Ulf Neubert

dBASE lebt!

Band 2

Grundlagen

dBASE Programmierung unter Windows

Für dBASE Plus und seine Vorgänger
dBASE SE, dBASE 2000, Visual dBASE

Inhaltsverzeichnis

1. Einführung

Aus der DOS-Ära stammend und danach lange für tot gehalten, hat **dBASE** in den 90er Jahren als ***Visual dBASE*** und ***dBASE for Windows*** sein verdientes Comeback erlebt. Anfangs noch als 16bit Version, später im 32bit-Gewand. Um den Jahrtausendwechsel wurde es als ***dBASE 2000*** angeboten und gerade als dieses Buch entsteht, heisst es zur Abwechslung ***dBASE Plus***.

Fast ebenso oft wie der Name hat in den letzten Jahren auch der Eigentümer gewechselt. Diese vielen Wechsel von Name und Inhaber, die oft nicht klar erkennbare Strategie und ein zuweilen eher bescheidenes Marketing haben vermutlich ihren Teil dazu beigetragen, dass leider viele Anwender mangels Vertrauen in das Produkt und die Zukunft dBASE den Rücken gekehrt haben.

Schlimmer noch, oft habe ich es erlebt, dass auch langjährige Programmierer von dBASE unter DOS die neuen Windows-Versionen nicht kannten, ja nicht mal wussten dass es sie gibt. Ich musste bei vielen Überzeugungsarbeit leisten und mehrfach betonen „*ja, dBASE gibt es auch unter Windows! dBASE lebt!*“, und so entstand dann auch der Titel für diese neue Buchreihe. Vielleicht trägt er ja dazu bei, etwas vom Glanz alter Zeiten für dBASE zurück zu bekommen.

Nach so häufig geänderten Namen in so kurzer Zeit möchte ich die ständige Änderung des Textes bei jeder Neuauflage des Buchs vermeiden und bin daher so frei, im folgenden Text einfach den etwas allgemeineren Begriff

dBWin

zu verwenden, als Synonym für die gerade aktuelle Version von dBASE unter Windows. Bei Erstellung dieses Textes war gerade ***dBASE Plus V 2.5*** aktuell, und kaum als die letzten Zeilen geschrieben waren erschien das Update ***V 2.6***.

Aber mit ***dBWin*** wird für die weiteren Ausgaben dieser Buchreihe ein fester Begriff beibehalten, egal wie das Produkt in ein paar Jahren heissen wird.

Wenn im Text bezug auf ältere Versionen genommen wird, gleich ob für Windows oder DOS, so wird dies immer ausdrücklich erwähnt. Ansonsten können Sie davon ausgehen, dass die aktuelle Version gemeint ist, bzw. die Version, die bei der Texterstellung gerade aktuell war, denn Ihre Version könnte schon wieder etwas neuer sein wenn, Sie dieses Buch lesen.

1.1 Allgemeines zu dieser Buchreihe

Ich werde mich mit dieser Reihe gezielt und in jeweils kleinen und leicht verdaulichen Häppchen pro Band einzelnen Schwerpunkten widmen.

Sie bekommen kein kiloschweres, waffenscheinpflichtiges Buch mit über 1.000 Seiten vorgesetzt, sondern mehrere kleinere überschaubare Ausgaben, die sich intensiv einem ganz bestimmten Thema oder einem Themenbereich widmen. Jedes der Bücher wird in sich abgeschlossen sein. Die ersten beiden Bände, *EINFÜHRUNG* und *GRUNDLAGEN*, werden aber eine gemeinsame Basis bilden, die in den folgenden Ausgaben stets als bekannt vorausgesetzt wird.

Geplante Themen sind z. B. Klassen und Objekte, Zugriff auf fremde Dateien und Datenbanken, SQL, die Feinheiten und Fallen von Windows, DDE, OLE, Web-Datenbanken, Programmierkniffe für Profis und vieles mehr. Einer der geplanten Bände wird sich intensiv mit Windows-Programmierung befassen, also dort beginnen wo andere Bücher zu dBASE bislang immer aufhörten.

Jeder Band hat seine Zielgruppe: Einsteiger, Fortgeschrittene oder Profis. Die Reihe startet mit zwei Bänden für Einsteiger und DOS-Umsteiger. Es soll aber nicht die ultimative Lösung für ein Problem angeboten werden, sondern die Bücher sind als Hilfe zur Selbsthilfe gedacht, sollen mögliche Wege aufzeigen und eine wertvolle Unterstützung für Ihre eigene Arbeit mit dBWin sein.

Diese Bücher werden auch nicht alles rund um dBASE abdecken können, sonst wären wir wieder beim 1.000-Seiten Wälzer. Sie werden also weder ein „Kompendium von A-Z“ darstellen, noch werden sie eine vollständige „Befehls-Referenz“ enthalten (die es aber bei Erfolg der Reihe evtl. später einmal separat geben könnte). Es werden Themen beleuchtet, von denen ich glaube, dass sie wichtig sind. Und das dann jeweils sehr intensiv.

Hier, liebe Leserin und lieber Leser, kommen Sie ins Spiel! Welche Themen und Schwerpunkte es künftig noch geben wird hängt nicht zuletzt auch von Ihnen, Ihren Anforderungen und Wünschen ab. Wenden Sie sich bitte direkt an mich und teilen Sie mir Ihre Wünsche mit. Je mehr Anfragen zu einem Thema eintreffen, desto grösser ist die Chance dass es später umgesetzt wird. Wünsche, Anregungen und Kritik richten Sie bitte per eMail an

dbase@ulfneubert.de

Bitte haben Sie Verständnis, dass Ihre eMails zu künftigen Themen nicht in jedem Fall individuell beantwortet werden können. Das ist zeitlich einfach nicht möglich. Aufgenommen werden Ihre Vorschläge aber auf jeden Fall.

1.2 Infos zum Autor

Ich arbeite seit vielen Jahren selbständig als Programmierer und bin Inhaber einer Software-Firma. Insbesondere mit dBWin habe ich in den letzten Jahren einige Projekte realisiert. Neben der Umstellung vieler alter DOS-Programme entwickelte ich auch ein sehr umfangreiches Programm zur Depotverwaltung für Vermögensberater und Investmentfonds-Vermittler. Es wurde konsequent von Anfang an mit dBWin entwickelt und hat noch heute treue Anwender. Es heisst *Avalonia*, ich habe es längst verkauft, betreue es aber immer noch mit.

Mit der Zeit hat sich sehr viel Erfahrung mit dBWin angesammelt. Natürlich habe ich Fehler gemacht und so manches Problem bereitete Kopfschmerzen, bevor es dann meistens doch auf die eine oder andere Art gelöst wurde.

Der Gedanke, diese Erfahrungen in Seminaren und Büchern auch anderen dBASE-Nutzern zugänglich zu machen, bestand schon länger. Letztendlich haben dann immer häufigere Nachfragen einiger Kunden nach solcher Unterstützung dazu geführt, die Idee in Form dieser Buchreihe umzusetzen.

Sie können mich und meine Firma gern im Internet besuchen:

www.cantaria.de
www.ulfneubert.de

Dort können Sie die aktuell verfügbaren und noch geplanten Ausgaben dieser Buchreihe ersehen und natürlich auch bestellen. Es ist auch geplant, die in den Büchern verwendeten Quellcodes dort zum Download anzubieten, allerdings nur bei den Ausgaben, in denen längere Programmcodes verwendet werden. Desweiteren werden Sie dort Leseproben aller Bücher dieser Reihe finden.

Die **Cantaria GmbH** führt individuelle Aufträge für Programmierung aus, ganz besonders für dBWin, aber natürlich auch anderes. Wir übernehmen komplette Projekte jeden Umfangs und ausdrücklich auch kleine Projektteile und einzelne Module, damit Sie entlastet werden. Gern unterstützen wir Sie per eMail und stehen Ihnen bei Fragen rund um dBWin helfend zur Seite. Auch eine individuelle dBWin-Schulung, bei Ihnen oder uns, ist möglich.

Allgemeine Anfragen zu Programmierleistungen, Übernahme von Projekten (oder Teilen davon), Tests und Optimierung Ihrer bereits erstellten eigenen Programme, Umsetzung von alten DOS-Programmen auf Windows sowie Anfragen zu individuellen Schulungen etc. richten Sie bitte per eMail an

info@cantaria.de

1.3 Syntax und Layout

Damit Sie stets wissen von was die Rede ist werden in diesem Buch einige gleiche Formate und Textdarstellungen für bestimmte Dinge verwendet.

Tasten und Buttons
Zu drückende Tasten wie **[Return]** oder **[Esc]** werden in eckige Klammern gesetzt und fett geschrieben. Mehrere gleichzeitig zu drückende Tasten werden mit **+** kombiniert, z. B. **[Alt] + [F4]** zum Beenden des Programms. Ebenso verhält es sich mit Buttons, z. B. wenn Sie auf **[Ok]** klicken sollen.

Menüpunkte, Dialogelemente, Dateinamen, Datenbankfelder etc.
Werden *kursiv* geschrieben, z. B. das Eingabefeld *Nachname* dient dazu ..., laden Sie die Datei *anfang.prg* in den Editor ..., das Feld *nummer* enthält ...

dBWin-Befehle und Programmbeispiele
Quellcodes werden mit anderer Schrift und etwas eingerückt geschrieben:

```
use adressen.dbf
? reccount()
```

Manchmal werden Befehlszeilen nummeriert. Die Nummern dienen aber nur für nachfolgende Erklärungen im Text und sind nicht von Ihnen abzutippen.

