

OPL Workshop 4 - Datenbanken: Neu/Ändern/Löschen

Ulrich Krinzner, 11/1998; Originalvorlage OPL Quelle: Dieter Gansberger

Unser bisheriger Exkurs in die OPL-Welt war eher einem spielerischen Thema gewidmet. Heute nähern wir uns dem "Ernst des Lebens" mit dem Thema "Datenbanken". Die allerdings finden sowohl in privater als auch geschäftlicher Umgebung Anwendung. Haben Sie vor, endlich Ihre CD Sammlung (oder was auch immer) zu katalogisieren, sind sie hier und heute in diesem Kurs richtig und der Begriff vom "Ernst des Lebens" ist damit auch schon wieder relativiert ...

Da Datenbanken ohne Zweifel von großem Nutzen sind, haben wir in verschiedenen Ausgaben bereits ausführlich darüber geschrieben. In einigen Ausgaben ab 4 /Dez 97 erfahren Sie, was Datenbanken eigentlich sind und wie man mit ihnen umgeht. Daher heute nur ein kurzer Ausflug.

Klären wir zuerst den Begriff Datenbank, der immer wieder zu Mißverständnissen zwischen Gesprächspartnern führt. (Computer-) "Datenbank" meint eigentlich eine oder mehrere Dateien, in denen in kontrollierter Art Daten abgelegt sind. Im lockeren Sprachgebrauch wird heute aber auch schon mal das zugehörige Verwaltungsprogramm als "Datenbank" bezeichnet. Da das Datenbankformat, also die Art, wie die Daten in der Datenbankdatei abgelegt werden, meist an ein bestimmtes Programm gekoppelt ist, ist die Begriffsverwischung verständlich. Nur: Man sollte im Zweifelsfall wissen, wovon man eigentlich redet.

Bei unserem Beispiel-Programm handelt es sich um den Typ "schlichter Merker", das weniger Ansprüche an das Einarbeiten stellt - und irgendwo muß man ja schließlich anfangen. Eine derartige Datenbank stellt man sich am besten als Tabelle vor, der man in Vorbereitung kommender Einträge zuerst einmal Spaltenüberschriften verpaßt, z.B. für ein schlichtes Telefonbuch NAME und NUMMER. Damit ist geregelt, daß alle folgenden Einträge einem festen Schema unterliegen. So wird bei einem ersten Eintrag "Franz" in die Spalte NAME geschrieben, "099-12321" landet in der NUMMER Spalte. Damit ist auch schon ein Datensatz komplett.

Datensatz in der Tabellen-Analogie ist also die Zeile, die alle zusammengehörenden Daten zusammenfaßt. Die Einordnung der Einzelkomponenten des Datensatzes erfolgt über "Felder" - in der Tabelle sind das die Spalten. Um zu wissen, um welches Feld des Datensatzes es gerade geht, benutzt man den "Feldbezeichner" - die Spaltenüberschrift der Tabelle. NAME in Datensatz 1 verweist also wieder auf den den Eintrag "Franz".

Um es noch anders zu verdeutlichen: Nehmen Sie einen Karteikasten. Der steht als Synonym für die "Datenbank". Wenn Sie ihn öffnen, finden Sie jede Menge Karteikarten - die "Datensätze". Besehen Sie sich eine Karte näher, zeigt sie in unserem Beispiel zwei "Felder": NAME: Franz und NUMMER: 099-12321. Jeder, der neue Karteikarten beschreibt, wird sich an dieses Schema halten müssen. Ansonsten wird man wohl nicht lange Freude an seiner Adreß-Sammlung haben.

Datenbankbefehle

Werfen wir nun einen gerafften Blick darauf, mit welchen OPL-Zutaten wir eine Datenbank anrühren. Die genauen Rezepte kommentieren wir dann in der Besprechung des Quelltextes. Ansonsten bauen wir auf unsere Ausführungen in den vorhergehenden Workshops auf. Die entscheidenden Befehle lauten:

CREATE: sorgt in der Stunde Null dafür, daß überhaupt eine Datenbank-Datei angelegt wird,

OPEN: öffnet eine bestehende Datenbank,

USE: benennt die Datenbank-Datei, mit der gerade gearbeitet werden soll, denn es ist erlaubt mit mehreren Dateien gleichzeitig zu arbeiten,

