PROGRAMMER'S CHOICE

Michael Kofler
Ralf Nebelo

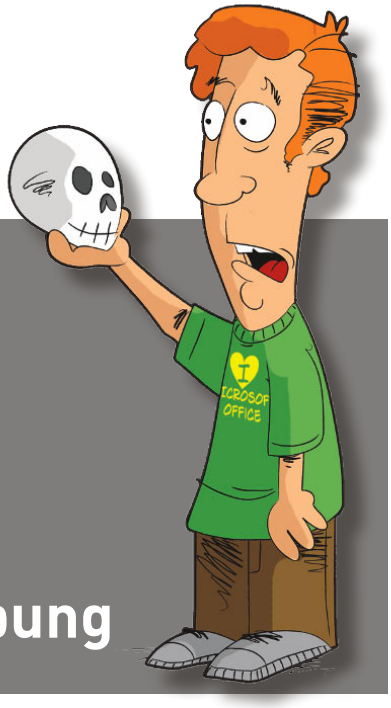
Excel programmieren

Anwendungen entwickeln und Abläufe automatisieren mit Excel 2010 und 2007

ADDISON-WESLEY

Sofort einsetzbares Add-in zur Nachbildung der klassischen Menü- und Symbolleiste

CD



3 Entwicklungsumgebung

Dieses Kapitel beschreibt detailliert die Bedienung der VBA-Entwicklungsumgebung. Diese Entwicklungsumgebung, auch Visual Basic-Editor genannt, steht seit Excel 97 zur Verfügung und wurde seither nur mehr in wenigen Details verändert. Sie erscheint in einem getrennten Fenster mit eigenen Menüs, Symbolleisten etc. Die Entwicklungsumgebung ermöglicht die Eingabe von Programmcode und die Definition neuer Formulare, hilft bei der Fehlersuche, enthält eine Objektreferenz (Objektkatalog) mit Querverweisen zur Hilfe, einen Direktbereich zum Test einzelner Anweisungen etc.

3.1 Komponenten von VBA-Programmen

Ein VBA-Programm ist immer Teil einer Excel-Arbeitsmappe – es ist also unmöglich, Excel-Programme außerhalb einer normalen Excel-Datei zu speichern, zu editieren oder auszuführen. Wenn hier von den Komponenten eines VBA-Programms die Rede

ist, sind also die VBA-Komponenten einer Excel-Datei gemeint, die in der VBA-Entwicklungsumgebung – auch »VBA-Editor« genannt – angezeigt werden.

HINWEIS

Es gibt zwei Sonderfälle zur Speicherung von VBA-Code: Zum einen können Sie den Code eines Moduls oder Dialogs als ASCII-Datei exportieren – Sie können diese Dateien aber nicht ausführen. Zum anderen können Sie eine Excel-Datei als Add-In speichern – dann steht der darin enthaltene VBA-Code allen Dokumenten zur Verfügung, und die Tabellenblätter werden unsichtbar. Obwohl ein Add-In rein optisch wenig mit einer Excel-Arbeitsmappe gemeinsam hat und eine ganz andere Aufgabe erfüllen soll (siehe Abschnitt 15.1), handelt es sich nichtsdestoweniger nur um einen Sonderfall einer normalen Excel-Datei.

Eine Excel-Anwendung kann folgende VBA-Komponenten umfassen:

» Normalen Programmcode (Module): Programmcode mit der Definition von Variablen, Prozeduren und Funktionen wird in sogenannten Modulen gespeichert. Ein Modul ist also eine Gruppe eigens programmierter Prozeduren (Unterprogramme), die in Excel genutzt werden können. In der Entwicklungsumgebung wird ein Modul in einem Textfenster für den Code angezeigt.

» Programmcode zur Definition von neuen Objektklassen (Klassenmodule): Rein optisch sieht ein Klassenmodul wie ein normales Modul aus – es handelt sich also ebenfalls um ein Textfenster mit Programmcode. Der Unterschied besteht darin, dass Klassenmodule zur Definition neuer Objekte dienen. Eine Einführung in die Programmierung von Klassenmodulen gibt Abschnitt 4.5.3.

» Programmcode mit Ereignisprozeduren zu Excel-Objekten: Jedes Excel-Blatt (Tabelle, Diagramm) sowie die gesamte Excel-Arbeitsmappe kennen Ereignisse – etwa den Wechsel von einem Blatt zum anderen, das Speichern oder Drucken der Arbeitsmappe etc. Wenn ein solches vordefiniertes Ereignis eintritt, kann automatisch eine sogenannte Ereignisprozedur ausgelöst werden. Der Programmcode für diese Ereignisprozeduren befindet sich in eigenen Modulen, die jeweils dem entsprechenden Excel-Objekt zugeordnet sind. Detaillierte Informationen zu Ereignisprozeduren gibt Abschnitt 4.4.1.

» Dialoge (`UserForm`): Dialoge bestehen seit Excel 97 aus zwei zusammengehörigen Teilen: dem eigentlichen Dialog mit seinen Steuerelementen und dem Programmcode mit den Ereignisprozeduren zu den Steuerelementen. (Diese Ereignisprozeduren sind zur Verwaltung des Dialogs erforderlich.) Der Entwurf und die Verwaltung von Dialogen sind Thema von Kapitel 7.

» Verweise: Solange Sie nur die Excel-Standardobjekte verwenden, brauchen Sie sich um Verweise nicht zu kümmern. Sobald Sie aber Objekte verwenden möchten, die in externen Objektbibliotheken definiert sind (etwa in der ADO-Bibliothek zur Datenbankprogrammierung), müssen Sie diese mit Extras|Verweise aktivieren. Die Verweise auf die genutzten Objektbibliotheken werden in der Excel-Datei gespeichert.

Die ersten vier Punkte dieser Aufzählung haben eine Gemeinsamkeit: Der VBA-Code wird in immer gleich aussehenden Codefenstern angezeigt. Die Werkzeuge zur Codeeingabe und zur Fehlersuche sind also in jedem Fall dieselben.

3.2 Komponenten der Entwicklungsumgebung

Primäre Aufgabe der Entwicklungsumgebung ist es, die Eingabe von Programmcode und die Definition von Formularen zu ermöglichen. Seit Excel 97 ist die VBA-Entwicklungsumgebung nicht mehr in Excel integriert, sondern verhält sich beinahe wie ein eigenständiges Programm. Die Entwicklungsumgebung wird in Excel durch Entwicklertools|Visual Basic aufgerufen und erscheint dann in einem eigenen Fenster.