Platzhalter für Parameter oder Variablen etc. stehen in spitzen Klammern:

```
seek <suchbegriff>
select <alias>
```

Optionale Befehlselemente werden in eckigen Klammern gesetzt:

```
set order to [indexname]
set filter to [bedingung]
```

Wahlweise Optionen werden mit einem | (Pipe) getrennt:

```
set exact on | off
```


Wichtige Punkte, Tips und Tricks, Hinweise auf Besonderheiten.


Vorsicht Falle, hier wird vor Bugs und häufigen Fehlern gewarnt.

1.6 Schwerpunkt dieser Ausgabe

Diese zweite Ausgabe ist der ideale Anschluss an den ersten Band der Reihe. Einige der dort bereits angeschnittenen Punkte werden hier vertieft, und auch viele neue Elemente von dBWin werden erstmals genauer betrachtet.

Auch diese Ausgabe zielt wieder im Schwerpunkt auf Ein- und Umsteiger. Also auf dBWin-Neulinge oder auf Umsteiger von dBASE unter DOS, sowie Umsteiger von den ersten 16bit-Versionen von dBASE unter Windows.

Aber auch wenn Sie schon länger mit dBWin arbeiten sollten Sie einen Blick in dieses Buch werfen. Besonders die Informationen über den Umgang mit Prozeduren, Variablen und Datentypen, sowie die komplette Erläuterung der globalen Systemvariablen und die Analyse wichtiger Objekte wie *_app* halten auch für dBWin-Kenner bestimmt das eine oder andere Aha-Erlebnis bereit.

Neben weiteren Details zu Befehlsfenster und Regiezentrum gibt es Infos zu Datenbank- und Formulardesigner, Objekt-Inspektor, Quellcode-Editor und anderen Helferlein. Danach finden Sie zahlreiche wertvolle Tips zur sicheren Programmierung mit dBWin, die Ihnen ein fundiertes KnowHow vermitteln. Auch die wesentlichen Formular-Komponenten werden eingehend erläutert.

ⓘ Dieser zweite Band schliesst nahtlos an Band 1 *EINFÜHRUNG* an. Wenn Sie den ersten Band nicht kennen ist das zwar auch ok, dennoch empfiehlt es sich, die beiden ersten Bücher dieser dBASE-Reihe als Einheit zu betrachten. Überschneidungen sind minimal und auf das absolut notwendige beschränkt, da ich auch die Leser berücksichtigen muss, die Band 1 (noch) nicht kennen.

Die weiteren geplanten Bände werden das in Band 1 & 2 vermittelte Wissen als Grundlagen voraussetzen und sich dann weiteren Schwerpunkten widmen.

ⓘ Bei solch einem Buch ist es immer eine Gratwanderung, weder einen Teil der LeserInnen zu überfordern, noch den anderen Teil zu langweilen. Vermutlich wird mir dieser Spagat nicht immer gelingen, das ist auch schlicht unmöglich. Im Zweifelsfall werde ich aber einer genaueren Beschreibung den Vorzug geben, auch auf die Gefahr hin, einen Profi damit für einen oder zwei Absätze zu langweilen. Aber in diesem Fall haben Sie als Fortgeschrittener immerhin den grossen Vorteil, diesen Absatz einfach elegant überspringen zu können, während ein Einsteiger sich das Wissen über ein nur unzureichend beschriebenes Detail keineswegs so einfach aus der Luft saugen kann.

So, und nun sitzen Sie bitte bequem, wir fangen an ...

2. Wichtige Elemente von dBWin

In diesem Kapitel erfahren Sie wichtige Dinge über grundlegende Elemente von dBWin. Es geht um das Befehlsfenster, das Regiezentrum, um Projekte und deren Einsatz, die Erstellung von EXE-Programmen und vieles mehr.

Nicht weiter erklärt werden Punkte, die Dinge, die im Grunde selbsterklärend sind. Was z. B. die Menüs *Datei - Neu* oder *Bearbeiten - Einfügen* machen ist so allgemein, dass ich mir erlaube, es als bekannt vorauszusetzen. Das gehört zum Thema Basiswissen für Windows und ist in vielen Programmen gleich. Ich traue auch jedem Leser zu, das ggf. einfach mal selbst auszuprobieren ...

Auch wenn es hier um Grundlagen geht, möchte ich nicht die vielen Dinge wiederholen, die Sie so oder so ähnlich von dutzenden Programmen und vermutlich auch aus anderen Büchern bereits kennen. Das langweilt Sie nur. Das Buch befasst sich mit dem, was speziell bei dBWin neu und wichtig ist.

Die Abschnitte dieses Kapitels sind meist relativ unabhängig voneinander. Damit können Sie gezielt Ihr Wissen zu einem speziellen Thema vertiefen, sich aber auch der Reihe nach mit den einzelnen Bereichen befassen, um so einen systematischen Überblick über das Ganze zu erhalten.

Wenn Sie etwas davon gut kennen überspringen Sie einfach die betreffenden Seiten. Wenn Sie etwas nicht oder noch nicht gut genug kennen lesen Sie es. Und zum lesen und lernen gehört auch immer ausprobieren. Spielen Sie, setzen Sie die Beispiele um, variieren Sie sie und passen Sie die gezeigten Wege an Ihre eigenen Wünsche und Anforderungen an.

ⓘ Es wird wohl das einzige mal innerhalb dieser Buchreihe bleiben, dass diese Basis-Elemente von dBWin so sehr im Detail erläutert werden. In den weiteren Bänden wird das Wissen um diese Dinge dann vorausgesetzt.

ⓘ Beachten Sie bitte auch den ersten Band dieser Reihe mit dem Titel *EINFÜHRUNG*, der nicht nur viele Elemente grundsätzlich erläutert, sondern auch noch eine Reihe von Anwendungsbeispielen gibt. Kurze inhaltliche Überschneidungen zwischen diesem Kapitel und Band 1 liessen sich dabei nicht immer vermeiden, wurden aber möglichst gering gehalten.

2.1 Befehlsfenster

Beginnen wir mit dem wohl wichtigsten Fenster, dem *Befehlsfenster*. Darin können Sie fast alles machen, was rund um die Programmierung mit dBWin nötig ist. Sie können Dateien zum bearbeiten aufrufen, kopieren, löschen, Datenbanken öffnen und bearbeiten, Projekte öffnen, die meisten dBASE-Befehle ungefährlich testen und natürlich auch ganze Programme starten.

Das Befehlsfenster, hier sind Sie am Drücker der Macht

So unscheinbar und leer es am Anfang auch aussieht, dieses Fenster bietet Ihnen die volle Kontrolle über dBWin. Lassen Sie sich also vom harmlosen Aussehen des Fensters nicht täuschen, das ist Understatement in Reinkultur.

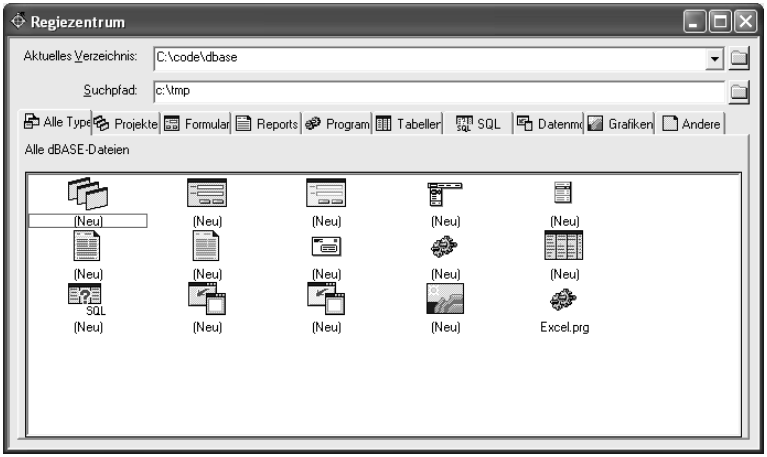
Das Befehlsfenster ist zweigeteilt. Der obere Bereich dient der Eingabe von Befehlen, der untere Bereich zeigt deren Ergebnisse oder bei Bedarf auch sonstige Meldungen von dBWin an. Die Trennlinie dieser beiden Bereiche in der Mitte des Fensters können Sie mit der Maus nach oben oder unten verschieben und so ganz nach Bedarf entweder den einen oder den anderen Bereich vergrössern. Die relativen Grössen der beiden Bereiche bleiben in etwa erhalten, wenn Sie das gesamte Fenster vergrössern oder verkleinern.

ⓘ Sie sehen kein Befehlsfenster? Dann bitte im Menü *Ansicht* öffnen. Wenn es das Menü nicht gibt bitte vorher den Projekt-Explorer schliessen, oder in ein anderes Unterfenster, z. B. in das Regiezentrum, wechseln.

In meiner Praxis hat es sich als ideal herausgestellt, wenn Sie die Mittellinie so verschieben, dass der obere *Eingabebereich* ca. ein Drittel und der untere *Ausgabebereich* ca. zwei Drittel des Fensters einnimmt. Aber das ist natürlich Ihre Entscheidung und Sie können das halten wie Sie wollen.