CLOSE: schließt die aktuelle Daten-Datei,

APPEND: fügt einen neu einzutragenden Datensatz an eine bestehende Datei an,

UPDATE: hilft beim Ändern eines aktuellen Datensatzes,

FIRST, LAST, POSITION: Da die Datenbank-Datei im Prinzip nur eine Art Tabelle ist, zeigt das Programm gewissermaßen mit dem Finger auf die Zeile (den Datensatz), die die aktuelle sein soll. Diesen Zeiger kann man selber setzen, um mit einem ganz bestimmten Datensatz zu arbeiten - entweder auf den ersten oder letzten Datensatz, oder auf einen ganz bestimmten anderen.

POS: ermittelt die gerade aktuelle Datensatznummer,

COUNT: zählt, wieviele Datensätze die aktuelle Datenbank-Datei überhaupt enthält. "0" steht für "keine Datensätze",

NEXT, BACK: setzt den Zeiger des aktuellen Datensatzes auf den nächsten bzw. vorhergehenden Datensatz.

Nachdem Sie die grundlegenden Befehle kennengelernt haben, werden wir überlegen müssen, was wir eigentlich erreichen wollen und wie wir das tun. Halten wir uns also zuerst vor Augen, was nötig ist.

OK. Wir nehmen wir uns also vor, eine Datenbank mit Namen "KUNDEN.ODB" zu erstellen und zu verwalten. Diese soll der Übersichtlichkeit halber lediglich zwei Felder (Name, Telefonnummer) enthalten.

Klar ist, daß wir die Datenbank-Datei am Beginn anlegen oder öffnen müssen, sonst läßt sich wohl schlecht mit ihr arbeiten. Folgt die Darstellung des Inhaltes der Datei auf dem Bildschirm. Wir möchten nur den jeweils aktuellen Datensatz anzeigen. Weil nach dem Öffnen der Datenbank immer der erste Datensatz der aktuelle ist, muß man die anderen irgendwie sichtbar machen können. Das Durchstöbern soll mit den Pfeiltasten <rechts> bzw. <links> realisiert werden ("rollen").

Der angezeigte aktuelle Datensatz soll bearbeitet werden können; wir beschränken uns in dieser Folge des Workshops auf NEU hinzufügen, ÄNDERN und LÖSCHEN. Die Einleitung der entsprechenden Unterprogramme soll über ein Menü oder Tastaturkürzel erfolgen. Auch die Beendigung des Programmes, mit dem auch das Schließen der Datenbank-Datei verbunden sein soll, muß natürlich gezielt möglich sein. Das wird ebenfalls über ein Menü oder ein Tastaturkürzel erfolgen.

Unsere Wünsche ordnen wir nun in eine Programmstruktur ein.

Ganz wie beim Schachspiel, bei dem man zuerst die Figuren auf dem Brett aufstellen muß, richtet man bei Programmen in einer ersten Phase einige Dinge vor - man initialisiert Variablen etc.. In unserem Programm veranlaßt die allererste Prozedur "startup:" den Aufruf der (in unserem Beispiel) einzigen Initialisierungs-Prozedur "iniFile:". Die legt die Datenbank-Datei an oder öffnet sie und nimmt die Bildschirmdarstellung vor. Unmittelbar danach überträgt sie die Programmlast an die Hauptprozedur "main:".

Wie schon in den vorhergehenden Folgen praktiziert, wird in "main:" in einer Schleife die Tastatur abgefragt. Je nach Tasteingabe werden dann die verschiedenen Unterprogramme angesteuert, die die "Knechtsarbeit" gemäß unseren oben skizzierten Vorstellungen zu leisten haben. Die Einzelheiten lernen wir anhand der Programm-Module kennen.

Achtung! Nicht in jedem Falle kommt man auf allen Psion Geräten mit den gleichen Programm-Texten hin. Da wir alle aktuellen Geräte berücksichtigen wollen, haben wir folgenden Kompromiß geschlossen: Der Basis-Programmtext ist für den Serie 3a/c/mx ausgelegt. An Stellen, die abweichend sind, haben wir die Alternativen mit den Geräte Kürzeln (SN = Siena, WA = WorkAbout, S5=Serie 5) im Programmtext nach einem Kommentarbefehl ("REM") in eckigen Klammern angegeben. Die Unterschiede resultieren im wesentlichen aus den verschiedenen Bildschirmgrößen und dem EPOC16 / EPOC32 Betriebssystem.

