durchlaufen.

- Falls die Schrittweite negativ ist und der Wert von Zähler nicht kleiner als der Wert von Ende ist, wird die Schleife ebenfalls wiederum durchlaufen.

- Falls die Schrittweite positiv ist und der Wert von Zähler größer als der Wert von Ende ist oder falls die Schrittweite negativ ist und der Wert von Zähler kleiner als der Wert von Ende ist, wird die Schleife beendet.

Exit For

Die Anweisung Exit For kann eingesetzt werden, um die Schleife aufgrund einer speziellen Bedingung sofort zu verlassen.

In dem folgenden Programm (p0229) werden durch Aufruf von vier Buttons vier unterschiedliche Schleifen durchlaufen:

```
Public Class frm0229
[ ... ]
   Private Sub cmdSchleife1_Click( ... ) Handles ...
      Dim i As Integer
      lblAnzeige.Text = ""
      For i = 3 To 7
         lblAnzeige.Text &= i & vbCrLf
      Next
   End Sub
[ ... ]
End Class
```

Zur Erläuterung:

- Als Zählervariable dient i.

- Die Schleife wird erstmalig mit i = 3 und letztmalig mit i = 7 durchlaufen.

- Es ist keine Schrittweite angegeben, also wird als Schrittweite 1 genommen.

- Statt Next hätte man zur größeren Deutlichkeit auch Next i schreiben können.

Das Ergebnis zeigt Abbildung 2.30.

```
Public Class frm0229
[ ... ]
   Private Sub cmdSchleife2_Click( ... ) Handles ...
      Dim i As Integer
      lblAnzeige.Text = ""
      For i = 3 To 11 Step 2
         lblAnzeige.Text &= i & vbCrLf
      Next
   End Sub
[ ... ]
End Class
```

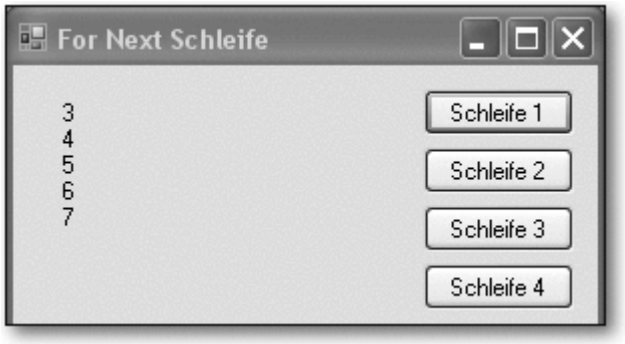

Abbildung 2.30 Erste Schleife

Das Ergebnis zeigt die Abbildung 2.31:

Abbildung 2.31 Zweite Schleife

```
Public Class frm0229
[ ... ]
   Private Sub cmdSchleife3_Click( ... ) Handles ...
      Dim i As Integer
      lblAnzeige.Text = ""
      For i = 7 To 3 Step -1
         lblAnzeige.Text &= i & vbCrLf
      Next
   End Sub
[ ... ]
End Class
```

Das Ergebnis zeigt Abbildung 2.32.

```
Public Class frm0229
[ ... ]
   Private Sub cmdSchleife4_Click( ... ) Handles ...
      Dim d As Double
      lblAnzeige.Text = ""
      For d = 3.5 To 7.5 Step 1.5
         lblAnzeige.Text &= d & vbCrLf
      Next
   End Sub
[ ... ]
End Class
```

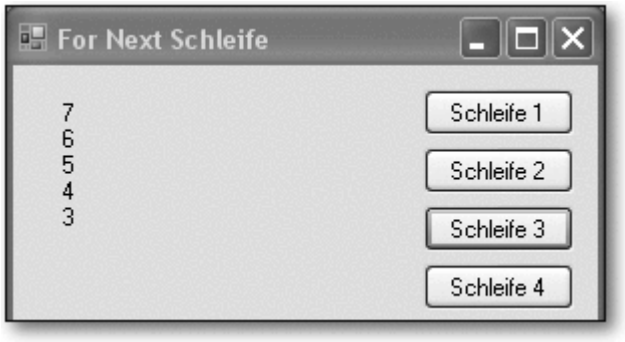

Abbildung 2.32 Dritte Schleife

Das Ergebnis sehen Sie in Abbildung 2.33:

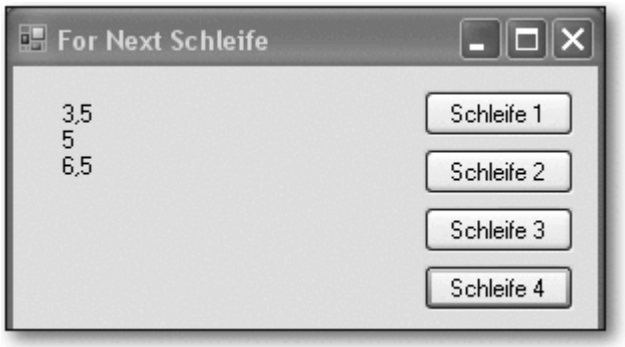

Abbildung 2.33 Vierte Schleife

2.6.2 Do ... Loop

Steuerung über Bedingung

Ist die Anzahl der Schleifendurchläufe nicht bekannt bzw. vor Beginn der Schleife nicht berechenbar, so sollte man die Do ... Loop-Schleife verwenden. Es gibt sie in fünf verschiedenen Varianten:

While

- Do While...Loop: Prüft die Bedingung zum Weiterlaufen der Schleife am Anfang der Schleife.
- Do...Loop While: Prüft die Bedingung zum Weiterlaufen der Schleife am Ende der Schleife.