Sie können das Befehlsfenster auch jederzeit schliessen oder zum Symbol verkleinern, wenn Sie es gerade mal nicht brauchen. Über das Menü *Ansicht - Befehlsfenster* können Sie es wieder öffnen, wenn es nicht angezeigt wird.

2.2 Regiezentrum



Das Regiezentrum in der Einstellung mit großen Symbolen

Das *Regiezentrum* bietet eine komfortable Verwaltung aller für Ihre dBWin-Programme verwendeten Dateien. Seien es Quellcodes, Formulare, Menüs, Projektdateien, Datenbanken, SQL-Abfragen oder Berichte. All das können Sie hier sehr einfach und bequem organisieren, ablegen, wiederfinden und zum bearbeiten oder ausführen aufrufen, und natürlich auch wieder löschen.

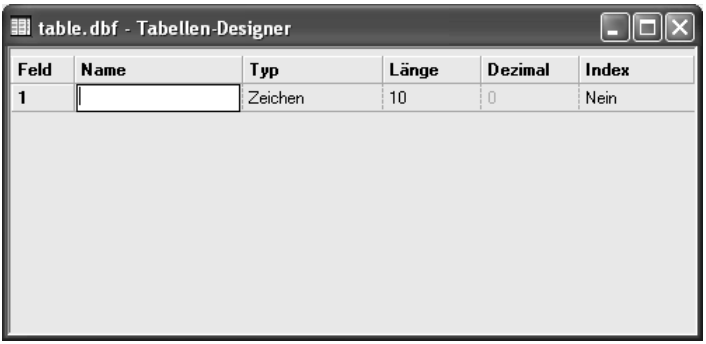
ⓘ Sie sehen kein Regiezentrum? Dann bitte im Menü *Ansicht* öffnen.

Sie *können* das Regiezentrum nutzen, *müssen* aber nicht. Es ist oft hilfreich, insbesondere für Einsteiger, aber es ist nicht zwingend nötig. Ich selbst habe es in all den Jahren mit dBWin so gut wie nie benutzt, nicht einmal bei Projekten die aus weit über 100 Quelldateien bestanden. Man kommt also auch sehr gut völlig ohne dieses Teil aus, das ist letztendlich eher eine Frage des Arbeitsstils und der persönlichen Gewohnheiten und Vorlieben.

Um das Regiezentrum für professionelle Arbeiten tatsächlich regelmässig zu benutzen fehlen einfach zu viele Funktionen. Der Name Regiezentrum ist daher doch ein wenig hoch gegriffen, mehr als eine rudimentäre Hilfe zur Dateiverwaltung ist es im Grunde nicht. Aber das ist Ihre Entscheidung. Wenn es Ihnen zusagt nutzen Sie es, wenn nicht lassen Sie es eben bleiben. Und dass Sie überhaupt diese Wahl haben ist ja auch ausgesprochen positiv.

2.4 Tabellen-Designer

Der Designer für Tabellen ist sehr wichtig, da Sie hier alle Eigenschaften der Tabelle, die Feldnamen sowie deren Datentypen und Längen, festlegen.



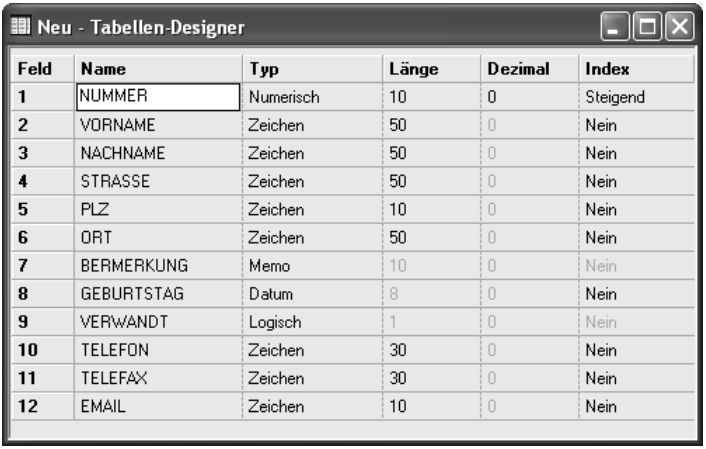
Feld	Name	Typ	Länge	Dezimal	Index
1		Zeichen	10	0	Nein

Der Tabellen-Designer mit einer neuen Tabelle

Eine vollständige Aufstellung der Feldarten, Datentypen, Längen und deren Besonderheiten finden Sie im Kapitel *Datentypen in Tabellen* auf Seite 292.

ⓘ Ich versuche, Brüche im Text und viele Verweise auf andere Kapitel zu vermeiden, aber hier ist es sinnvoll. Die Erklärung der Feldarten in Tabellen werden Sie sicher häufiger nachschlagen, auch unabhängig vom Designer. Daher macht es in diesem Fall Sinn, sie in einem eigenen Kapitel zu erläutern.

Eine einfache Adressdatei könnten beispielsweise so aussehen:



Feld	Name	Typ	Länge	Dezimal	Index
1	NUMMER	Numerisch	10	0	Steigend
2	VORNAME	Zeichen	50	0	Nein
3	NACHNAME	Zeichen	50	0	Nein
4	STRASSE	Zeichen	50	0	Nein
5	PLZ	Zeichen	10	0	Nein
6	ORT	Zeichen	50	0	Nein
7	BEMERKUNG	Memo	10	0	Nein
8	GEBURTSTAG	Datum	8	0	Nein
9	VERWANDT	Logisch	1	0	Nein
10	TELEFON	Zeichen	30	0	Nein
11	TELEFAX	Zeichen	30	0	Nein
12	EMAIL	Zeichen	10	0	Nein

Eine einfache Tabelle für Adressen

2.7 Projekt-Explorer und Projekte

Im Regelfall ist es sinnvoll, ein neues Projekt in einem eigenen Verzeichnis auf der Festplatte zu beginnen. In diesem Verzeichnis befinden sich dann alle wesentlichen Dateien des Projekts. Ich vermute mal, dass Sie Ihre Festplatte bereits so eingerichtet haben, dass es irgendwo ein oder mehrere Bereiche für Ihre Quellcodes gibt. Vielleicht gibt es dort auch bereits ein Unterverzeichnis für dBWin, und dort wiederum legen Sie das Verzeichnis für das Projekt an.

Leider geht das nicht gleich über das Regiezentrum von dBWin. Ich setze aber voraus, dass Sie wissen wie man ein Verzeichnis anlegt, zum Beispiel mit dem Windows-Explorer. Es geht aber auch im dBWin Befehlsfenster:

```
c:
cd\
md code
md code\dbase
md code\dbase\lernen                && nur ein Beispiel
cd code\dbase\lernen
```

Mit obiger Befehlsfolge aktivieren Sie das Laufwerk *C:*, wechseln in dessen Hauptebene, legen dort die Verzeichnis-Struktur *code\dbase\lernen* an und wechseln anschliessend in das neu erstellte Verzeichnis. Man fühlt sich doch gleich um Jahrzehnte jünger, fast wie damals beim guten alten MS-DOS ...

Für die jüngere Generation: mit *cd* wechseln Sie ein Verzeichnis, das steht für *change directory*. Mit *md* legen Sie ein neues Verzeichnis an, *make directory*. Diese alten Befehle funktionieren heute noch, sowohl in dBWin, als auch im für viele wichtigsten Teil von Windows, der *MS-DOS Eingabeaufforderung*. Es gab mal eine Zeit, da war das die einzige Art den PC zu bedienen, und ob Sie es glauben oder nicht, es hat funktioniert. Manche sagen besser als heute ...

 Wenn Sie das bei Ihnen auf einem anderen Laufwerk und/oder in einem anderen Unterverzeichnis haben wollen ist das kein Problem. Sie müssen dann eben obige Befehle entsprechend anpassen, sie waren nur als Beispiel gedacht.

 Aus Erfahrung rate ich, Verzeichnisse mit Umlauten oder Leerzeichen zu vermeiden. Namen mit mehr als 8 Zeichen sind ok, aber Geschmacksache. Das gilt nicht nur für Verzeichnisse, sondern auch für Dateien. Auch wenn ein Monster *c:\meine quellcodes\dbase programme\üben\lernprogramm1.prg* möglich ist, heisst das noch lange nicht, dass sowas auch sinnvoll ist.

2.8 Debugger

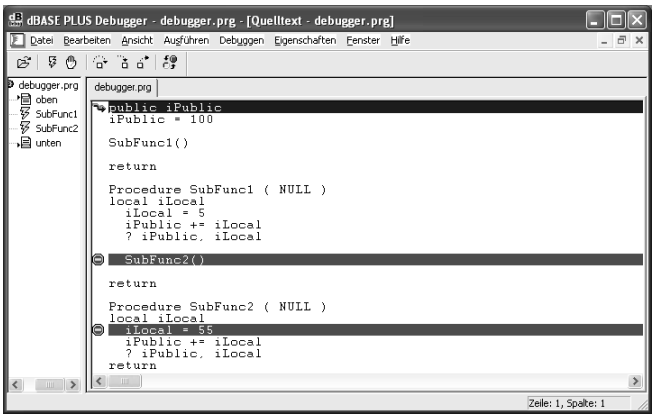
Mit dem Debugger können Sie Ihr Programm schrittweise verfolgen und jeden Befehl einzeln ausführen lassen. Dabei können Inhalte von Variablen geprüft und Bedingungen für Haltepunkte gesetzt werden. Zwar funktioniert das nur bei Programmen die innerhalb von dBWin ablaufen und nicht bei bereits fertig compilierten EXE-Programmen, aber dennoch ist der Debugger recht nützlich.