Ein zweites Handicap haben wir so gelöst: Aus Platzgründen haben wir Kompromisse in der Übersichtlichkeit des Programmtextes eingehen müssen. In OPL ist es erlaubt, mehrere Befehle in eine Zeile zu schreiben, wenn man sie durch Leerzeichen und Doppelpunkt voneinander trennt. Am Ende der Zeile jedoch ist eine derartige Trennung nicht mehr erforderlich. Findet man dort dennoch einen Doppelpunkt, handelt es sich um einen Unterprogrammaufruf! Der Doppelpunkt folgt in diesem Falle ohne Leerzeichen unmittelbar auf den Unterprogramm-Namen.

Das große Sezieren ...

Gehen wir nun munter ins chirurgische Praktikum. Den kompletten Programmtext finden Sie diesmal am Textende - das bewahrt den Überblick etwas leichter.

Die Startprozedur "startup:" dürfte Ihnen wohl keine Rätsel aufgeben. Spannender wird es aber bereits in der Datei-Initialisierungs-Prozedur "iniFile:". Sie prüft zuerst indirekt, ob das Verzeichnis MEINE_DB existiert, indem sie mit MKDIR (make directory, Verzeichnis erstellen) veranlaßt, ein Verzeichnis zu erstellen. Wenn dieses bereits existiert - und das ist spätestens beim zweiten Start des Programmes der Fall, gäbe das normalerweise eine Fehlermeldung. Die wird jedoch clever mit der Anweisung TRAP unterdrückt. Aufgrund von TRAP kommt Fehlerbehandlungsroutine nicht zum Zuge - das Programm wird einfach fortgesetzt.

Im nächsten Schritt wird geklärt, ob die Datei "KUNDEN.ODB" bereits existiert. Im Bedarfsfall wird sie neu mit CREATE erstellt, andernfalls eine vorhandene mit OPEN lediglich geöffnet. TRAP wird diesmal verwendet, um eine Fehlermeldung für den Fall zu verhindern, daß die Datei bereits irgendwie in Gebrauch ist. In der hier verwendeten Schreibweise für den Dateinamen ohne Nennung des Laufwerkes befindet sich die Datei angelegte Datei immer auf dem Laufwerk "C:" (Serie 5) oder "M:" (alle anderen).

Die erforderlichen Parameter hinter CREATE bzw. OPEN sind:

- der vollständige Dateiname,
- ein Buchstabe der im Zusammenhang mit USE erlaubt, offene Datenbankdateien anzusprechen (es können mehrere gleichzeitig benutzt werden, die Befehle können aber immer nur auf eine angewandt werden),
- die Variablen, die für die Feldbezeichner der Datensätze stehen. Letztere benötigen im Gegensatz zu der üblichen Verfahrensweise keine vorherige Deklaration!

showknd:

Den Abschluß von "iniFile" bildet die Bildschirmdarstellung der geöffneten Datenbank durch das Unterprogramm "showknd:". Daß wir uns hier laufend mit verkrüppelten Namen für die Prozeduren abgeben müssen, hängt einfach mit den Festlegungen für OPL zusammen und ist häufig der Grund, sich mit kurzen Anglismen zu behelfen. Die Situation entspannt sich erst für Besitzer der Serie 5.

Die gleich zu Anfang von "showknd:" angesprungene Hilfsprozedur "db2knd:" besorgt die notwendigen Daten, die auf dem Bildschirm dargestellt werden sollen. Sie holt sie aus dem aktuellen Datensatz. Wenn die Datei gerade erst geöffnet wurde, ist das der allererste. Dabei sorgt USE A für Klarheit, welche der offenen Dateien zu verwenden ist. In unserem Programm ist das der reine Luxus, da ja nur mit einer einzigen Datei gearbeitet wird, dient aber bereits der Vorbereitung auf Programmiererweiterungen.