Until

- Do Until...Loop: Prüft die Bedingung zum Abbruch der Schleife am Anfang der Schleife.
- Do...Loop Until: Prüft die Bedingung zum Abbruch der Schleife am Ende der Schleife.
- Do...Loop: Die Bedingung zum Weiterlaufen oder Abbruch der Schleife wird nicht geprüft, daher ist eine Verzweigung in der Schleife und ein Exit Do zur Beendigung der Schleife notwendig.

Der allgemeine Aufbau sieht wie folgt aus:

```
Do { While | Until } Bedingung
   [ Anweisungen ]
```

```
   [ Exit Do ]
   [ Anweisungen ]
Loop
```

oder

```
Do
   [ Anweisungen ]
   [ Exit Do ]
   [ Anweisungen ]
Loop { While | Until } Bedingung
```

Zufallsgenerator

Im folgenden Programm (p0230) werden alle fünf Möglichkeiten genutzt. Es sollen jeweils so lange Zahlen addiert werden, bis die Summe der Zahlen 5 erreicht. Da die Zahlen durch einen Zufallsgenerator erzeugt werden, ist die Anzahl der Schleifendurchläufe nicht vorhersagbar.

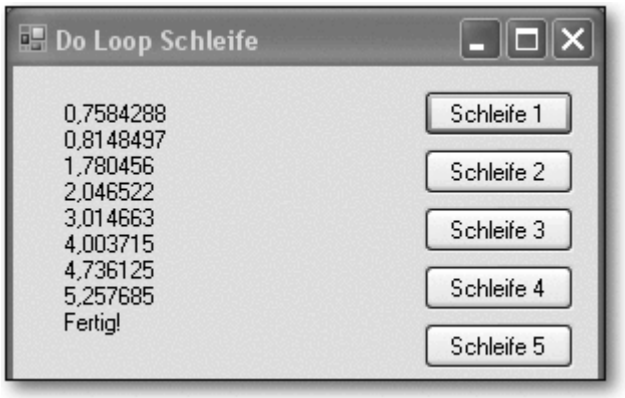

Abbildung 2.34 Do…Loop-Schleife

Randomize, Rnd

Der Zufallszahlengenerator wird mithilfe der Funktion Rnd() realisiert. Diese liefert quasizufällige Zahlen zwischen 0 und 1. Der Zufallszahlengenerator muss mithilfe der Prozedur Randomize() vor der Benutzung initialisiert werden, da andernfalls immer die gleichen »Zufallszahlen« geliefert würden. Die Initialisierung wird pro Programmaufruf einmalig beim Laden des Formulars vorgenommen.

Load, MyBase

Dieser Zeitpunkt wird durch das Ereignis Load gekennzeichnet. Man erstellt den Rahmen dieser Ereignisprozedur, indem man einen Doppelklick auf einer freien Stelle des Formulars ausführt. Die zugehörige Ereignisprozedur behandelt das Ereignis MyBase.Load. Mit MyBase kann man auf die Klasse zugreifen, in der man sich befindet; in diesem Fall ist das die Klasse des Formulars frm0230.

```
Public Class frm0230
   Private Sub frm0230_Load( ... ) Handles MyBase.Load
      Randomize()
   End Sub

   Private Sub cmdSchleife1_Click( ... ) Handles ...
      Dim Summe As Single
      lblAnzeige.Text = ""
      Summe = 0
      Do While Summe < 5
```

```
            Summe += Rnd()
            lblAnzeige.Text &= Summe & vbCrLf
        Loop
        lblAnzeige.Text &= "Fertig!"
    End Sub

    Private Sub cmdSchleife2_Click( ... ) Handles ...
        Dim Summe As Single
        lblAnzeige.Text = ""
        Summe = 0
        Do
            Summe += Rnd()
            lblAnzeige.Text &= Summe & vbCrLf
        Loop While Summe < 5
        lblAnzeige.Text &= "Fertig!"
    End Sub

    Private Sub cmdSchleife3_Click( ... ) Handles ...
        Dim Summe As Single
        lblAnzeige.Text = ""
        Summe = 0
        Do Until Summe >= 5
            Summe += Rnd()
            lblAnzeige.Text &= Summe & vbCrLf
        Loop
        lblAnzeige.Text &= "Fertig!"
    End Sub

    Private Sub cmdSchleife4_Click( ... ) Handles ...
        Dim Summe As Single
        lblAnzeige.Text = ""
        Summe = 0
        Do
            Summe += Rnd()
            lblAnzeige.Text &= Summe & vbCrLf
        Loop Until Summe >= 5
        lblAnzeige.Text &= "Fertig!"
    End Sub

    Private Sub cmdSchleife5_Click( ... ) Handles ...
        Dim Summe As Single
        lblAnzeige.Text = ""
        Summe = 0
        Do
            Summe += Rnd()
            lblAnzeige.Text &= Summe & vbCrLf
            If Summe >= 5 Then Exit Do
        Loop
        lblAnzeige.Text &= "Fertig!"
    End Sub
End Class
```

Zur Erläuterung:

Im Folgenden wird nur die erste Ereignisprozedur cmdSchleife1_ Click() erläutert. Die anderen sind vergleichbar aufgebaut, und auf die unterschiedliche Schleifensteuerung wurde bereits weiter oben eingegangen.