Aufgerufen wird er z. B. über das Hauptmenü *Erstellen - Debugger*, wenn Sie den Quelltext des Programm im Editor geladen haben. Oder im *Regiezentrum* klicken Sie die *.prg*-Datei mit der rechten Maustaste an und aktivieren im Mausmenü die Auswahl *Debugger*. Und natürlich gibt es auch einen Befehl:

```
debug <programmdatei>
```

Leider ist zumindest bei *dBASE Plus 2.5* der Debugger voller Tücken. Häufig hängt dBWin, Windows meldet *„Das Programm reagiert nicht mehr“*. Sollte dies bei Ihrer Version auch so sein hilft es manchmal, wenn Sie dBWin nach jedem Debug-Vorgang neu starten. Und der Debugger selbst sollte dann besser nicht mit dem Befehl *debug* aufgerufen werden, sondern nur über die Menüs, da er bei *debug* besonders häufig hängt, zumindest in meiner Version. Ab Version 2.6 (Herbst 2005) sollen die Probleme lt. Hersteller behoben sein.

Der Debugger hat ein Fenster, das Sie unabhängig vom dBWin-Fenster auf dem Bildschirm platzieren können. Sie sehen darin den Quellcode und die blaue Markierung zeigt an, wo sich das Programm befindet und welche Zeile des Codes als nächste ausgeführt wird. Die Steuerung der wichtigsten Punkte erfolgt über die Menüs *Ausführen* und *Debuggen*, bzw. über die Symbolleiste.



Ein Programm mit zwei Haltepunkten im Debugger

2.10 BDE-Administrator

Das ist ein eigenständiges Programm, das über ein extra Symbol in der Gruppe von dBWin gestartet wird. Hier legen Sie zahlreiche Grundeinstellungen für den Zugriff auf fremde Datenbanken, z. B. per ODBC-Treiber und SQL, fest.

Leider ist der BDE-Admin und seine Dialoge sehr technisch und wenn Sie nicht sowieso schon wissen um was es hier geht hilft Ihnen die dazugehörende Online-Dokumentation auch nicht immer weiter. Für dBWin selbst genügt es aber schon, sich nur mit ein paar der vielen Einstellungen zu beschäftigen.

ⓘ Beenden Sie bitte dBWin und alle damit erstellten Programme, bevor Sie den BDE-Admin aufrufen. Auch andere Programme, welche mit der BDE von Borland arbeiten (so Sie das wissen ...), sollten Sie sicherheitshalber beenden.

Sie starten den BDE-Administrator (Datei *bdeadmin.exe*) über dieses Symbol:

Sie bekommen eine kryptische Fehlermeldung beim Start des BDE-Administrators? Das passiert oft, wenn Sie verschiedene dBWin-Versionen parallel nutzen. Ein Neustart von Windows löst das Problem fast immer ...

Die Einstellungen der BDE werden teils in der Registry von Windows, teils in einer Datei namens *idapi.cfg* gespeichert. Im Normalfall wird diese Datei auch gleich beim Start des BDE-Admins geöffnet und zur Kontrolle inkl. Pfad in der Titelleiste des Programms angezeigt. Das ist sehr nützlich, denn so haben Sie immer eine Kontrolle, dass Sie auch die richtige *idapi.cfg* bearbeiten, falls Sie durch Installationen mehrerer dBWin-Versionen mehrere BDEs einsetzen. Auch das Menü *Objekt - Konfiguration öffnen* öffnet eine Konfigurationsdatei.

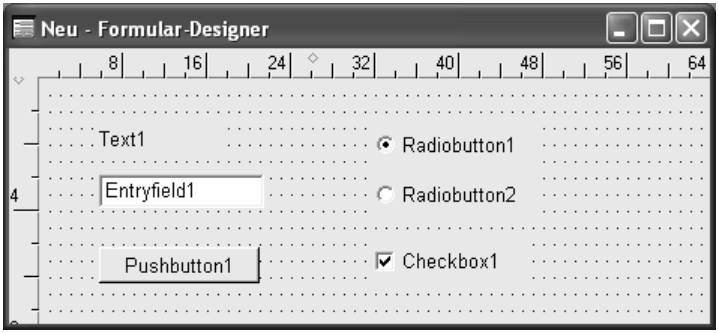
ⓘ **BDE** steht für **B**orland **D**atabase **E**ngine. Die Firma *Borland*, bekannt durch die Programmierwerkzeuge *Turbo-Pascal* und *Delphi*, war einer der Eigentümer von dBASE und für die Entwicklung der BDE verantwortlich. Diese Datenbank-Schnittstelle von Borland dient u. a. dazu, dass Sie mit dBASE auch auf andere Datenbanken zugreifen können. Fürs erste brauchen Sie sich darüber den Kopf nicht zerbrechen, es genügt wenn Sie die BDE zusammen mit dBWin installiert haben und minimal konfigurieren.

3. Formulare erstellen

Der *Formular-Designer* erstellt nicht nur Formulare, sondern fast alle Arten von Windows-Fenstern. Neben Formularen für die Ein- und Ausgabe Ihrer Daten können das Dialoge für Einstellungen im Programm, Meldungsfenster, Anzeigen für den Fortschritt laufender Aktionen und noch vieles mehr sein.

Dieser Band über die *GRUNDLAGEN* von dBWin gibt Ihnen grundlegende Infos zu den wichtigsten Punkten der Formulare und der Formular-Komponenten. Die reale Anwendung werden Sie schnell im Rahmen praktischer Beispiele erlernen, die zu den häufig benutzten Komponenten eigens gezeigt werden.

Band 1 enthielt ja bereits einfache Beispiele zum Entwurf simpler Formulare. Diese Einführung wird hier nun vertieft und die Komponenten, die Sie in den Formularen verwenden können, werden einzeln und ausführlicher dargestellt.

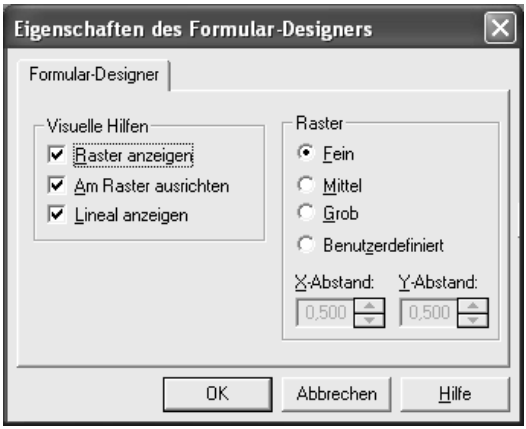


Formular-Designer mit Raster und Lineal

Mit dem Formular-Designer erstellen Sie das Basislayout für Ihre Fenster. Feintuning und das gewisse Etwas, das ein Profiprogramm von 0815-Ware unterscheidet, können Sie bis zu einem gewissen Punkt auch damit umsetzen. Ab und an wird es aber auch nötig sein, direkt am Quellcode des Formulars bzw. seiner Komponenten zu arbeiten. Das ist auch problemlos möglich, denn ein mit dem Designer erstelltes Formular wird immer als Quellcode in der entsprechenden *.wfm*-Datei abgelegt. Sie können diesen Quellcode mit dem dBWin Quelltext-Editor oder mit Ihrem eigenen Editor jederzeit ändern.

Sie öffnen den Formular-Designer z. B. über das *Regiezentrum*, dort auf der Seite *Formulare* auf das erste der diversen *Neu* klicken, oder die Tasten **[Umsch] + [F2]** betätigen. Oder über das Hauptmenü *Datei - Neu - Formular*.

Ob ein Raster im Formular angezeigt ist, und wenn ja in welcher Grösse, das können Sie in den *Eigenschaften des Formular-Designers* selbst bestimmen. Ebenso kann dort die Anzeige des Lineals ein- und ausgeschaltet werden. Sie erreichen die Eigenschaften durch einen Klick mit der rechten Maustaste ins Formular und dann die entsprechende Auswahl im Mausmenü, oder über das Hauptmenü *Eigenschaften* (aber nur wenn der Formular-Designer aktiv ist).



Eigenschaften des Formular-Designers

Die Option *Am Raster ausrichten* ist eine grosse Hilfe. Ist sie aktiv, werden alle Objekte an den Eckpunkten des Rasters platziert. Das erleichtert die einheitliche Positionierung der verschiedenen Bestandteile des Formulars.

ⓘ Rasterpunkte und Lineal berechnen sich in der Grundeinstellung aus der verwendeten Schrift für ein Formular. Die Grundschrift für Formulare und alle Objekte darin ist bei *dBASE Plus* „Arial“ in der Grösse 10 Punkte, während es bei *Visual dBASE* noch „MS Sans Serif“ mit 8 Punkten war. Sie können das in den Eigenschaften *ScaleFontName* und *ScaleFontSize*, beide zu finden in der Kategorie *Schrift* des Objektinspektors, bei fast allen Komponenten ändern.