Auf die Daten in der Datenbank greift man mit ihren Feldbezeichnern und dem vorangestellten "Verwaltungsbuchstaben" (engl. handle) zu und weist sie einer allgemeinen Variablen gleichen Typs zu, beispielsweise "name\$ = A.name\$". Die gleichen Bezeichnungen ("name\$") beißen sich hier nicht, das vorangestellte "A" und der Punkt sorgen für den kleinen Unterschied.

Anschließend macht "showknd:" Gebrauch davon, Sonderzeichen (mit einem ASCII-Wert unter 32) auf dem Bildschirm auszugeben, indem die Funktion CHR\$() verwendet wird. Der als Argument verwendete Wert sorgen dafür, daß ein Tabulator (ASCII-Wert 9) ausgegeben wird. Wenn es überhaupt einen Datensatz in der Datenbank-Datei gibt, sagt die Bildschirm-Routine sogar, wieviele Datensätze insgesamt vorhanden sind (COUNT) und zeigt, welchen Satz man gerade bearbeitet (POS). Enthält die Datenbank keinen Datensatz, heißt es dann natürlich "0/0".

main:

Das Programm hat schon einen recht langen Weg zurückgelegt, bevor es so richtig in Tritt kommt. Aber durch den Aufruf von "main:" geht es nun so richtig los! Für den Serie 5 sind die Änderungen zu beachten, die aus den anderen Nutzungs- und Bedienungseigenschaften resultieren. So wurde die Psion-Taste durch die Strg-Taste ersetzt und Strg liefert -gleichzeitig mit anderen Tasten gedrückt- einen anderen Wert, als die Psion-Tasten-Kombination. Außerdem werden Sie feststellen, daß sich einige Buttons und Bildschirm-Elemente mit dem Stift bedienen lassen, ohne extra programmiert worden zu sein - ein Service von EPOC32!

In einer ewigen Schleife wird die Tastatur mit GET auf bestimmte Werte hin abgefragt. Um entscheiden zu können, ob auch die <Psion>-Taste mitgedrückt wurde, wird die Funktion KMOD sofort nach jedem GET abgefragt. Die Variable mod% "merkt" sich das Ergebnis, bis es benötigt wird. Danach wird umgehend über die Entscheidungsstruktur IF / ELSEIF / ENDIF geprüft, ob die Menütaste, eine andere Taste zusammen mit der <Psion>-Taste oder eine Sondertaste mit einem Rückgabewert größer als 255 (z.B. Pfeiltasten) gedrückt wurde. In jedem dieser Fälle setzt das Programm mit einer separaten Unteroutine fort, alles andere fällt unter den Tisch und die Schleife beginnt von vorn.

Die Unterrouinen im einzelnen:

showmen%:

... sorgt dafür, daß das Psion-typische Menü aufgeblendet wird. Wenn Sie das Program laufen lassen, kommen Sie am Beispiel recht schnell dahinter, wie es zu programmieren ist. Die grundsätzliche Anordnung beginnt immer mit dINIT, stellt das Menü mit mCARD zusammen und endet mit der Aktivierung durch den Befehl MENU. Wichtig zu wissen ist, welche Werte wie in die Variable "keycode%" kommen. Hinter jedem Menüeintrag finden Sie eine Zahlenangabe (in einer quasi-Konstanten versteckt, z.B. %a für 97). Dieser Wert wird nach dem Drücken einer Taste durch den MENU-Befehl übernommen und kann dort auf verschiedene Art verwendet und zu Entscheidungen benutzt werden. Ein <Esc> (das Verlassen des Menüs ohne eine Auswahl getroffen zu haben) liefert eine Null zurück. Wir nutzen hier die Zuordnung zur Variablen "keycode%", mit der dann das Unterprogramm "DoMenu: ()" aufgerufen wird.

DoMenu: ()

...bietet gleich mehrere "Spitzfindigkeiten". Wird die Routine von "showmen%:" aus aufgerufen, kann der übergebene Parameter k% die Werte von %x, %n, %a, %l oder Null annehmen. Das ist der einfache Fall. Ruft aber "main:" von der Stelle "ELSEIF (mod% AND 8)" die Routine mit dem Übergabewert "taste%" auf, gibt es ein Problem. "taste%" enthält nämlich nicht den reinen Code der gedrückten Taste! Durch das Mitdrücken der Psion-Taste wird dem ursprünglichen Wert eine Wert von 512 hinzuaddiert, der das korrekte Verhalten der Unteroutine verhindern würde.