- Der Inhalt des Labels wird von alten Ausgaben gelöscht.

  Summe berechnen

- Die Variable Summe wird zunächst mit dem Wert 0 initialisiert. Dies ist in Visual Basic eigentlich nicht nötig, gehört aber zum guten Programmierstil, da man auf diese Weise den Wert der Variablen zu Beginn der Schleife sicherstellen kann.

- Zu Beginn der Schleife wird geprüft, ob die Summe der Zahlen kleiner als 5 ist. Trifft dies zu, kann die Schleife durchlaufen werden.

    -
    - Hinweis: Bei einer solchen kopfgesteuerten Schleife (es wird im Kopf der Schleife geprüft) kann es vorkommen, dass sie niemals durchlaufen wird.

- Der Wert der Variablen Summe wird um eine Zufallszahl zwischen 0 und 1 erhöht.

- Der Inhalt des Labels wird um den aktuellen Wert der Summe und einen Zeilenumbruch verlängert.

- Man kommt zum Ende der Schleife, zur Anweisung Loop. Sie führt dazu, dass das Programm wieder zu Beginn der Schleife fortsetzt. Es wird wiederum geprüft, ob die Summe der Zahlen kleiner als 5 ist. Sobald dies nicht mehr zutrifft, läuft das Programm hinter der Anweisung Loop weiter.

- Es wird die Ausgabe »Fertig!« erzeugt.

2.6.3 With

Mithilfe von With führt man eine Reihe von Anweisungen für ein einzelnes Objekt durch. Dabei wird der einmal erstellte Bezug zum Objekt mehrfach verwendet. Bei einem längeren Objektnamen ist dies sehr nützlich und übersichtlich. Der Aufbau sieht wie folgt aus:

```
With Objekt
   [ Anweisungen ]
End With
```

Ein Beispiel (p0231):

```
Public Class frm0231
   Private Sub cmdAnzeige_Click( ... ) Handles ...
      With lblAnzeige
         .BorderStyle = BorderStyle.Fixed3D
         .BackColor = Color.Yellow
         .Text = "Test"
         .Location = New Point(50, 100)
      End With
   End Sub
End Class
```

Zur Erläuterung:

- Die Eigenschaften des Labels lblAnzeige werden mithilfe von With geändert.

- Die Eigenschaften Rahmenstil, Hintergrundfarbe, Textinhalt und Position werden nacheinander gesetzt. Dabei muss zu Beginn der Anweisung jeweils nur ein Punkt angegeben werden. Da das Programm sich innerhalb des With-Blocks befindet, ist es klar, auf welches Objekt sich die Änderungen beziehen.

2.6.4 Übungen

Anhand einer Reihe von Übungsaufgaben zu Schleifen (und Verzweigungen) werden im Folgenden einige typische Probleme der Programmierung in Visual Basic trainiert. Der visuelle Teil der Lösung enthält in der Regel nur ein einfaches Textfeld zur Eingabe, einen oder zwei Buttons zum Durchführen der Aufgabe und ein einfaches Label zur Ausgabe.

Übung p0232:

For-Schleife: Schreiben Sie ein Programm mit einer einfachen Schleife, das nacheinander die folgenden Zahlen ausgibt: 35, 32.5, 30, 27.5, 25, 22.5, 20.

Übung p0233:

For-Schleife: Erweitern Sie die vorherige Aufgabe. Am Ende der Zeile sollen Summe und Mittelwert aller Zahlen angezeigt werden.

Abbildung 2.35 Übung p0232 bzw. p0233

Übung p0234:

Do…Loop-Schleife: Schreiben Sie ein Programm, mit dessen Hilfe eine eingegebene Zahl wiederholt halbiert und ausgegeben wird. Das Programm soll beendet werden, wenn das Ergebnis der Halbierung kleiner als 0,001 ist.