ⓘ Sie können auch eine Skalierung in cm, Zoll oder Bildpunkten einstellen. Dazu ändern Sie die Eigenschaft *metric* im Formular. Die Vorgabe für *metric* ist 0, das bedeutet eine Skalierung entsprechend der eingestellten Grundschrift. Eine Änderung von *metric* auf eine fixe Skalierung wie z. B. *cm* klingt zwar logischer, dennoch rate ich dazu, die Vorgabe 0 zu lassen. Damit sind Sie flexibler, denn durch Änderung der Grundschrift passt sich autom. auch das Formular mit allen darin enthaltenen Objekten auf die neue Schriftgrösse an. Das erleichtert später mal die Einstellung einer variablen Grösse zur Laufzeit.

3.1 Der Objektinspektor

Dieser sehr wichtige Teil des Formular-Designers ist nicht nur für Formulare, sondern für alle Arten von Objekten ein überaus nützlicher Helfer. Sie können damit alle Eigenschaften des Formulars und seiner Bestandteile eintragen.

Da dieser Band bereits ein eigenes Kapitel über ihn enthält verzichte ich an dieser Stelle darauf, unnütz Seiten durch doppelte Beschreibungen zu füllen und verweise Sie stattdessen ausnahmsweise auf den anderen Teil des Buches.

Ist der Objektinspektor nicht sichtbar öffnen Sie ihn bitte durch Klick mit der rechten Maustaste ins Formular (im Formular-Designer), dann im Mausmenü auf *Objektinspektor*. Oder mit **[F11]**. Oder Menü *Ansicht - Werkzeugfenster*.

ⓘ Keine Angst, der Objektinspektor ist viel harmloser als er Ihnen anfangs vielleicht erscheint. Sie werden ihn beim Design von Formularen und auch an anderer Stelle sehr häufig benutzen und ihn so schnell und gut kennenlernen.

3.2 Die Feldpalette

Die Feldpalette ist mitunter nützlich, wenn Sie bereits eine Datenbank mit dem Formular verbunden haben. Dann enthält die Feldpalette eine Liste aller Felder der Tabelle, und Sie können durch Auswahl und anschliessenden Klick an die gewünschtem Stelle im Formular platzieren können. Dabei wird sowohl ein Objekt für das Tabellenfeld als auch ein Text für die Feldbeschriftung erzeugt.

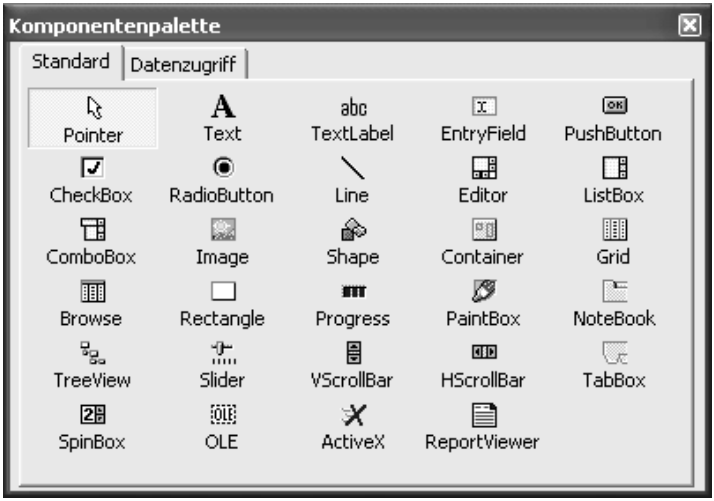
ⓘ Je nach Datentyp wird dabei entweder ein *entryfield* bei Textfeldern oder auch eine *spinbox* bei numerischen Feldern erzeugt. Aber dazu später mehr ...

Hier sehen Sie eine beispielhafte Feldpalette mit einigen Tabellenfeldern.

Beispiel einer Feldpalette

3.3 Die Komponenten eines Formulars

Ein sehr wichtiger Bestandteil des Formular-Designers ist das Fenster, in dem Sie die verschiedenen Komponenten eines Formulars auswählen. Das können Texte, Eingabefelder, Push- und Radiobuttons, Listen, Grafiken, Rechtecke, Linien und noch vieles andere mehr sein. Einige davon werden Sie häufig, manche ab und an und ein paar vielleicht nie benötigen. Das hängt ganz davon ab, welche Programme Sie entwickeln und wie diese aufgebaut sein sollen.



Die Komponenten-Palette mit den wählbaren Objekten

Bereits die Seite *Standard* enthält eine Vielzahl von Objekten für Sie bereit. Klicken Sie einfach den gewünschten Objekttyp in der Komponentenpalette an und wechseln Sie wieder ins Fenster mit dem in Arbeit befindlichen Formular. Der Mauscursor wird sich dort ändern (je nach Objekt anders) und Sie können das gewählte Objekt dann an der gewünschten Stelle im Formular platzieren.

3.3.1 Pointer, die Komponente die keine ist

Das ist als einzige Auswahl kein eigenständiges Objekt, sondern dient einfach nur dazu im Formular den Mauscursor als Zeiger (engl. Pointer) zu benutzen. Damit können Sie die bereits im Formular befindlichen Objekte auswählen, verschieben, in der Grösse ändern und ihre vielen Eigenschaften bearbeiten.

ⓘ Die Auswahl in der Komponentenpalette wechselt autom. auf *Pointer*, sobald Sie ein anderes Objekt ausgewählt und im Formular platziert haben. Das erleichtert es, das neu eingefügte Objekt danach auszuwählen, um es an die korrekte Position zu verschieben und seine Eigenschaften zu bearbeiten.

4. Basiswissen Programmierung

Hier erhalten Sie fundiertes Grundwissen zur Programmierung mit dBWin. Die Abschnitte dieses Kapitels sind im wesentlichen unabhängig voneinander, so dass Sie gezielt Ihr Wissen zu einem speziellen Thema vertiefen können. Ebenso können Sie sich der Reihe nach mit den Themen beschäftigen und so einen fundierten Überblick über das gesamte Spektrum erhalten.

Fast immer wird es darum gehen, Programmcode zu schreiben. Nur so erzielen Sie wirklich tiefe Einblicke und bekommen nachhaltiges KnowHow über die besprochenen Punkte und deren teils offenkundige, teils aber auch etwas versteckte und manchmal sogar verzwickte Zusammenhänge. Dieses Kapitel beginnt dort, wo viele ähnliche Bücher zu dBWin früher immer aufhörten, jenseits der Assistenten und Designer und mitten drin im Quellcode.

So manches wird für Sie evtl. längst bekannt sein, dann überspringen Sie den Abschnitt einfach. Aber mit der Gefahr, dass Sie damit vielleicht auch ein kleines Detail überspringen, das Ihnen bislang so garnicht bewusst war.

Es ist wichtig, hier auf diese Dinge einzugehen, denn nicht alle LeserInnen werden das jeweilige Wissen bereits locker aus dem Ärmel schütteln. Auch wenn es Ihnen lästig erscheinen mag: wenn eines der Themen für Sie neu ist oder Sie unsicher sind, dann tippen Sie die Beispiele bitte ab. Der Lerneffekt ist dadurch viel grösser, als wenn Sie sich nur die Seiten durchlesen und sich hier mal „aha“ und dort mal „soso“ denken, aber nichts selbst tun.

Handeln Sie! Setzen Sie die gezeigten Beispiele um, variieren Sie, probieren Sie eigene Ideen und andere Versionen der aufgezeigten Wege. Machen Sie dabei ruhig Fehler, aus nichts lernt man mehr und besser als aus den eigenen Fehlern. Die praktische Erfahrung am PC ist durch kein noch so intensives, aber am Ende doch rein passives Lesen dutzender Buchseiten zu ersetzen.

ⓘ Es wird wohl das einzige mal innerhalb dieser Buchreihe bleiben, dass diese allgemeinen Grundlagen erläutert werden. In den weiteren Bänden wird das Wissen um Prozeduren, Variablen, Makros usw. dann vorausgesetzt.

ⓘ Sie können fast alle dBASE-Befehle noch immer auf die ersten vier Zeichen kürzen und z. B. *modi stru* statt *modify structure* schreiben. Ich rate im Interesse gut lesbarer und leicht zu pflegender Programme aber davon ab.

4.1 Procedure und Function

Damit teilen Sie Ihr Programm in viele kleine und kleinste Häppchen auf. Es macht Ihr Werk übersichtlicher und erleichtert die Fehlersuche erheblich. Auch Änderungen und Erweiterungen, die Sie vielleicht erst nach Monaten oder Jahren einmal ausführen, gehen damit viel leichter von der Hand, weil Sie sich im „alten Code von damals“ viel schneller wieder zurecht finden.

Es gibt viele gute Gründe, Prozeduren und Funktionen zu verwenden. Der erste ist, dass sie helfen, Ihren Programmcode sinnvoll zu strukturieren.

Ein weiterer wichtiger Grund ist, dass Sie damit Ihr Programm in viele kleine Einzelbausteine zerlegen können. Viele Bausteine werden spezielle Aufgaben für das jeweilige Programm zu erledigen haben. Aber es wird auch eine ganze Reihe von Problemstellungen geben, die nicht auf ein Programm beschränkt sind, sondern die allgemein formuliert und völlig unabhängig vom Projekt gelöst werden können. Und wenn Sie diese Routinen in eigenen Dateien sammeln, haben Sie schon bald einen kleinen Schatz an Lösungen, den Sie auch für spätere Aufgaben immer wieder neu verwenden können.