HINWEIS

Falls Sie die standardmäßig ausgeblendete Befehlsregisterkarte Entwicklertools noch nicht sichtbar gemacht haben, holen Sie das wie folgt nach: Öffnen Sie die Registerkarte Datei, und wählen Sie Optionen. Klicken Sie links im Dialogfeld auf Menüband anpassen, schalten Sie im rechten Listenfeld das Kontrollkästchen vor Entwicklertools ein, und schließen Sie das Dialogfeld mit OK.

Alternativ können Sie die Entwicklungsumgebung auch mit der Tastenkombination Alt+F11 aktivieren.

Statt mit Entwicklertools|Visual Basic bzw. mit Alt+F11 kann der Wechsel von Excel in die Entwicklungsumgebung auch durch ein Symbol in der *Symbolleiste für den Schnellzugriff* erfolgen: Führen Sie in Excel Datei|Optionen|Symbolleiste für den Schnellzugriff aus, stellen Sie das linke Listenfeld auf »Alle Befehle« ein, markieren Sie in der Liste darunter den Befehl »Visual Basic«, und klicken Sie auf Hinzufügen. Schließen Sie das Dialogfeld mit OK. Künftig reicht ein einfacher Mausklick aus, um in das Fenster der Entwicklungsumgebung zu springen.

Zu beinahe allen Komponenten der Entwicklungsumgebung sind Kontextmenüs definiert, die mit der rechten Maustaste aufgerufen werden und eine effiziente Ausführung der wichtigsten Kommandos ermöglichen. Probieren Sie es aus!

HINWEIS

Der Wechsel zwischen Excel und der Entwicklungsumgebung funktioniert nur, wenn in der gerade aktiven Komponente kein Dialog geöffnet ist. In Excel darf keine Zelle bzw. kein Objekt bearbeitet werden. Ansonsten wird der Blattwechsel ohne Fehlermeldung verweigert.

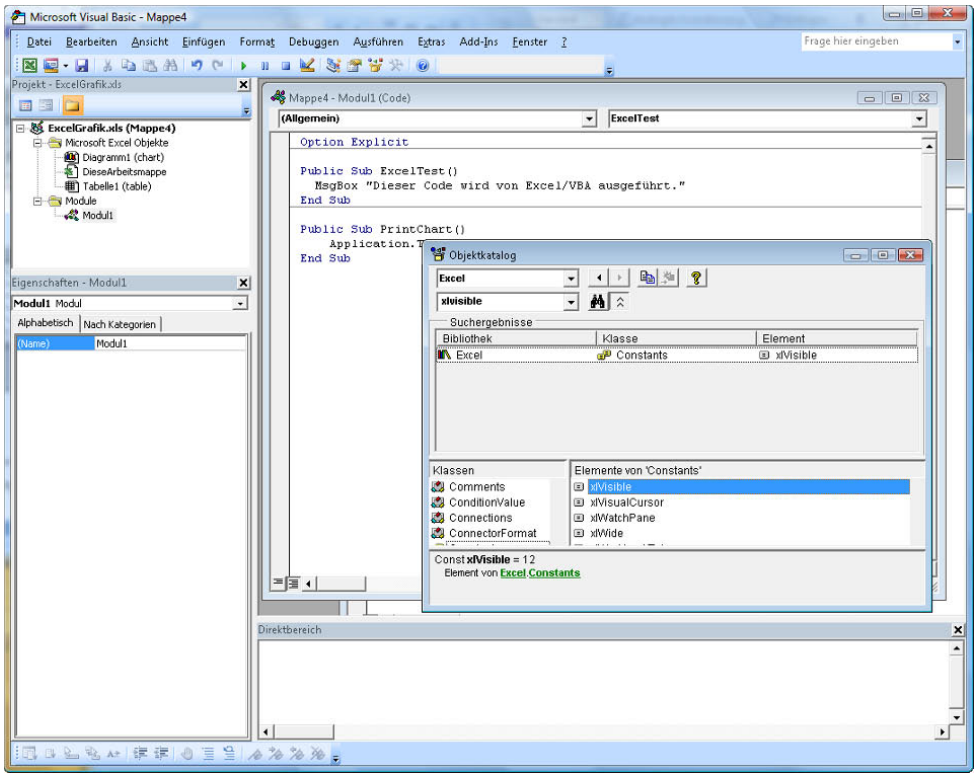


Abbildung 3.1: Die VBA-Entwicklungsumgebung

KAPITEL

Nicht alle Schritte zur Programmentwicklung werden tatsächlich in der VBA-Entwicklungsumgebung durchgeführt. Beispielsweise wird die Makroaufzeichnung direkt in Excel gesteuert. Aus diesem Grund, und um thematisch zusammengehörige Themen gemeinsam zu behandeln, finden Sie weitere Informationen zur Entwicklungsumgebung in anderen Kapiteln:

Objektkatalog, Bibliotheksverweise: Abschnitt 4.3

Hilfsmittel zur Fehlersuche: Abschnitt 6.1

Dialogeditor: Abschnitt 7.3

Projektfenster

Das Projektfenster (Ansicht|Projekt-Explorer oder Strg+R dient zur Orientierung in Excel-Programmen. Zu jeder geladenen Excel-Datei wird im Projektfenster eine Gruppe mit allen dazugehörigen Modulen und Dialogen angezeigt. Durch einen Doppelklick auf den jeweiligen Eintrag werden die Komponenten in einem Fenster angezeigt und können bearbeitet werden.

Die einzelnen Komponenten eines Projekts können wahlweise alphabetisch geordnet oder wie in Bild 3.1 thematisch gruppiert werden. Die Umschaltung erfolgt durch das dritte Symbol im Projektfenster (mit der irreführenden Bezeichnung Ordner wechseln).

TIPP

Wenn Sie mehrere Excel-Dateien gleichzeitig bearbeiten, können Sie einzelne Dateien (»Projekte«) im Projektfenster zusammenklappen (Klick auf das Symbol + oder -). Alle Fenster dieses Projekts werden damit ausgeblendet. Das kann die Orientierung in der Entwicklungsumgebung erheblich erleichtern.

In der Defaulteinstellung werden das Projektfenster und die meisten anderen Komponenten nicht als frei verschiebbare Fenster angezeigt, sondern sind am Randbereich der Entwicklungsumgebung fixiert. Das ist nur dann praktisch, wenn Sie mit einem großen Monitor arbeiten. Andernfalls wird der zur Verfügung stehende Platz schlecht genutzt. Um die Komponenten der Entwicklungsumgebung frei platzieren zu können, führen Sie Extras|Optionen|Verankern aus und deaktivieren sämtliche Optionen dieses Dialogblatts.