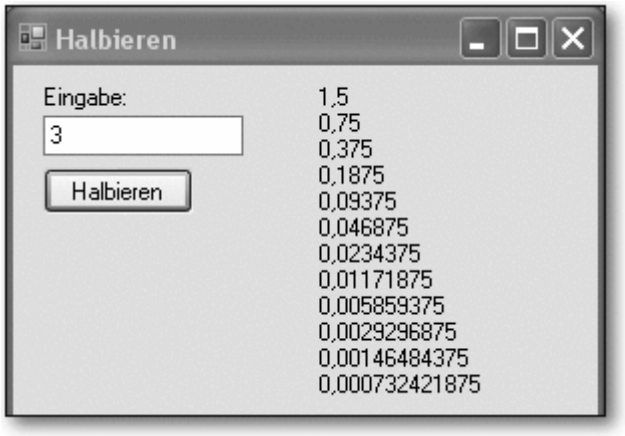

Abbildung 2.36 Übung p0234

Übung p0235:

If… Else-Schleife: Schreiben Sie ein Programm, mit dem das Spiel »Zahlenraten« gespielt werden kann: Per Zufallsgenerator wird eine Zahl zwischen 1 und 100 erzeugt, aber nicht angezeigt. Der Benutzer soll so lange Zahlen eingeben, bis er die Zahl erraten hat. Als Hilfestellung soll jedes Mal ausgegeben werden, ob die eingegebene Zahl größer oder kleiner als die zu ratende Zahl ist.

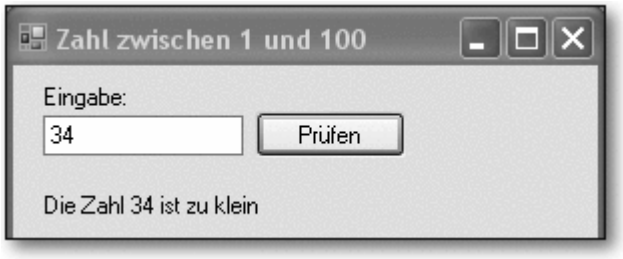

Abbildung 2.37 Übung p0235

Übung p0236:

For-Schleife, If…Else oder Select Case: Erweitern Sie das Programm aus Übung p0222. Schreiben Sie ein Programm, das zu einer Reihe von Gehältern u. a. den Steuerbetrag berechnet und ausgibt. In der folgenden Tabelle sind die Steuersätze angegeben.

Gehalt	Steuersatz
bis einschl. 12.000 €	12 %
von 12.000 bis einschl. 20.000 €	15 %
von 20.000 bis einschl. 30.000 €	20 %
über 30.000 €	25 %

Es sollen für jedes Gehalt von 5.000 € bis 35.000 € in Schritten von 3.000 € folgende vier Werte ausgegeben werden: Gehalt, Steuersatz, Steuerbetrag, Gehalt abzüglich Steuerbetrag. Jedes Gehalt soll mit den zugehörigen Werten in einer eigenen Zeile ausgegeben werden.

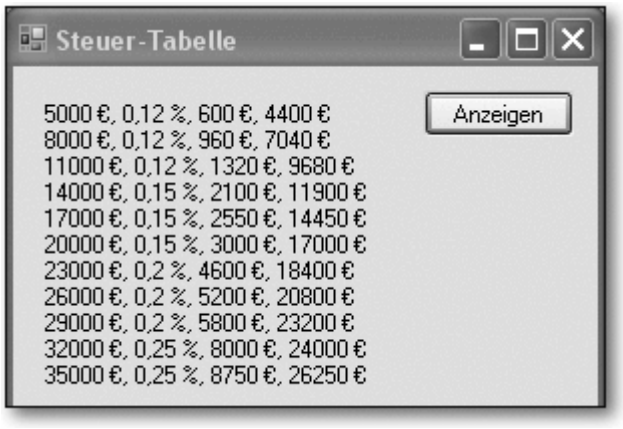

Abbildung 2.38 Übung p0236

2.7 Listenfeld und Kombinationsfeld

Mithilfe der Steuerelemente »Listenfeld« und »Kombinationsfeld« kann eine einfache oder mehrfache Auswahl aus mehreren Möglichkeiten getroffen werden. Im Zusammenhang mit diesen Feldern werden häufig Schleifen benötigt, wie sie im vorherigen Abschnitt behandelt wurden.

2.7.1 Listenfeld

ListBox

Ein Listenfeld (ListBox) zeigt eine Liste mit Einträgen an, aus denen der Benutzer einen oder mehrere auswählen kann. Enthält das Listenfeld mehr Einträge, als gleichzeitig angezeigt werden können, erhält es automatisch einen Scrollbalken.

Items

Die wichtigste Eigenschaft des Steuerelements ListBox ist die Auflistung Items. Sie enthält die einzelnen Listeneinträge. Listenfelder können zur Entwurfszeit gefüllt werden, indem der Eigenschaft Items in einem eigenen kleinen Dialogfeld die Einträge hinzugefügt werden. In der Regel wird man ein Listenfeld aber zur Laufzeit füllen.

2.7.2 Listenfeld füllen

Items.Add

Bisher wurden die Eigenschaften und Ereignisse von Steuerelementen behandelt. Darüber hinaus gibt es jedoch auch spezifische Methoden, die auf diese Steuerelemente bzw. auf deren Eigenschaften angewendet werden können. Beim Listenfeld ist dies u. a. die Methode Add() der Eigenschaft Items. Diese wird am sinnvollsten zum Zeitpunkt des Ladens des Formulars genutzt.

Im nachfolgenden Programm p0237 wird ein Listenfeld für »Italienische Speisen« zu Beginn des Programms mit den folgenden Werten gefüllt: »Spaghetti«, »Grüne Nudeln«, »Tortellini«, »Pizza«, »Lasagne«.