Je kleiner die einzelnen Bausteine sind, in die Sie Ihr Programm zerlegen, desto höher ist die Chance, dass Sie die Teile später für ganz andere Projekte erneut verwenden können. Das ist ja gerade das schöne an unserem Beruf, dass wir eine einmal gefundene und funktionierende Lösung immer wieder neu verwenden können, ohne erneut viel Arbeit damit zu haben (und diese gesparte Arbeit oft trotzdem in Rechnung stellen können ...). Die schon vorhandene Lösung muss nur noch ins neue Programm eingebunden werden, fertig. Und je unabhängiger und flexibler die Lösung programmiert ist, desto einfacher lässt sie sich später auch in völlig andere Programme einbinden.

Und was ist nun der Unterschied zwischen *Function* und *Procedure*? Keiner! dBWin unterscheidet nicht (mehr) zwischen diesen beiden Varianten. Beide Schlüsselworte sind beliebig miteinander austauschbar, und es ist Ihrem persönlichen Geschmack überlassen, ob sie nur eines oder beide verwenden.

ⓘ Früher gab es den wesentlichen Unterschied, dass Funktionen immer einen Ergebniswert zurückgaben, Prozeduren dagegen nicht. Heute können beide Varianten ganz nach Bedarf ein Ergebnis zurückliefern oder auch nicht. In anderen Programmiersprachen ist diese Unterscheidung oft noch immer wichtig, so dass das hier gesagte sich nur auf dBWin bezieht.

Bevor wir ein praktisches Beispiel sehen noch ein kurzer Ausflug in die Theorie und die allgemeine Syntax von Prozeduren und Funktionen:

```
Procedure <name> [(parameterliste)]
Function <name> [(parameterliste)]
```

Wir haben also das Schlüsselwort *Procedure* oder *Function*, gefolgt von einem eindeutigen Namen, woran sich optional eine in runden Klammern gesetzte Liste von einem oder mehreren Parametern anschliesst. Mehrere Parameter werden dabei immer mit einem Komma getrennt geschrieben.

Unter Parametern versteht man beliebige Werte, also z. B. Zahlen oder Zeichenketten, die Sie beim Aufruf an die Prozedur mit übergeben, und die dann innerhalb der Prozedur verwendet werden, für was auch immer.

Sie kennen Parameter bereits von vielen zu dBASE gehörenden Funktionen, auch dort sind sie völlig normal. Hierzu ein simples Beispiel:

```
iZahl = val ( "123" )
```

Wir übergeben die Zeichenkette *„123“* als Parameter an die Funktion *val()*, die wiederum macht ihre Arbeit, wandelt die Zeichenkette in eine Zahl um und liefert das Ergebnis (als numerischen Datentyp) wieder zurück. In unserem Beispiel enthält die Variable *iZahl* anschliessend den numerischen Wert 123.

Sie können den Rückgabewert einer Funktion auch gleich weiterverarbeiten:

```
? 1000 + val ( "123" )
```

Und Sie können die Rückgabe auch einfach ignorieren:

```
val ( "123" )
```

was im gezeigten Fall zwar keinen Sinn macht, aber gehen tut es. Sie sehen, programmieren ist wie das richtige Leben, man kann viel sinnloses tun ...

Wo das dennoch einmal Sinn macht? Nun, wenn eine Funktion ein Ergebnis zurückliefert, das Sie in einigen Situationen auswerten wollen, in anderen Fällen dagegen ignorieren können, dann tun Sie genau das: Sie werten das Ergebnis aus wenn Sie es brauchen, und ignorieren es wenn es nicht nötig ist. Ignorieren heisst in diesen Fällen, dass Sie das gelieferte Ergebnis nicht mal mit einem schiefen Seitenblick betrachten, also auch nicht einer im Grunde völlig nutzlosen Dummy-Variablen zuweisen, sondern schlicht so tun, als würde die Funktion garnichts zurückliefern.

4.4 #define

Dieser Befehl kann gleich mehrere Funktionen erfüllen. Sie können damit einen Ersatzbegriff für oft benötigte Konstanten, egal von welchem Datentyp, festlegen. Mit ihm können Sie Einfluss auf die Compilierung Ihres Programms nehmen und ganz gezielt Teile in die Compilierung einbeziehen bzw. davon ausnehmen. Und zuletzt können Sie, als schon fortgeschrittene Programmiertechnik, Makros definieren, die Sie später wie „normale“ Befehle oder ähnlich wie Funktionsaufrufe in Ihrem Code verwenden.

Sie können beliebig viele *#define*-Befehle schreiben, entweder direkt in den *.prg*-Dateien, oder (was häufig gemacht wird) in eigenen *.h*-Dateien, den sog. *Header-Dateien*, die Sie dann wiederum per *#include* in die Programmdatei aufnehmen (siehe dazu bitte das vorherige Kapitel).

Meist ist es sinnvoller, *#define*-Befehle in eigene Headerdateien zu schreiben, nicht zuletzt auch aus dem Grund, dass Sie diese Header-Dateien dann in beliebig viele Programm-Dateien einbinden können. Ein Beispiel dazu folgt.

4.4.1 Konstanten mit #define

Damit definieren Sie oft benötigte Angaben als Konstanten. Sie weisen einem Begriff einen Platzhalter zu, und im Programm schreiben Sie nur noch diese Konstante bzw. diesen Platzhalter, statt des tatsächlichen Begriffs.

Der Name der Konstante bzw. des Platzhalters ist nahezu beliebig, solange er den üblichen Konventionen der Namensgebung in dBWin entspricht, er also eindeutig ist und noch nicht anderweitig für Variablen oder dBASE-Befehle verwendet wird. Der Name darf keine Leerzeichen enthalten, kann aber zur besseren Lesbarkeit mit einem Unterstrich _ in mehrere Bestandteile getrennt werden. Ein Bindestrich ist dagegen nicht zulässig. Zahlen sind erlaubt, aber nicht als erstes Zeichen, das erste Zeichen muss ein Buchstabe oder _ sein. Umlaute sind zulässig, aber ich rate grundsätzlich davon ab, Umlaute beim Programmieren für *define*, Variablen oder Prozeduren etc. zu verwenden.

Der Begriff, für den Sie einen Platzhalter definieren, kann ein Text, eine Zahl, ein Datum, ein logischer Wert, ein dBASE-Befehl, eine selbst geschriebene Prozedur oder Funktion, ja sogar ein anderes *define* sein. Also alles was Sie auch „ganz normal“ im Code schreiben können, können Sie im Regelfall auch als *define* definieren und dann über dieses *define* im Code platzieren. Das kann man natürlich auf die Spitze treiben, es gilt also immer abzuwägen, ob ein *define* für den jeweiligen Zweck auch wirklich sinnvoll ist. Wann ein *define* Sinn macht und wann es Unsinn ist sagt Ihnen mit der Zeit Ihre Erfahrung.

4.5 Variablen und ihre Geltungsbereiche

Variablen gehören zum täglichen Brot eines Programmierers. Sie werden sie am laufend Band verwenden, oft vielleicht auch ohne sich dessen bewusst zu sein, z. B. als Parameter an Prozeduren oder als Eigenschaften von Objekten. Sie werden sehr selten ein Programm schreiben, das keine Variablen enthält.

Eine Variable ist ein mit einem Namen versehener Zwischenspeicher, der für alles mögliche verwendet werden kann. Für Berechnungen, für Eingaben des Benutzers, zur Formatierung von Zeichenketten, für Vergleiche, als flexible Parameter an Funktionen oder Unterprogramme und vieles mehr. Sie können mit Variablen fast alles machen. Nur schade, dass Sie sie nicht auch zum Pizza holen schicken oder sich den Rücken von ihnen massieren lassen können.

Es gibt diese munteren Gesellen in den verschiedensten Formen. Lokale und globale, statische und private. Dazu die Formular-Variablen, so genannte Applikations-Variablen, vordefinierte dBASE-Variablen und so weiter. Mit allen Varianten von Variablen werden wir uns in diesem Kapitel befassen.

Vorab noch ein Wort zum Begriff „Speicher-Variable“. Er kennzeichnet eine Variable, die sich im Speicher Ihres PCs befindet. Was ist daran besonders? Nichts. Eine Speicher-Variable und eine Variable sind also dasselbe, auch wenn in manchen Texten diese Begriffe so verwendet werden, dass man meinen könnte, es gäbe ausser den Speicher-Variablen noch weitere Arten, Drucker-Variablen oder CD-Variablen vielleicht. Dem ist nicht so, ich kann Sie beruhigen. Der *Inhalt* einer Variable kann natürlich wechseln und kann auch gedruckt oder auf eine CD gebrannt werden, doch die Variable selbst bleibt dabei immer Speicher und dort stirbt sie, sobald der Strom weg ist.

4.5.1 Namen von Variablen

Variablen haben einen Namen, und diesen Namen bekommen sie von Ihnen. Er unterliegt nur wenigen Einschränkungen, Gross- und Kleinschreibung ist zum Beispiel beliebig und gemischt verwendbar. Ihrer Phantasie sind also (fast) keine Grenzen gesetzt. Die wenigen Punkte, die Sie bei der Vergabe von Namen für Variablen unbedingt beachten müssen, sind bei dBWin:

- der Name muss eindeutig sein (also kein dBASE-Befehl etc.)
- es sind Buchstaben, Zahlen und der Unterstrich _ erlaubt
- das erste Zeichen darf jedoch keine Zahl sein
- bis max. 32 Zeichen lange Namen werden unterschieden

Umlaute sind erlaubt, aber Sie wissen, ich rate im Zweifel lieber davon ab.