Eigenschaftsfenster

Im Eigenschaftsfenster (Ansicht|Eigenschaftsfenster oder F4) können diverse Merkmale des gerade aktuellen Objekts eingestellt werden. Als »Objekte« gelten Module ebenso wie Dialoge und die darin enthaltenen Steuerelemente. Die größte Rolle spielt das Eigenschaftsfenster beim Entwurf neuer Dialoge: Jedes Element eines solchen Dialogs kennt Dutzende von Eigenschaften. Bei normalen Codemodulen kann dagegen nur der Name des Moduls eingestellt werden. Dieser Name darf keine Leerzeichen enthalten. Bei Objektmodulen weicht der VBA-Name im Regelfall vom Excel-Blattnamen ab.

Wie im Projektfenster können auch im Eigenschaftsfenster die Einträge alphabetisch oder nach Gruppen geordnet werden. Die Umschaltung erfolgt hier allerdings mit Blattregistern und ist nur dann sinnvoll, wenn Objekte sehr viele Eigenschaften unterstützen.

Falls Sie Steuerelemente direkt in Tabellenblätter einbetten, können Sie das Eigenschaftsfenster auch in Excel benutzen. Der Aufruf in Excel erfolgt allerdings nicht mit F4, sondern über den Kontextmenüeintrag Eigenschaften bzw. über den Eigenschaften-Befehl der Symbolleiste Entwicklertools.

Objektkatalog

Die Programmierung in Excel basiert auf mehreren Objektbibliotheken, deren wichtigste die `Excel`-Bibliothek ist. Jede Bibliothek ist mit zahlreichen Objekten ausgestattet, die Objekte wiederum mit vielen Eigenschaften, Methoden und Ereignissen. Die einzige Chance, in dieser Fülle von Schlüsselwörtern den Überblick zu bewahren, bietet der Objektkatalog, der mit Ansicht|Objektkatalog bzw. F2 angezeigt wird.

TIPP

Wenn sich der Cursor gerade über einem Schlüsselwort im Codefenster befindet, wird danach mit Shift+F2 automatisch im Objektkatalog gesucht.

Der Objektkatalog bietet in vielen Situationen auch den schnellsten Weg zur Hilfe.

Im Objektkatalog werden sowohl die Objekte aller aktivierten Bibliotheken als auch (in fetter Schrift) alle in Modulen selbst definierten Funktionen und Prozeduren angezeigt. Im Listenfeld links oben können Sie die Anzeige auf Objekte einer bestimmten Bibliothek einschränken. Das ist besonders praktisch, wenn die Suche nach einer Zeichenkette sehr viele Ergebnisse liefert. Eine Suche führen Sie durch, indem Sie im zweiten Listenfeld eine Zeichenkette eingeben und Return drücken.

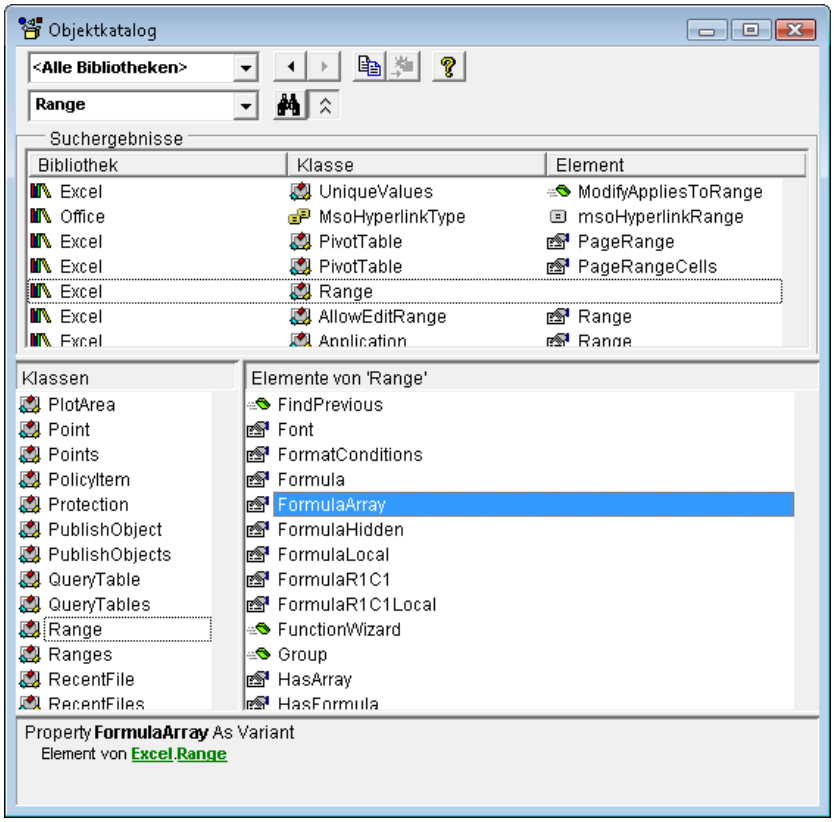


Abbildung 3.2: Der Objektkatalog

Normalerweise werden im Katalog nur »offiziell« unterstützte Schlüsselwörter angezeigt. Daneben gibt es eine Menge verborgener Schlüsselwörter, die entweder intern verwendet werden oder aus Gründen der Kompatibilität zu früheren Versionen aufgenommen wurden. Mit dem Kontextmenükommando Verborgene Elemente anzeigen können Sie auch diese Schlüsselwörter in grauer Schrift anzeigen.

TIPP

Normalerweise sind alle Schlüsselwörter alphabetisch geordnet. Durch das Kontextmenükommando Elemente gruppieren erreichen Sie, dass die Einträge stattdessen in Gruppen geordnet werden, d. h. zuerst alle Eigenschaften, dann die Methoden und schließlich die Ereignisse. Im Regelfall ist diese Form der Anzeige übersichtlicher.