Abbildung 2.39 Listenfeld mit Scrollbalken

```
Public Class frm0237
   Private Sub frm0237_Load( ... ) Handles MyBase.Load
      lstSpeisen.Items.Add("Spaghetti")
      lstSpeisen.Items.Add("Grüne Nudeln")
      lstSpeisen.Items.Add("Tortellini")
      lstSpeisen.Items.Add("Pizza")
      lstSpeisen.Items.Add("Lasagne")
   End Sub
End Class
```

Zur Erläuterung:

- Das Ereignis frm0237_Load wird ausgelöst, wenn das Formular geladen wird, das diesen Namen trägt und dessen Verhalten in der gleichnamigen Klasse beschrieben wird.

- Die einzelnen Speisen werden der Reihe nach dem Listenfeld hinzugefügt. »Lasagne« steht anschließend ganz unten.

2.7.3 Wichtige Eigenschaften

Die folgenden Eigenschaften eines Listenfelds bzw. der Auflistung Items werden in der Praxis häufig benötigt:

- Items.Count gibt die Anzahl der Elemente in der Liste an.

SelectedItem

- SelectedItem beinhaltet das aktuell vom Benutzer ausgewählte Element der Liste. Falls kein Element ausgewählt wurde, ergibt SelectedItem nichts.

- SelectedIndex gibt die laufende Nummer des aktuell vom Benutzer ausgewählten Elements an, beginnend bei 0 für das oberste Element. Falls kein Element ausgewählt wurde, ergibt SelectedIndex den Wert –1.

Items(i)

- Über Items (Index) kann man die einzelnen Elemente ansprechen, das oberste Element ist Items(0).

Das folgende Programm p0238 veranschaulicht alle diese Eigenschaften:

```
Public Class frm0238
   Private Sub frm0238_Load( ... ) Handles MyBase.Load
      [ ... wie oben ... ]
   End Sub

   Private Sub cmdAnzeige_Click( ... ) Handles ...
      lblAnzeige1.Text = "Anzahl: " _
         & lstSpeisen.Items.Count
      lblAnzeige2.Text = "Ausgewählter Eintrag: " _
         & lstSpeisen.SelectedItem
      lblAnzeige3.Text = "Nummer des ausgewählten " _
         & "Eintrags: " & lstSpeisen.SelectedIndex
      lblAnzeige4.Text = "Alle Einträge:" & vbCrLf
      For i = 0 To lstSpeisen.Items.Count - 1
         lblAnzeige4.Text &= lstSpeisen.Items(i) _
            & vbCrLf
      Next
   End Sub
End Class
```

Abbildung 2.40 Anzeige nach Auswahl eines Elements

Zur Erläuterung:

- Die Anzahl der Elemente wird über lstSpeisen.Items.Count ausgegeben, in diesem Fall sind es 5.

- Der ausgewählte Eintrag steht in lstSpeisen.SelectedItem, seine Nummer in lstSpeisen.SelectedIndex.

- Eine For-Schleife dient zur Ausgabe aller Elemente. Sie läuft von 0 bis lstSpeisen.Items.Count - 1. Dies liegt daran, dass bei einer Liste mit fünf Elementen die Elemente mit 0 bis 4 nummeriert sind.

- Die einzelnen Elemente werden mit lstSpeisen.Items(i) angesprochen. Die Variable i beinhaltet bei der Schleife die aktuelle laufende Nummer.

2.7.4 Wechsel der Auswahl

SelectedIndexChanged

Ähnlich wie beim Kontrollkästchen oder bei der Optionsschaltfläche ist das wichtigste Ereignis einer ListBox nicht der Click, sondern das Ereignis SelectedIndexChanged. Dieses Ereignis zeigt nicht nur an, dass die ListBox vom Benutzer bedient wurde, sondern auch, dass sie ihren Zustand geändert hat. Dies kann z. B. auch durch Programmcode geschehen. Eine Ereignisprozedur zu SelectedIndexChanged() wird in jedem Fall durchlaufen, sobald die ListBox (vom Benutzer oder vom Programmcode) geändert wurde.

Allerdings wird der Programmablauf meist so gestaltet, dass bei einem anderen Ereignis die aktuelle Auswahl der ListBox abgefragt wird und anschließend je nach Zustand unterschiedlich reagiert wird.

Das nachfolgende Programm p0239 veranschaulicht diesen Zusammenhang:

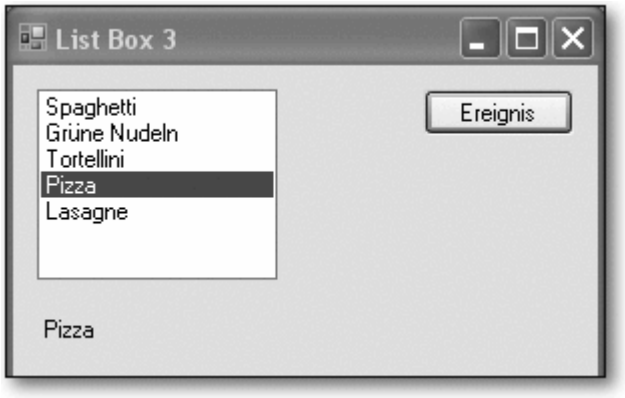

Abbildung 2.41 Anzeige nach dem Ereignis

```
Public Class frm0239
   Private Sub frm0239_Load( ... ) Handles MyBase.Load
       [ ... wie oben ... ]
   End Sub

   Private Sub cmdEreignis_Click( ... ) Handles ...
      lstSpeisen.SelectedIndex = 3
   End Sub

   Private Sub lstSpeisen_SelectedIndexChanged( ... ) _
         Handles ...
      lblAnzeige.Text = lstSpeisen.SelectedItem
   End Sub
End Class
```