5. Nützliches, Tips und Tricks

5.1 _app, ein ganz besonderes Objekt

Unter der Bezeichnung *_app* (steht für die Applikation, also Ihr Programm), ist ein globales Objekt von dBWin vordefiniert. Dieses Objekt kann von Ihnen benutzt werden, um grundsätzliche Einstellungen und ein bischen Feintuning an dBWin und an Ihren mit dBWin erstellten Programmen vorzunehmen.

Einige Eigenschaften von *_app* sind schreibgeschützt, können von Ihnen also nur ausgelesen, aber nicht geändert werden. Eine vollständige Übersicht über alle Eigenschaften von *_app* erhalten Sie mit dem Befehl *inspect*, der aber nur innerhalb der dBWin-Entwicklungsumgebung funktioniert.

```
inspect ( _app )
```

ⓘ Leider sind viele Eigenschaften von *_app* nur dürftig dokumentiert. Und was bei *dBASE 2000* noch dokumentiert war fehlt bei *dBASE Plus 2.5* in der Hilfe. Ob das jetzt einfach „nur“ Schlamperei ist oder die Infos weggelassen wurden weil sich hier bei Updates bald Änderungen ergeben, bleibt offen.

Viele Eigenschaften sind anhand ihres Namens oder des Inhalt identifizierbar, andere wiederum sind ein wenig geheimnisvoll. Die folgende Beschreibung soll Licht ins Dunkel bringen und die wichtigsten Teile von *_app* erläutern.

Auch bei der Steuerung von DDE- und OLE-Funktionen ist *_app* hilfreich, aber das wird vermutlich später mal Thema eines eigenes Buchs werden..

Auf alle im folgenden erläuterten Eigenschaften greifen Sie zu, indem Sie einfach *_app.* vor den Namen der Eigenschaft setzen, z. B. *_app.charset.* Und dann verwenden Sie es wie eine „ganz normale“ Variable.

```
sVar = _app.charset          && Inhalt verwenden
_app.statusBar = .f.         && Inhalt ändern (*)
```

(*) sofern der Inhalt einer Eigenschaft von Ihnen änderbar ist. Außerdem überwacht dBWin den Datentyp der Eigenschaften und Sie bekommen einen Fehler, wenn Sie z. B. einer numerischen Eigenschaft einen Text zuweisen.

Groß- und Kleinschreibung ist, wie fast überall in dBWin, nicht relevant. Im Text wurde zumeist die Schreibweise der Online-Doku V 2.5 übernommen.

5.1.1 Wichtige Eigenschaften von *_app*

_app.allowDEOExeOverride*, logisch, Vorgabe *true
Hinter diesem Wortmonster versteckt sich eine flexible Möglichkeit, wie Sie Teile Ihres Programms auslagern und bei Bedarf austauschen können, ohne dass eine Neu-Compilierung des Hauptprogramms nötig ist. Allerdings geht das deutlich über die Themen, die dieser Band bespricht, hinaus. Bei Bedarf und Interesse wird das evtl. mal Teil eines zukünftigen Buchs werden.

_app.allowYieldOnMsg*, logisch, Vorgabe *false
Wenn diese Eigenschaft auf *false* gesetzt ist, werden einige Mechanismen von Windows unterdrückt, während langwierige Operationen und Befehle ausgeführt werden. Das beschleunigt theoretisch den Ablauf des Programms, ob es aber auch praktisch relevant ist hängt sehr von den Umständen ab.

Ist die Eigenschaft dagegen *true*, werden die Mechanismen nicht unterdrückt. Meist sind das interne Abläufe zur Anzeige der Fenster oder um ein Programm zu beenden. Das erlaubt Ihrem Programm theoretisch, schneller darauf zu reagieren. Aber auch hier stehen sich Theorie und Praxis oft gegenüber.

In der Regel hat sowieso Windows das letzte Wort und steuert selbst, welche Meldungen zu einem Programm durchgeschleust werden und welche nicht. Aber Sie können bei Bedarf auch ruhig selbst einmal mit dieser Einstellung experimentieren, ob sie in Ihrem Fall vielleicht eine Verbesserung bringt. Ich habe keine für Anwender wirklich spürbare Änderung entdecken können.

***_app.baseclassName* und *_app.className*, Strings (schreibgeschützt)**
bezeichnen die (Basis-)Klasse selbst, Änderung weder möglich noch sinnvoll.

***_app.charSet* , String (schreibgeschützt)**
enthält den eingestellten Zeichensatz für dBWin, z. B. „DOS437“ oder etwas ähnliches. Der Eintrag hängt von Ihrer Konfiguration des Zeichensatzes in der BDE ab. Sie können diese Info aber auch über eine Funktion abfragen:

```
? charset()                  && ergibt jeweils
? _app.charset               && dieselbe Anzeige
```

***_app.detailNavigationOverride*, numerisch**
Ist ein Detail, das im Zusammenhang mit *rowsets* verwendet wird. Möglich sind die Angaben 0 (Vorgaben wie beim jeweiligen *rowset*), 1 (Details von *rowsets* zugänglich machen) und 2 (*rowset*-Details nicht zugänglich).

5.2 Fehler gehören dazu

Natürlich, besser als Fehler abzufangen wäre es, erst garkeine zu machen. Aber bleiben wir doch bitte realistisch. Jeder Entwickler macht Fehler, jedes Programm enthält Fehler, das eine mehr, das andere etwas weniger. Bei der heutigen Komplexität von Software ist es meiner Meinung nach auch nicht mehr erstes Ziel, „fehlerfreie“ Programme zu schreiben, das ist fast unmöglich. Vielmehr geht es darum, ein Programm möglichst fehlertolerant zu machen.

Was das heisst? Nun, jedes Programm sollte immer und überall gut auf Fehler vorbereitet sein. Sei es durch Gründe die in ihm selbst verborgen liegen, durch äussere „ungünstige“ Umstände, oder weil der Anwender sich „falsch“ verhält.

Das mit den eigenen Fehlern ist so eine Sache. Hier hilft nur eines: Erfahrung. Und fundierte Erfahrung gewinnen Sie nur durch praktisches programmieren. Kein Kurs und kein Buch dieser Welt kann Ihre eigene Arbeit am PC ersetzen. Dennoch werden Sie auch nach Jahrzehnten noch Fehler machen. Das ist so.

Unter Fehlern von anderen müssen Sie wie auch Ihre Anwender leiden. Fehler in Windows, Fehler im Druckertreiber, abgeschmierte Server-Verbindungen und was es da so alles gibt. Das nervt unendlich, das kostet viel Zeit und Geld. Aber auch das ist so. Finden Sie sich damit ab, ändern wird es sich ja eh nicht.

Bleibt als letzte Fehlerquelle der Anwender. Als Entwickler sollten Sie erstens immer mit jeglichem Blödsinn des Benutzers rechnen (das ist jetzt nicht böse gemeint, sondern einfach nur realistisch), und zweitens darauf gefasst sein, dass der Anwender etwas tut, das so an dieser Stelle niemals vorgesehen war.

Wenn z. B. ein Eingabefeld ein Datum erwartet, der Anwender aber den Text „letzten Montag“ eingibt und das Programm daraufhin abstürzt ist zwar auf den ersten Blick der User Schuld, der das Programm nur falsch bedient hat. Aber wenn man es etwas genauer betrachtet liegt es an uns Entwicklern, ein Programm so zu gestalten, dass derartige Falscheingaben im Idealfall garnicht erst möglich sind, oder wenn sie doch mal passieren eben abgefangen werden.

 Alle Eingaben des Anwenders sollten Sie daher eingehend testen und sicherstellen, dass möglichst nur solche Eingaben überhaupt erlaubt sind, die gerade auch Sinn machen. Hier heisst es im Zweifel eben testen, testen, testen. Auch und besonders Grenzfälle und extreme Werte sollten Sie prüfen. Dazu nur ein einfaches Beispiel: bei der Eingabe eines numerischen Werts können Sie die Zahlen 1, 10 und 100 testen. Das ist auch ok, aber wirklich spannend wird es doch erst, wenn Sie 1.23, 0, -1, 65535 oder ähnliche Daten eingeben.

5.3 Eigene *.EXE-Programme erstellen

Nichts leichter als das, wenn Sie erst einmal ein Projekt haben und Ihr Code zumindest soweit korrekt ist, dass er sich fehlerfrei kompilieren lässt.

Wenn Sie ein Projekt laden und der Projekt-Explorer das aktive Unterfenster von dBWin ist, gibt es auch ein Hauptmenü namens *Erstellen*. Aus mir völlig unverständlichen Gründe ist dieser Menüpunkt aber nur sichtbar, wenn der Projekt-Explorer aktiv ist, was die Arbeit mit dBWin leider unnötig erschwert.

Dort können Sie eine einzelne Datei kompilieren, die Sie gerade in der Liste der Dateien des Projekt-Explorers ausgewählt haben. Oder Sie können alle Dateien eines Projekts in einem Durchlauf kompilieren. Ebenso können Sie aus den kompilierten Objektdateien ein EXE-Programm generieren lassen.