Editoroptionen

Durch Extras|Optionen wird ein vierblättriger Dialog für diverse Einstellungen der Entwicklungsumgebung angezeigt. Die meisten Einstellungen sind leicht verständlich; zu den anderen einige Anmerkungen:

Automatische Syntaxüberprüfung (Blatt Editor): Wenn diese Option aktiviert ist, wird nach der Eingabe einer fehlerhaften Zeile eine Fehlermeldung angezeigt. Während der ersten VBA-Gehversuche ist das vielleicht ganz nützlich, nach ein paar Tagen werden die ständigen Fehlermeldungen aber lästig. Wenn Sie die Option deaktivieren, werden fehlerhafte Zeilen immer noch in roter Farbe angezeigt, was vollkommen ausreichend ist.

Variablendeklaration erforderlich: Wenn diese Option aktiviert ist, wird in jedes neue Modul die Zeile `Option Explicit` eingefügt. Das bedeutet, dass Sie nur Variablen verwenden können, die Sie mit `Dim` deklariert haben. Diese Option vermeidet Programmfehler aufgrund von Tippfehlern und führt zu einem korrekten Code. Unbedingt aktivieren!

Elemente automatisch auflisten, Automatische QuickInfo, Automatische Datentipps: Diese drei Optionen geben an, ob im Codefenster automatisch Informationen über die erlaubten Methoden und Eigenschaften, den aktuellen Inhalt von Variablen und über die erlaubten Parameter eingeblendet werden. Lassen Sie die Optionen in der Defaulteinstellung (also aktiviert) – die Informationen sind ausgesprochen nützlich.

Standardmässig ganzes Modul anzeigen: Diese Option bewirkt, dass im Codefenster nicht nur eine einzelne Prozedur, sondern alle Prozeduren des gesamten Moduls angezeigt werden.

Blatt Editorformat: Hier können Sie den gewünschten Zeichensatz sowie die Farben für verschiedene Syntaxelemente einstellen.

Allgemeine Optionen

Ausblenden des Projekts schliesst Fenster (Blatt Allgemein): Wenn diese Option aktiviert ist, werden alle Fenster eines Projekts geschlossen, sobald das Projekt im Projektfenster zusammengeklappt wird (Klick auf Minussymbol). Beim Aufklappen erscheinen die Fenster wieder. Die Option dient dazu, auch bei mehreren Projekten gleichzeitig eine gewisse Ordnung in der Entwicklungsumgebung zu halten.

Bearbeiten und Fortfahren: Bei manchen Änderungen im Code – etwa bei der Deklaration neuer Variablen – müssen alle aktuellen Variableninhalte gelöscht werden. Wenn die Option Benachrichtigen vor Zustandsänderung aktiviert ist, werden Sie vor der Durchführung solcher Änderungen gewarnt.

Bei jedem Fehler unterbrechen: Diese Option hebt die Wirkung von Fehlerbehandlungsroutinen auf. Trotz `On-Error`-Anweisungen wird das Programm unterbrochen. Die Option ist manchmal zur Fehlersuche sehr praktisch (siehe auch Kapitel 6).

In Klassenmodul/Bei nicht verarbeiteten Fehlern unterbrechen: Die beiden Optionen führen nur dann zu unterschiedlichen Resultaten, wenn Sie Klassenmodule entwickeln. Wenn in einem Klassenmodul ein Fehler auftritt, wird das Programm im ersten Fall im Klassenmodul und im zweiten Fall dort unterbrochen, wo die Methode oder Eigenschaft der Klasse aufgerufen wurde, die den Fehler verursacht hat (siehe auch Abschnitt 4.5).

Kompilieren: VBA-Programme werden automatisch zu einem Pseudocode kompiliert, der effizienter ausgeführt werden kann als der zugrunde liegende ASCII-Code. (Es handelt sich aber nicht um einen Maschinencode, wie er von C-Compilern erzeugt wird.) Die beiden Kompilieren-Optionen steuern, wann kompiliert wird. Die Defaulteinstellung (beide Optionen aktiviert) bedeutet, dass sofort mit der Programmausführung begonnen wird und nur jene Prozeduren kompiliert werden, die gerade benötigt werden. Der Vorteil: ein rascher Programmstart. Der Nachteil: Manche offensichtlichen Fehler werden erst spät gemeldet. Bei größeren Projekten ist es zumeist sinnvoll, die Optionen zu deaktivieren, weil dann mögliche Syntaxfehler im Code sofort gemeldet werden (und nicht irgendwann später, wenn die Prozedur erstmalig benötigt wird).

Blatt Verankern: Wie bereits oben in einem Tipp erwähnt, gelten in der Defaultkonfiguration die meisten Komponenten der Entwicklungsumgebung als sogenannte verankerte Fenster. Diese Fenster kleben gewissermaßen an einem Ende der Entwicklungsumgebung. Wenn Sie mit einem ausgesprochen kleinen Monitor arbeiten, können Sie alle Kontrollkästchen dieses Dialogblatts deaktivieren. Anschließend lassen sich alle Fenster ohne Einschränkungen verschieben und überlappend anordnen.

Projekteigenschaften

Mit Extras|Eigenschaften von VBAProject wird ein Dialog zur Einstellung der Eigenschaften des gerade aktuellen Projekts angezeigt. Im Blatt Allgemein können Sie eine Kurzbeschreibung des Projekts und den Dateinamen einer dazugehörigen Hilfedatei angeben. Im Blatt Schutz können Sie den VBA-Teil einer Excel-Datei ausblenden und durch ein Passwort absichern.

Welchen Stellenwert Microsoft dem Passwortschutz in Excel gibt, hat man beim Versionswechsel von Excel 7 auf Excel 97 gesehen. In Excel 7 ausgeblendete und per Passwort abgesicherte Module waren in Excel 97 ohne Weiteres jedermann zugänglich! Der Passwortschutz der Excel-Version 2003 war zwar deutlich besser, ließ sich aber ebenfalls knacken beziehungsweise umgehen. Trotz weiterer Verbesserungen gilt das leider auch für Excel 2007 und 2010. Es gibt diverse kommerzielle Tools, die den vermeintlichen Schutz in wenigen Sekunden aushebeln.

Bedingte Kompilierung

Manchmal kommt es vor, dass Sie parallel zu einem Programm eine zweite Version verwalten möchten (etwa eine Demo-Version mit eingeschränkten Merkmalen oder eine Debug-Version mit zusätzlichen Sicherheitsabfragen). Dazu können Sie in Extras|Eigenschaften von VBAProject|Allgemein im Textfeld Argumente für bedingte Kompilierung eine Konstante definieren, beispielsweise `demo=1`. Im Programmcode können Sie den Inhalt der Konstanten dann mit `#If`-Anweisungen auswerten.