Zur Erläuterung:

- In der Ereignisprozedur cmdEreignis_Click() wird die Nummer des ausgewählten Elements auf 3 gesetzt. Dadurch wird in der ListBox »Pizza« ausgewählt. Im Label wird die geänderte Auswahl sofort angezeigt, da das Ereignis lstSpeisen_SelectedIndexChanged ausgelöst wurde.

- In der zugehörigen Ereignisprozedur lstSpeisen_SelectedIndex Changed() wird die Anzeige des ausgewählten Elements ausgelöst. Dieses wird unmittelbar nach der Auswahl angezeigt. Die Auswahl kann durch einen Klick des Benutzers in der Liste oder auch durch Programmcode ausgelöst werden.

2.7.5 Wichtige Methoden

Die Methoden Insert() und RemoveAt() kann man zur Veränderung der Inhalte des Listenfelds nutzen:

Insert

- Mithilfe der Methode Insert() kann man Elemente zum Listenfeld an einer gewünschten Stelle hinzufügen.

RemoveAt

- Die Methode RemoveAt() löscht ein Element an der gewünschten Stelle.

Im nachfolgenden Programm p0240 werden die beiden Methoden eingesetzt, um ein Listenfeld zu verwalten. Es können Elemente eingefügt, gelöscht und geändert werden. Um sicherzustellen, dass es sich hierbei um sinnvolle Operationen handelt, sind jeweils bestimmte Bedingungen zu beachten.

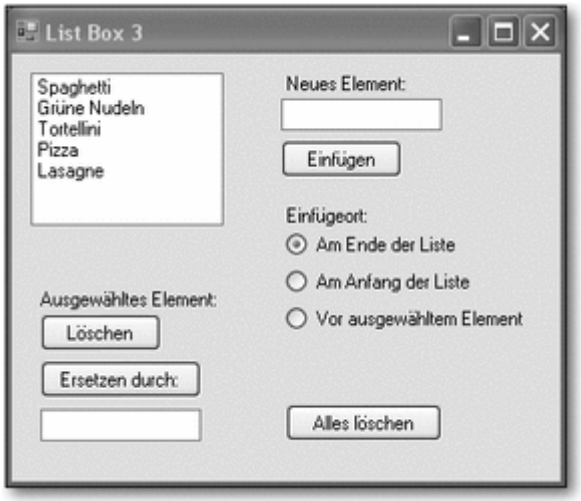

Abbildung 2.42 Verwaltung eines Listenfelds

```
Public Class frm0240
   Private Sub frm0240_Load( ... ) Handles MyBase.Load
      [ ... wie oben ... ]
   End Sub

   Private Sub cmdLöschen_Click( ... ) Handles ...
      If lstSpeisen.SelectedIndex <> -1 Then
         lstSpeisen.Items.RemoveAt(_
            lstSpeisen.SelectedIndex)
      End If
   End Sub

   Private Sub cmdEinfügen_Click( ... ) Handles ...
      If txtNeu.Text = "" Then
         Exit Sub
      End If
      If optAnfang.Checked Then
         lstSpeisen.Items.Insert(0, txtNeu.Text)
      ElseIf optAuswahl.Checked And _
            lstSpeisen.SelectedIndex <> -1 Then
         lstSpeisen.Items.Insert(_
               lstSpeisen.SelectedIndex, _
               txtNeu.Text)
      Else
         lstSpeisen.Items.Add(txtNeu.Text)
      End If
      txtNeu.Text = ""
   End Sub

   Private Sub cmdErsetzen_Click( ... ) Handles ...
      Dim X As Integer
      If txtErsetzen.Text <> "" And _
            lstSpeisen.SelectedIndex <> -1 Then
         X = lstSpeisen.SelectedIndex
         lstSpeisen.Items.RemoveAt(X)
         lstSpeisen.Items.Insert(X, txtErsetzen.Text)
         txtErsetzen.Text = ""
      End If
   End Sub
```

```
   Private Sub cmdAllesLöschen_Click( ... ) Handles ...
      lstSpeisen.Items.Clear()
   End Sub
End Class
```

Zur Erläuterung:

- In der Prozedur cmdLöschen_Click() wird zunächst untersucht, ob ein Element ausgewählt wurde, ob also der Wert von SelectedIndex ungleich -1 ist. Falls dies der Fall ist, wird dieses Element mit der Methode RemoveAt() gelöscht. Diese benötigt den Wert von SelectedIndex zur Auswahl des zu löschenden Elements. Wurde kein Element ausgewählt, geschieht nichts.