Für die wesentlichen Funktionen gibt es natürlich auch wieder Befehle:

```
compile <dateiname>          && eine Datei kompilieren
compile <dateiname> auto     && (siehe Text unten)
compile *.prg                && alle .prg kompilieren
compile *.wfm                && alle .wfm kompilieren
compile *.*                  && alles kompilieren
```

Der Zusatz *auto* kompiliert nicht nur die angegebene Datei, sondern auch alle weiteren, die darin per *set procedure to <dateiname>* eingebunden werden. Damit ist *auto* sehr gut dazu geeignet, alle relevanten Dateien eines grösseren Projekts zu kompilieren, zumindest wenn Sie *set procedure to* ... verwenden.

 Achtung, *compile *.** kompiliert alles was nicht bei drei auf´m Baum ist. Der Befehl erkennt leider nicht, ob eine Datei anhand ihrer Endung oder ggf. anhand ihres Inhalts überhaupt zum kompilieren geeignet ist. Damit werden sogar Logdateien, Texte, Bitmaps und was sich sonst noch so alles im aktiven Verzeichnis befindet kompiliert. In der Folge hagelt es Fehlermeldungen ...

Der Menüpunkt *Alle neu erstellen* compiliert alle Dateien eines Projekts und erstellt danach die EXE, ist also ähnlich (aber nicht identisch) wie die Befehle

```
compile <hauptdateiname> auto     && kompilieren
build from <projektname>          && EXE erstellen
```

ⓘ Der Befehl *compile *.prg* nimmt sich aller kompilierbaren *prg*-Dateien im aktuellen Verzeichnis an, während die Menübefehle *Alle kompilieren* und *Alle neu erstellen* alle relevanten Dateien des aktiven Projekts kompilieren.

5.4 Die dBASE Runtime

Die Runtime, zu dt Laufzeit-Umgebung, ist auf allen PCs nötig, auf denen die von Ihnen erstellten EXE-Programme ausgeführt werden sollen. Die Runtime wird separat von CD installiert, dazu gibt es eine eigene Auswahl im Setup.

Installation der Runtime erfolgt separat

Die Installation selbst ist denkbar einfach. Sie geben ein Zielverzeichnis ein, klicken ein paarmal auf **[OK]**, nicken ggf. noch die Lizenzbedingungen ab, und der Rest geschieht von selbst. Danach evtl. den PC neu starten, fertig.

ⓘ Auf Ihrem Entwicklungs-PC, der die komplette Entwicklungsumgebung von dBWin enthält, muss die Runtime nicht mehr zusätzlich installiert werden.

Die Runtime dürfen Sie als Besitzer einer gültigen dBWin-Lizenz zusammen mit Ihrem dBWin-Programm an Ihre Kunden ausliefern bzw. dort installieren. Weitere Lizenzgebühren fallen hierbei weder für Sie noch für Ihre Kunden an. Zumindest ist das derzeit so, und man kann nur hoffen dass es auch so bleibt.

Achten Sie darauf, dass zu jeder Haupt-Version auch stets die passende Runtime installiert ist. Ein mit *dBASE 2000* erstelltes Programm läuft nicht mit der Runtime von *dBASE Plus*, umgekehrt funktioniert´s natürlich auch nicht. Das ist auch ein wichtiger Punkt, wenn Sie Ihre Programme erstmals mit einer neuen Haupt-Version von dBWin kompilieren und die Updates Ihren Kunden schicken. Dabei muss dann i.d.R. immer eine neue Runtime installiert werden! Bei kleineren Updates, z. B. von 2.5 auf 2.6, ist das dagegen oft nicht nötig.

ⓘ Nach meiner Erfahrung können Runtimes von verschiedenen Versionen parallel auf einem PC aktiv sein. So ist der Betrieb von z. B. *dBASE 2000* und *dBASE Plus* Programmen auf demselben Computer i.d.R. problemlos möglich.

6. Anhang

6.1 Ein Wort zum Abschied

Liebe Leserin, lieber Leser, ich möchte mich an dieser Stelle herzlichst für Ihr Interesse, Ihre Geduld und Ihr grosses Engagement bedanken, das Sie gezeigt haben, wenn Sie das Buch bis hierhin durchgearbeitet haben.

Sie werden sicher davon profitiert haben. Der eine vielleicht etwas mehr, die andere evtl. etwas weniger, da hängt von den jeweiligen Vorkenntnissen ab. Und davon, wie Sie die gezeigten Beispiele in der Praxis mitgemacht haben.

Wie bereits erwähnt ist dies nur ein Buch einer Serie zu dBWin. Infos zu weiteren geplanten, in Arbeit befindlichen oder bereits erhältlichen Themen finden Sie auf der Website des Autors, die auf den ersten Seiten vorgestellt wurde. Ein Blick dorthin lohnt und Sie bleiben immer auf dem Laufenden. Weitere Bände für Einsteiger, Fortgeschrittene und Profis sind geplant.

Wenn Sie Fragen, Anmerkungen, Lob und Tadel oder sonstiges zu diesem Buch loswerden wollen, dann tun Sie es bitte. Wir alle lernen aus Fehlern, profitieren von Anregungen und Meinungen von anderen, und es gibt wohl kaum etwas, das man nicht immer noch ein bischen besser machen könnte.

Auf meiner privaten Website finden Sie auch einen kleinen Fragebogen zum Download. Sie würden mir sehr helfen, wenn Sie ihn ausfüllen. Vielen Dank.

Sie finden im Anhang des Buchs keine Web-Links zu dBASE unter Windows. Dazu ist das Internet zu vielen und zu schnellen Änderungen unterworfen. Die Gefahr, dass viele der hier abgedruckten Links nicht mehr funktionieren ist einfach zu gross. Es würde Sie und mich nur ärgern, wenn ich hier viele Links aufführe, von denen dann nach ein paar Monaten sowieso die Hälfte nicht mehr funktionieren. Aber dazu gibt es ja die Suchmaschinen ...

Abschliessend noch der Hinweis, dass es bei künftigen Versionen von dBWin, die sich stark von der für dieses Buch verwendeten Version unterscheiden, vermutlich eine Ergänzung oder je nach Bedarf auch eine Neuauflage dieses Bandes geben wird. Kleinere Ergänzungen und Korrekturen können Sie bei Bedarf aber auch auf meiner Website finden.

Nun bleibt mir nur noch eines: *Ihnen viel Erfolg und natürlich auch viel Freude bei der Programmierung mit dBASE unter Windows zu wünschen!*

6.2 Weitere Bücher dieser Reihe

Dieses Buch ist nur eines einer ganzen Reihe von neuen Büchern zu dBWin. Bei Fertigstellung dieser Ausgabe sind die folgenden Titel verfügbar/geplant:

Band 1 - Einführung (Einsteiger, DOS-Umsteiger)
Installation - Konfiguration von dBWin und BDE - kurze Einführungen zu Befehlsfenster und Regiezentrum - Tabellen anlegen und verwalten - Daten erfassen, ändern und löschen - Formulare erstellen - Beispielprogramme u. a.

Band 2 - Grundlagen (Einsteiger, DOS-Umsteiger)
Weitere Details zu Befehlsfenster, Regiezentrum und Projekt-Explorer - Quelltext-Editor - Projekte verwalten - EXE-Programme - INI-Dateien - Basiswissen Programmierung - Variablen, Geltungsbereiche, Datentypen - Defines und Präprozessor - Formular-Komponenten - globales Objekt _app - Parameter an Programme - Datenbankfelder - Indizes - Tips und Tricks u. a.

Band 3 - Klassen und Objekte (Fortgeschrittene)
Basisklassen - eigene Objekte - objektorientierte Programmierung - Vererbung (in Planung, irgendwann 2006, bitte auf der Website des Autors informieren).

ⓘ Insbesondere die Bände 1 und 2 ergänzen sich in idealer Weise. Es gibt nur sehr wenig Überschneidungen und es empfiehlt sich, dBWin mit Band 1 kennenzulernen und danach mit Band 2 tiefer in diese Materie einzusteigen. Das in den beiden Bänden 1 und 2 vermittelte Wissen wird in allen folgenden Ausgaben, die dann Schwerpunktthemen haben, als bekannt vorausgesetzt.

Weitere Ausgaben sind geplant, bzw. teilweise bereits in Arbeit. Mögliche Themen sind u. a. Schnittstellen zu anderen Programmen (Excel, Word u.a.), Datenformate, SQL, OLE, DEO, DLL, Programmierung der Windows -API, Debugging und Fehlermanagement, Profitips aller Art, Drucken und Reports, Internet-Anwendungen, Umstieg von DOS, Windows-Spezialitäten etc.

Wenn Sie das hier lesen gibt es vielleicht bereits weitere Ausgaben. Einfach mal ab und an auf meiner Homepage vorbeischauen (in jedem Buch auf den ersten Seiten zu finden), oder bei (Internet-)Buchhändlern stöbern.

Bis auf weiteres werden die Bücher als „Books on Demand“ herausgegeben. Das hält Aufwand und Kosten im Rahmen und ermöglicht dennoch eine gute Verbreitung über die üblichen Vertriebswege. Alle Bände sind sowohl direkt bei mir, als auch bei allen wichtigen Internet-Händlern und natürlich auch in (fast) jedem „normalen“ Buchladen zu beziehen. Falls Ihr lokaler Buchhändler keine „Books on Demand“ bestellen will kaufen Sie künftig eben woanders ...

6.3 Stichwortverzeichnis

E

F

G

H

I, J

K

L

M

N

O

P

Q

R

S

T

U

V, W

X, Y, Z