Je nach Ergebnis der `#If`-Abfrage wird entweder der eine oder der andere Zweig ausgeführt. Im Unterschied zu normalen `If`-Abfragen erfolgt die Unterscheidung zwischen den beiden Varianten allerdings schon bei der Kompilierung. Das Kompilat enthält nur eine Variante und keine `#If`-Abfragen, es ergibt sich also kein Geschwindigkeitsnachteil. Die folgenden Zeilen zeigen, wie Programmcode mit `#If`-Anweisungen aussehen kann:

```
Sub Command1_Click()
  #If demo Then
    MsgBox "In der Demoversion kann nichts gespeichert werden"
  #Else
    ' ... Programmcode zum Speichern
  #End If
End Sub
```

3.3 Codeeingabe in Modulen

In jedem Codefenster kann VBA-Code eingegeben werden. Wie bereits erwähnt, gibt es zahlreiche Objekte, die mit Code ausgestattet werden: die Excel-Datei als Ganzes (Modul *Diese Arbeitsmappe* im Projektfenster), jedes Tabellenblatt (*Tabelle 1, Tabelle 2* etc.), Dialoge, normale Module und Klassenmodule.

Während die Objekte *Diese Arbeitsmappe* und *Tabelle n* durch Excel vorgegeben sind, müssen Dialoge und Module durch Einfügen|UserForm, Einfügen|Modul und Einfügen|Klassenmodul in der Entwicklungsumgebung erst erzeugt werden. (Bei der Makroaufzeichnung wird automatisch ein Modul erzeugt.)

Anschließend sollten Sie dem Dialog bzw. Modul einen aussagekräftigen Namen geben. Dazu müssen Sie das Eigenschaftsfenster F4 verwenden. Leerzeichen und die meisten Sonderzeichen sind dabei verboten.

Während der Name bei normalen Modulen hauptsächlich die Orientierung in großen Projekten erleichtert, ist der Name von Dialogen und Klassenmodulen auch für die Codeausführung wichtig. Nachträgliche Änderungen führen dann zu zusätzlichem Aufwand und sollten möglichst vermieden werden.

Erste Experimente

Für erste VBA-Experimente öffnen Sie in Excel eine neue Arbeitsmappe, wechseln mit Alt+F11 in die Entwicklungsumgebung und fügen dort mit Einfügen|Modul ein neues Modul ein. Das kürzeste mögliche VBA-Makro, das Sie zu Testzwecken eingeben können, sieht folgendermaßen aus:

```
Sub beispiel()
  Debug.Print "Mein erstes Programm!"
End Sub
```

Mit der Methode `Print`, die auf das Objekt `Debug` angewendet wird, gibt dieses Programm den Text »Mein erstes Programm« im Direktbereich aus. Sie können dieses Programm einfach mit F5 starten (dazu muss sich der Eingabecursor innerhalb der Prozedur befinden). Das Ergebnis wird im Direktbereich angezeigt. Da dieses Fenster normalerweise unsichtbar ist, muss es über Ansicht|Direktfenster oder mit Strg+G aktiviert werden.

Wenn Sie das Programm eingeben, werden Sie feststellen, dass Excel einige Programmteile – nämlich die Schlüsselwörter `Sub`, `Debug`, `Print`, `End` und `Sub` – farbig hervorhebt. Das erhöht nicht nur die Übersichtlichkeit, es deutet auch darauf hin, dass die Programmzeilen syntaktisch korrekt sind.

Falls Excel bei der Eingabe einer Programmzeile einen Fehler erkennt, meldet es sich mit einer Fehlermeldung. Sie können diese Meldung vorläufig ignorieren; die gesamte Zeile wird dann rot dargestellt. Das Makro kann aber erst ausgeführt werden, wenn alle Syntaxfehler beseitigt sind.

Automatische Vervollständigung von Schlüsselwörtern und Variablennamen

Die Codeeingabe wird durch sogenannte *IntelliSense*-Funktionen erleichtert. So werden während der Eingabe automatisch kleine Listenfelder angezeigt, die alle derzeit gültigen Vervollständigungen der Eingabe anzeigen. Bei der Eingabe von Funktionen bzw. Methoden wird deren Parameterliste angezeigt etc.

IntelliSense hat einige neue Tastenkürzel mit sich gebracht: Wenn gerade kein Listenfeld angezeigt wird, können Sie mit Strg+SPACE das gerade eingegebene Schlüsselwort (oder den Variablennamen) vervollständigen. Wenn es dabei mehrere Möglichkeiten gibt, erscheint automatisch das Listenfeld.

Wenn das Listenfeld bereits angezeigt wird, können Sie mit den Cursortasten den gewünschten Eintrag auswählen. ⇆ (nicht Return!) vollendet die Auswahl. Mit Esc entkommen Sie dem Listenfeld und können die manuelle Eingabe fortsetzen (beispielsweise zur Eingabe eines noch nicht definierten Variablennamens).

Definition neuer Prozeduren

Wenn Sie eine neue Prozedur schreiben möchten, bestehen dazu mehrere Möglichkeiten. Bei allgemeinen Prozeduren (Unterprogrammen, Funktionen) können Sie mit Einfügen|Prozedur eine Schablone für eine neue Prozedur per Mausklick erstellen. (Die Bedeutung der Schlüsselwörter `Sub`, `Function`, `Property`, `Public`, `Private` werden in Abschnitt 4.2 genau beschrieben.)

Wenn Sie ein wenig Übung und Erfahrung mit Visual Basic haben, werden Sie die Definition einer neuen Prozedur noch schneller erledigen, indem Sie die Anweisungen `Function Name` oder `Sub Name` im Codefenster eingeben. Visual Basic vervollständigt die Prozedurdefinition automatisch durch `End Function` oder `End Sub`.

Cursorbewegung im Programmcodefenster

Der Textcursor kann innerhalb eines Unterprogramms bzw. einer Funktion wie gewohnt mit den Cursortasten bewegt werden. Bild ↑ und Bild ↓ bewegen den Textcursor seitenweise durch eine Prozedur. Wenn der Cursor bereits am Anfang bzw. am Ende des Unterprogramms steht, wird die vorangegangene bzw. die nächste Prozedur angezeigt. (Die Reihenfolge der Prozeduren orientiert sich an der Reihenfolge, in der die Prozeduren definiert wurden.)

Strg + Bild ↑ und Strg + Bild ↑ bzw. Strg + Bild ↓ und Strg + Bild ↓ zeigen unabhängig von der aktuellen Position des Textcursors in jedem Fall das vorige bzw. nächste Unterprogramm an. F6 wechselt den aktiven Ausschnitt, wenn das Fenster geteilt ist.