- In der Prozedur cmdEinfügen_Click() wird zunächst untersucht, ob in der TextBox etwas zum Einfügen steht. Ist dies der Fall, wird untersucht, welcher Einfügeort über die Optionsschaltflächen ausgesucht wurde.

- Wurde als Einfügeort das Ende der Liste gewählt, so wird der Inhalt der TextBox mit der bekannten Methode Add()am Ende der Liste angefügt.

- In den beiden anderen Fällen wird die Methode Insert() zum Einfügen des Inhalts der TextBox vor einem vorhandenen Listeneintrag genutzt. Diese Methode benötigt den Index des Elements, vor dem eingefügt werden soll. Dies ist entweder der Wert 0, falls am Anfang der Liste eingefügt werden soll, oder der Wert von SelectedIndex, falls vor dem ausgewählten Element eingefügt werden soll.

- Anschließend wird die TextBox gelöscht, damit nicht versehentlich zweimal das gleiche Element eingefügt wird.

- In der Prozedur cmdErsetzen_Click() wird untersucht, ob in der TextBox etwas zum Ersetzen steht und ob ein Element zum Ersetzen ausgewählt wurde. Ist dies der Fall, wird

- der Wert von SelectedIndex in der Variablen X gespeichert,

- das zugehörige Element mit der Methode RemoveAt() gelöscht,

- der neue Text an der gleichen Stelle mit der Methode Insert() eingefügt

- und die TextBox gelöscht, damit nicht versehentlich zweimal das gleiche Element eingefügt wird.

- In der Prozedur cmdAllesLöschen_Click() dient die Methode Clear() zum Leeren der ListBox.

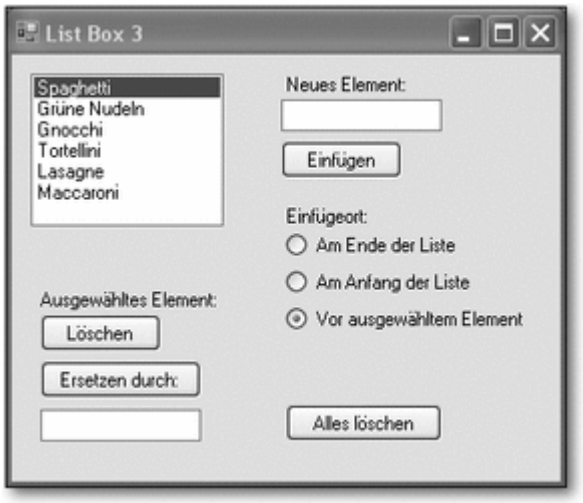

Abbildung 2.43 Nach einigen Veränderungen

2.7.6 Mehrfachauswahl

SelectionMode

Man kann dem Benutzer ermöglichen, gleichzeitig mehrere Einträge aus einer Liste auszuwählen, wie er dies auch aus anderen Windows-Programmen kennt. Dazu wird zur Entwicklungszeit die Eigenschaft SelectionMode auf den Wert MultiExtended gesetzt. Der Benutzer kann anschließend mithilfe der Strg-Taste mehrere einzelne Elemente auswählen oder mithilfe der ⇧-Taste (wie für Großbuchstaben) einen zusammenhängenden Bereich von Elementen markieren.

Hinweis: Nach dem Einfügen einer neuen ListBox in ein Formular steht die Eigenschaft SelectionMode zunächst auf dem Standardwert One, d. h. es kann nur ein Element ausgewählt werden.

SelectedIndices

Die Eigenschaften SelectedIndices und SelectedItems beinhalten die Nummern bzw. die Einträge der ausgewählten Elemente. Sie ähneln in ihrem Verhalten der Eigenschaft Items. Das nachfolgende Programm p0241 verdeutlicht dies:

Abbildung 2.44 Mehrere ausgewählte Elemente

```
Public Class frm0241
   Private Sub frm0241_Load( ... ) Handles MyBase.Load
      [ ... wie oben ... ]
   End Sub
```

```
   Private Sub cmdAnzeigen_Click( ... ) Handles ...
      Dim i As Integer
      lblAnzeige.Text = ""
      For i = 0 To lstSpeisen.SelectedItems.Count - 1
         lblAnzeige.Text &= _
               lstSpeisen.SelectedItems(i) & vbCrLf
      Next
   End Sub
End Class
```

Zur Erläuterung:

SelectedItems(i)

- In der Prozedur cmdAnzeigen_Click() werden alle ausgewählten Elemente mithilfe einer Schleife durchlaufen. Diese Schleife läuft von 0 bis SelectedItems.Count – 1. Die ausgewählten Elemente selbst werden über SelectedItems(i) angesprochen.

2.7.7 Kombinationsfelder

Das Steuerelement Kombinationsfeld (ComboBox) vereinigt die Merkmale eines Listenfelds mit denen eines Textfelds. Der Benutzer kann einen Eintrag aus dem Listenfeldbereich auswählen oder im Textfeldbereich eingeben. Das Kombinationsfeld hat im Wesentlichen die Eigenschaften und Methoden des Listenfelds.

DropDownStyle

Man kann mithilfe der Eigenschaft DropDownStyle zwischen drei Typen von Kombinationsfeldern wählen:

- DropDown: Dies ist der Standard – Auswahl aus einer Liste (Aufklappen der Liste mit der Pfeiltaste) oder Eingabe in das Textfeld. Das Kombinationsfeld hat die Größe einer Text-Box.

- DropDownList: Die Auswahl ist begrenzt auf die Einträge der aufklappbaren Liste, also ohne eigene Eingabemöglichkeit. Dieser Typ Kombinationsfeld verhält sich demnach wie ein Listenfeld, ist allerdings so klein wie eine TextBox. Ein Listenfeld könnte zwar auch auf diese Größe verkleinert werden, aber die Scroll-Pfeile sind dann sehr klein.

- Simple: Die Liste ist immer geöffnet und wird bei Bedarf mit einer Bildlaufleiste versehen. Wie beim Typ DropDown ist die Auswahl aus der Liste oder die Eingabe in das Textfeld möglich. Beim Erstellen eines solchen Kombinationsfelds kann die Höhe wie bei einer ListBox eingestellt werden.

Die Eigenschaft SelectionMode gibt es bei Kombinationsfeldern nicht. Das folgende Programm (p0242) führt alle drei Typen von Kombinationsfeldern vor:

```
Public Class frm0242
   Private Sub frm0242_Load( ... ) Handles MyBase.Load
      cmbWerkzeug1.Items.Add("Zange")
      cmbWerkzeug1.Items.Add("Hammer")
      cmbWerkzeug1.Items.Add("Bohrer")
      cmbWerkzeug1.Items.Add("Schraubendreher")
      [ ... das Gleiche für Feld 2 und 3 ... ]
   End Sub
```

```
   Private Sub cmdAnzeigen1_Click( ... ) Handles ...
      lblAnzeige1.Text = cmbWerkzeug1.Text
   End Sub

   Private Sub cmdAnzeigen2_Click( ... ) Handles ...
      lblAnzeige2.Text = cmbWerkzeug2.SelectedItem
   End Sub

   Private Sub cmdAnzeigen3_Click( ... ) Handles ...
      lblAnzeige3.Text = cmbWerkzeug3.Text
   End Sub
End Class
```

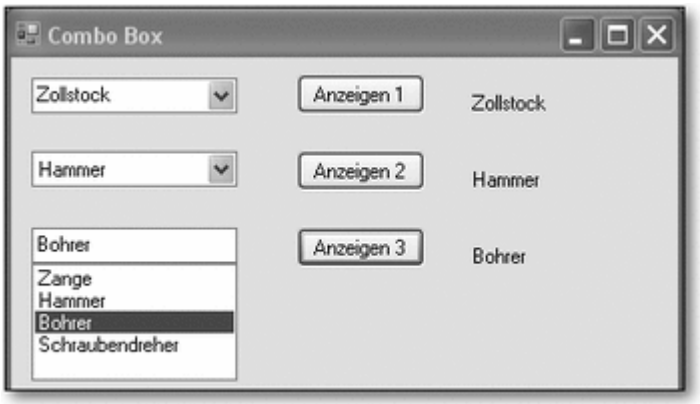

Abbildung 2.45 Drei verschiedene Kombinationsfelder

Zur Erläuterung:

- Das erste Kombinationsfeld hat den DropDownStyle DropDown. Hat der Benutzer einen Eintrag auswählt, so erscheint dieser in der TextBox des Kombinationsfelds. Falls er selber einen Eintrag eingibt, wird dieser ebenfalls dort angezeigt. Die Eigenschaft Text enthält den Inhalt dieser TextBox, also immer den »Wert« des Kombinationsfelds.

- Das zweite Kombinationsfeld hat den DropDownStyle DropDownList. Es gibt also keine TextBox. Wie beim Listenfeld ermittelt man die Auswahl des Benutzers über die Eigenschaft SelectedItem.

- Das dritte Kombinationsfeld hat den DropDownStyle Simple. Im Programm kann es genauso wie das erste Kombinationsfeld behandelt werden. Die Eigenschaft Text beinhaltet also immer den »Wert« des Kombinationsfelds.

Übung p0243:

Schreiben Sie ein Programm, das zwei Listenfelder beinhaltet, in denen jeweils mehrere Elemente markiert werden können. Zwischen den beiden Listenfeldern befinden sich zwei Buttons, jeweils mit einem Pfeil nach rechts bzw. nach links. Bei Betätigung eines der beiden Buttons sollen die ausgewählten Elemente in Pfeilrichtung aus der einen Liste in die andere Liste verschoben werden.

Bei der Lösung kann neben der Eigenschaft SelectedItems z. B. auch die Eigenschaft SelectedIndices genutzt werden. Eine solche Auflistung beinhaltet dann nicht die ausgewählten Einträge, sondern deren Indizes. Mit dem Löschen mehrerer Einträge aus einem Listenfeld sollte man vom Ende der Liste her beginnen. Der Grund hierfür ist: Löscht man eines der vorderen Elemente zuerst, stimmen die Indizes in der Auflistung SelectedIndices nicht mehr.

Abbildung 2.46 Liste vor dem Verschieben

Abbildung 2.47 Liste nach dem Verschieben