⇧ + F2 bewegt den Cursor zum Code der Prozedur, auf dessen Namen der Cursor gerade steht (Kommando Ansicht|Definition). Wenn die betroffene Prozedur in einer anderen Datei des Projekts definiert ist, wechselt Visual Basic automatisch in das betreffende Codefenster. Strg + ⇧ + F2 springt zurück zur vorherigen Position. Visual Basic verwaltet dazu einen mehrstufigen Puffer für die Rücksprungpositionen.

Zum raschen Springen zu einem anderen Programmteil können Sie schließlich auch den Objektkatalog verwenden, in dem (unter anderem) sämtliche von Ihnen programmierte Prozeduren verzeichnet sind – siehe den folgenden Abschnitt.

Blöcke ein- und ausrücken

Damit der Programmcode leichter zu lesen ist, werden Blöcke innerhalb von Verzweigungen und Schleifen normalerweise eingerückt (wie in allen Programmlistings dieses Buchs). Die Einrückungen erfolgen nicht automatisch, sondern müssen durch die Eingabe von Leer- oder Tabulatorzeichen vorgenommen werden. Wenn Sie später die Struktur des Programms ändern (z. B. durch eine zusätzliche Sicherheitsabfrage), müssen Sie oft zahlreiche Zeilen ein- oder ausrücken. Anstatt das für jede Zeile manu-

ell zu erledigen, können Sie sich von Visual Basic helfen lassen: Markieren Sie den gesamten Zeilenblock mit der Maus, und geben Sie dann ⇥ bzw. ⇧+⇥ ein. Visual Basic rückt den gesamten Block um eine Tabulatorposition ein oder aus.

Die Tabulatorweite kann im Optionsfenster (Extras|Optionen|Editor) beliebig eingestellt werden – sogar auf ein einziges Zeichen. Die Defaulteinstellung lautet vier Zeichen, in diesem Buch wurden aber nur zwei Zeichen verwendet, was zu einem weniger stark auseinandergezogenen Programmcode führt. (Visual Basic arbeitet übrigens nicht mit echten Tabulatoren. Die Tabulatorweite gibt nur an, wie viele Leerzeichen durch ⇥ eingefügt werden.)

Variablendeklaration

In leeren Modulen steht in der ersten Zeile zumeist die Anweisung `Option Explicit`. Die Anweisung bewirkt, dass alle Variablen vor ihrer Verwendung deklariert werden müssen. Wenn die Anweisung nicht automatisch erscheint, sollten Sie in Extras|Optionen|Editor den Eintrag Variablendeklaration erforderlich aktivieren. Excel fügt die Anweisung dann in neuen Modulen automatisch ein. (In bereits geöffneten Modulen müssen Sie die beiden Schlüsselwörter selbst eintippen.)

Kommentare

Ein Hochkomma »'« ⇧+# leitet Kommentare ein. Kommentare sind sowohl am Zeilenanfang als auch im Anschluss an eine Anweisung erlaubt. Das Kommentarzeichen kann unter anderem dazu verwendet werden, fehlerhafte Zeilen vorläufig in Kommentare zu verwandeln. Kommentare werden üblicherweise grün angezeigt.

TIPP

Während des Tests eines Programms ist es oft praktisch, einige Zeilen Code vorübergehend durch vorangestellte Kommentarzeichen zu deaktivieren. Die Entwicklungsumgebung sieht zum Auskommentieren mehrerer Zeilen bzw. zum Widerrufen dieses Kommandos zwar zwei Symbole in der Bearbeiten-Symbolleiste, aber keine Menükommandos vor. Diesen Mangel können Sie mit Ansicht|Symbolleisten|Anpassen leicht beheben.

Mehrzeilige Anweisungen

Sehr lange Anweisungen können auf mehrere Zeilen verteilt werden. Dazu müssen an einer beliebigen Stelle (aber nicht *in* einem Schlüsselwort) ein Leerzeichen und anschließend der Unterstrich »_« eingegeben werden. Beispiel:

```
Selection.Sort Key1:=Range("A3"), Order1:= xlAscending, _
  Header:=xlGuess, OrderCustom:=1, MatchCase:=False, _
  Orientation:=xlTopToBottom
```

HINWEIS

Variablen können so deklariert werden, dass sie nur in einer Prozedur, im ganzen Modul oder im gesamten Programm (global) verwendet werden können. Details zum Gültigkeitsbereich von Variablen finden Sie in Abschnitt 4.2.2.

HINWEIS

Wenn Sie mehrzeilige Anweisungen mit Kommentaren versehen, dürfen Sie das erst in der letzten Zeile der Anweisung tun. Die folgende Anweisung ist syntaktisch falsch:

```
Selection.Sort Key1:=Range("A3"), _   'nicht erlaubt!
  Order1:= xlAscending, _             'nicht erlaubt!
  Header:=xlGuess                     'dieser Kommentar ist o.k.
```

Änderungen rückgängig machen

Wenn Sie versehentlich einen markierten Bereich löschen oder eine Änderung am Programmcode rückgängig machen möchten, können Sie den bisherigen Zustand des Programms mit dem Kommando Bearbeiten|Rückgängig bzw. mit Alt+← wiederherstellen. Mit Bearbeiten|Wiederherstellen bzw. mit Strg+Z können Sie auch das Rückgängig-Kommando wieder rückgängig machen. Diese Undo- und Redo-Funktion arbeitet mehrstufig, d. h., Sie können mehrere Änderungen zurücknehmen.

Automatisch speichern

Excel läuft zwar in aller Regel stabil, aber ein Absturz ist nie ganz ausgeschlossen. Aus diesem Grund sollten Sie Ihre Arbeitsmappe möglichst oft speichern! Bei Excel 2000 können Sie die Add-In-Erweiterung Automatisch speichern aktivieren, die Ihre Arbeitsmappe in regelmäßigen Abständen speichert. Dazu führen Sie in Excel (nicht in der Entwicklungsumgebung) Extras|Add-In-Manager aus und aktivieren die Option Automatisches Speichern. Es erscheint nun alle zehn Minuten eine Rückfrage, ob Sie noch ungesicherte Dateien speichern möchten. Ab Excel 2002 ist eine vergleichbare Funktion direkt in Excel integriert (also nicht mehr als Add-In). Das Speicherintervall kann nun unter Datei|Optionen|Speichern eingestellt werden.

3.4 Makros ausführen

Es wurde oben bereits erwähnt, dass Sie das Makro, in dessen Codebereich sich der Cursor gerade befindet, sehr bequem durch F5 aufrufen können. Diese Art des Makrostarts ist allerdings nur in der Entwicklungsumgebung möglich, wo Sie ein Makro alternativ auch mit dem Menübefehl Extras|Makro starten können. In Excel selbst steht Ihnen für den gleichen Zweck das Kommando Entwicklertools|Makros zur Verfügung. Mit beiden Befehlen können Sie alle parameterlosen Makros starten, die in irgendeiner der zurzeit geladenen Dateien definiert sind.

Daneben bestehen auch elegantere Möglichkeiten zum Makrostart:

» Im Dialog Entwicklertools|Makros können Sie über den Button Optionen jedem Makro eine Tastaturabkürzung der Art Strg+Anfangsbuchstabe zuweisen. Aus unerfindlichen Gründen steht der Optionen-Button nur im Makro-Dialog von Excel, nicht aber im Makros-Dialog der Entwicklungsumgebung zur Verfügung.

» Makros können mit dafür vorgesehenen Ereignisroutinen verbunden werden. Sie werden dann beim Auftreten bestimmter Ereignisse (etwa dem Verstreichen einer vorgegebenen Zeit, dem Aktivieren eines Tabellenblatts etc.) automatisch aufgerufen. Ereignisse sind Thema von Abschnitt 4.4.

» Sie können ein neues Makrostartsymbol in die Symbolleiste für den Schnellzugriff einfügen. Dazu klicken Sie mit der rechten Maustaste auf die Symbolleiste und wählen den Befehl Symbolleiste für den Schnellzugriff anpassen. Stellen Sie das Listenfeld Befehle auswählen auf »Makros« ein, markieren Sie das gewünschte Makro in der Liste darunter, und klicken Sie auf Hinzufügen.

» Sie können eigene Menüs und Symbolleisten anlegen und Makrostartbefehle darin verankern. Allerdings werden diese Befehle nur noch in der Befehlsregisterkarte Add-Ins angezeigt, weil es in Excel 2007/2010 ja offiziell keine Menüs und Symbolleisten mehr gibt. Im Unterschied zu der Symbolleiste für den Schnellzugriff gelingt die Integration von Makrostartbefehlen in selbst erstellte Menüs und Symbolleisten aber nur noch mit den Mitteln der Programmierung. Welche das sind, ist Gegenstand des Abschnitts 8.1.

» Und schließlich können Sie in jede andere Registerkarte des Menübands ebenfalls Buttons zum Aufruf von Makros einfügen. Für diese weitestgehende Form der Integration in die Excel-Oberfläche ist unter anderem das manuelle Anlegen beziehungsweise Anpassen einer XML-Datei notwendig, die Sie anschließend der Excel-Datei (die jetzt ein »getarnter« ZIP-Container ist) hinzufügen müssen. Die Einzelheiten des Verfahrens, das sich komplizierter anhört, als es ist, werden in Abschnitt 8.2 beschrieben.

Die gerade aufgezählten Methoden gelten für Befehlsmakros, die mit dem Schlüsselwort `Sub` eingeleitet werden. Funktionsmakros (Schlüsselwort `Function`) sind dagegen nicht zum direkten Aufruf geeignet. Sie können innerhalb anderer Makros sowie als Rechenfunktionen in der Formel einer Tabellenzelle eingesetzt werden.

Makros unterbrechen

Alle Makros können jederzeit mit Strg+Untbr gestoppt werden. Wenn Sie im daraufhin erscheinenden Dialog Makrofehler den Button Testen auswählen, können Sie den Code bearbeiten. Sie können sich dort einzelne Variablen ansehen und das Makro anschließend mit F5 fortsetzen.

Der Direktbereich (Testfenster)

Das Fenster des Direktbereichs stellt eine Hilfe zum Testen neuer Prozeduren und zur Fehlersuche dar. (In früheren Versionen wurde der Direktbereich als Testfenster bezeichnet.) Die Anweisung `Debug.Print` führt eine Ausgabe im Direktbereich durch. Der Direktbereich wird über Ansicht|Direktfenster oder mit Strg+G aktiviert.

Das Direktfenster enthält die letzten 200 mit `Debug.Print` durchgeführten Ausgaben. Im Direktbereich können Sie Anweisungen angeben, die mit Return sofort (direkt) ausgeführt werden. Der Direktbereich eignet sich besonders zum Austesten von Variablen oder Eigenschaften – etwa durch Anweisungen wie `?varname` oder `?Application.ActiveSheet.Name` (gibt den Namen des gerade aktiven Tabellenblatts aus). Das Fragezeichen gilt dabei als Abkürzung für die `Print`-Methode. Fließkommazahlen werden im Testfenster generell mit maximal acht Nachkommastellen angezeigt, auch wenn 16 Nachkommastellen existieren.

Im Direktbereich sind auch Wertzuweisungen an Variablen oder Eigenschaften sowie der Start von Makros durch die Angabe des Namens möglich. Sie dürfen im Direktbereich ohne vorherige Deklaration neue Variablen einführen (auch dann, wenn im Programmcode `Option Explicit` gilt).

KAPITEL

Das Überwachungsfenster ermöglicht die stetige Anzeige des Inhalts diverser Eigenschaften oder Variablen. Der Umgang mit diesem Fenster wird ausführlich in Kapitel 6 beschrieben, in dem es um die Fehlersuche (Debugging) geht. Dort wird auch das Fenster zur Anzeige aller gerade aktiven Prozeduren beschrieben.

3.5 Makroaufzeichnung

Prinzipiell bestehen zwei Möglichkeiten zur Erstellung eines Makros: Entweder Sie geben das Makro über die Tastatur ein, oder Sie verwenden das Kommando Entwicklertools|Makro aufzchn. (warum dieser Befehl in Excel 2010 derart abgekürzt ist, in Excel 2007 dagegen nicht, ist kaum zu verstehen). In der Realität kommt die Mischform dieser beiden Varianten am häufigsten vor: Sie verwenden zuerst die Makroaufzeichnung, um das Grundgerüst des Makros zu erstellen, und verändern anschließend via Tastatur die Details des Makros nach Ihren Vorstellungen.

Der große Vorteil der Makroaufzeichnung besteht darin, dass Sie sich die endlose Suche nach den gerade erforderlichen Schlüsselwörtern ersparen. Selbst wenn das aufgezeichnete Makro nur in Grundzügen dem entspricht, was Sie eigentlich erreichen möchten, sind doch zumindest die angegebenen Objekte, Eigenschaften und Methoden brauchbar.

Nachteile der Makroaufzeichnung bestehen unter anderem darin, dass Excel oft einen unnötig umständlichen Code aufzeichnet; bei der Aufzeichnung von Dialogeingaben werden etwa *alle* Einstellmöglichkeiten in den Code aufgenommen (auch wenn nur eine einzige Einstellung verändert wurde).

Aufzeichnung starten und beenden

Die Makroaufzeichnung beginnt normalerweise in einem Tabellenblatt mit dem oben erwähnten Kommando. Anschließend müssen Sie den Namen des aufzuzeichnenden

Makros und die gewünschte Arbeitsmappe angeben, in der das Makro aufgezeichnet werden soll (üblicherweise in »Dieser Arbeitsmappe«).

Die Makroaufzeichnung wird durch ENTWICKLERTOOLS|AUFZEICHNUNG BEENDEN oder durch das Anklicken des entsprechenden Symbols (ein kleines schwarzes Quadrat in der Statusleiste von Excel) beendet.

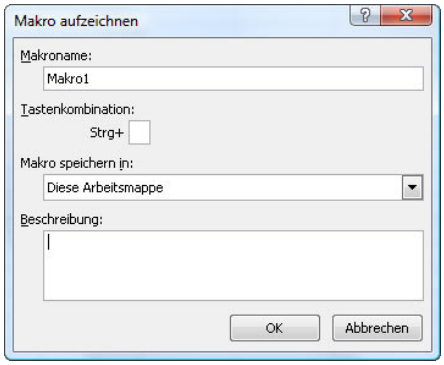


Abbildung 3.3: Start der Makroaufzeichnung

Excel erzeugt bei der Makroaufzeichnung meistens ein neues Modul. Sie können den Code des Makros aber ohne Weiteres nach der Aufzeichnung über die Zwischenablage in ein anderes Modul kopieren und das nun leere Modul wieder löschen.

Die persönliche Makroarbeitsmappe

Makros, die speziell zur gerade aktiven Excel-Datei gehören, sollten immer in »Diese Arbeitsmappe« aufgezeichnet werden. Makros, die permanent zur Verfügung stehen sollen (unabhängig davon, welche Excel-Datei gerade geladen ist), sollten dagegen in »Persönliche Makroarbeitsmappe« gespeichert werden. Diese Arbeitsmappe wird unter dem Namen *Personal.xlsb* im Benutzerverzeichnis *C:\Users\Benutzername\AppData\Roaming\Microsoft\Excel\Xlstart* gespeichert und bei jedem Start von Excel automatisch geladen. Damit stehen die darin gespeicherten Makros jederzeit zur Verfügung.

Häufig ist das Fenster der persönlichen Makroarbeitsmappe in Excel ausgeblendet, damit die Arbeitsmappe keinen Platz am Bildschirm wegnimmt. Gleichzeitig stellt dieses Verfahren einen Schutz gegen ungewollte Veränderungen dar. Die Arbeitsmappe kann über ANSICHT|EINBLENDEN sichtbar gemacht werden.

Absolute und relative Makroaufzeichnung

Zellbezüge können während der Makroaufzeichnung wahlweise relativ zur Startposition oder mit absoluten Adressen aufgezeichnet werden. Welche der beiden Varianten günstiger ist, hängt von der Anwendung Ihres Makros ab. Zur Umschaltung zwischen den beiden Modi müssen Sie den Befehl Relative Aufzeichnung in der Registerkarte Entwicklertools verwenden.

Es ist erlaubt, diese Einstellung während der Makroaufzeichnung beliebig oft zu verändern. Die falsche Einstellung dieses Menüeintrags ist aber eine häufige Ursache dafür, dass zuvor aufgezeichnete Makros nicht richtig funktionieren!

3.6 Tastenkürzel

Der Abschnitt gibt einen Überblick über die wichtigsten Tastenkürzel, die während der Programmentwicklung benötigt werden. Nicht mit aufgenommen wurden Tastenkürzel, die generell unter Windows gelten (etwa Strg + C zum Kopieren in die Zwischenablage). Die Tastenkürzel wurden nach Bereichen geordnet, in denen sie am häufigsten benötigt werden.

Wechsel des aktuellen Fensters	
Alt + F11	wechselt zwischen Excel und der Entwicklungsumgebung
Strg + ⇆	wechselt zwischen allen Visual Basic-Fenstern
Alt + F6	wechselt zwischen den beiden zuletzt aktiven Fenstern
Strg + G	ins Direktfenster (Debug-Fenster) wechseln
Strg + R	ins Projektfenster wechseln
F2	in den Objektkatalog wechseln
F4	ins Eigenschaftsfenster wechseln
F7	ins Codefenster wechseln

Eigenschaftsfenster	
⇧ + ⇆	springt ins Objektlistenfeld
Strg + ⇧ + X	springt zur Eigenschaft mit dem Anfangsbuchstaben X

Programmausführung	
F5	Programm starten
Strg + Untbr	Programm unterbrechen
F8	ein einzelnes Kommando ausführen (Single Step)
⇧ + F8	Kommando/Prozeduraufruf ausführen (Procedure Step)
Strg + F8	Prozedur bis zur Cursorposition ausführen
Strg + ⇧ + F8	aktuelle Prozedur bis zum Ende ausführen
F9	Haltepunkt setzen
Strg + F9	Ort des nächsten Kommandos bestimmen

Codefenster	
⇆	markierten Zeilenblock einrücken
⇧ + ⇆	markierten Zeilenblock ausrücken
Strg + Y	Zeile löschen
Alt + ←	Änderung widerrufen (Undo)
Strg + Z	Widerruf rückgängig machen (Redo)

Codefenster	
Strg + ↑ / ↓	Cursor zur vorigen/nächsten Prozedur
⇧ + F2	zur Prozedurdefinition bzw. zur Variablendeklaration
Strg + ⇧ + F2	zurück zur letzten Cursorposition (Undo zu ⇧ + F2)
F6	Codeausschnitt wechseln (bei zweigeteiltem Fenster)
Strg + F	Suchen
F3	Weitersuchen
Strg + H	Suchen und Ersetzen
Strg + SPACE	Schlüsselwort/Variablennamen vervollständigen
⇆	Auswahl im IntelliSense-Listenfeld durchführen
Esc	IntelliSense-Listenfeld verlassen