Durch den Rechenoperator AND (UND) wird die zusätzliche 512 wieder abgestreift und der reine Tasten-Code wieder "sichtbar". Man stelle sich das wie ein digitales Sieb vor. Da der Wert einer Zahl von ihrem Platz der Einsen und Nullen in der digitalen Darstellung abhängt, kann man sie dadurch verändern, daß man nur an den Stellen ein Loch in das digitale Sieb bohrt, die man weiterverwenden will (man spricht auch von "Maskieren").

Ist in einem Sieb über dem Loch etwas, kann es durch das Loch fallen und kann darunter weiterverwendet werden. Ist aber erst gar nichts über dieser Stelle im Sieb, kann auch nichts darunter ankommen. Es kann aber auch nichts unter dem Sieb ankommen, wenn an der betreffenden Stelle im Sieb kein Loch ist, egal ob darüber etwas ist oder nicht.

Der Programmautor benutzt hier hexadezimale Zahlen, weil diese nach Eingewöhnung eine bessere Übersicht gewährleisten (tatsächlich!). Durch die UND-Verknüpfung des Wertes k% mit dem hexadezimalen Wert \$FF (dez. 255) werden alle Zahlen auf Werte unter 255 beschnitten.

Das Ergebnis wird durch die inzwischen bekannte Funktion CHR\$() in einen String verwandelt und der Variablen "k\$" zugeordnet. Die Funktion LOC sucht nun in dem Inhalt des Strings "menuops\$", ob darin irgendein Zeichen mit dem Inhalt von "k\$" übereinstimmt. Das ist immer der Fall, wenn die <Psion>-Taste zusammen mit der x-, a-, n- oder l-Taste gedrückt wurde oder eine entsprechende Auswahl in der Menüansicht mit <Enter> bestätigt wurde. Alles andere wird wegen der "IF"-Bedingung nicht verfolgt.

Ein besonderes Verfahren ermöglicht es, Prozeduren über Strings aufzurufen. Genau davon machen wir Gebrauch. Beim ersten Mal ist das Verfahren recht schwer zu durchschauen. Kopieren Sie es einfach! Mit Hilfe des "@-Operators" kann man so in Abhängigkeit vom Ergebnis einer String-Operation, die wir uns gerade angesehen haben, das Programm steuern. In unserem Beispiel ist das Programm deshalb in der Lage, wahlweise die Routinen "wahlx%:", "wahl%:", "wahla%:" oder "wahl%:" anzuspringen.

rollen: (taste%)

Nur wenn die Datenbank mehr als einen Eintrag enthält, macht es Sinn, sich in der Datenbank zu bewegen (zu "rollen"). Durch gezielte Abfragen findet das Unterprogramm heraus, bei welchem Datensatz es sich befindet und ob überhaupt Veränderungen möglich sind - über Beginn und Ende hinaus geht es nämlich nicht, zumindest nicht in unserem Beispiel.

Am Ende jeder Veränderung wird mit "showknd:" der Bildschirm mit aktualisierten Werten wieder aufgebaut. Gibt es keine Veränderung, erlaubt "GIPRINT", eine kurzzeitig am Bildschirm erscheinende Meldung zur Information abzusetzen. Den Rest können Sie mit dem Wissen aus obigem Vorspann inzwischen selber interpretieren.

Was treibt nun das Programm, wenn die einzelnen Menüpunkte aufgerufen werden?

wahlx%: bzw. wahle%:

Besonders einfach ist das für den Programmausstieg zu beantworten. Die Routine "wahlx%:" ("wahle%:" am Serie 5!) ist auch schön kurz. Als erstes wird die aktuelle Datenbank mit CLOSE geschlossen. "USE A" ist wiederum nur Vorbereitung auf Kommendes und könnte hier eingespart werden. Mit einem lässigen "STOP" endet das Programm im wohlverdienten Ruhezustand. So einfach ist das, keine Geheimnisse!

Umfangreicher sind die anderen Routinen. Tragen wir zuerst neu einen Namen und Telefonnummer ein: "wahl%:".

wahl%:

Damit keine Verarbeitungsfehler mit den allgemeinen Variablen "name\$" und "tel\$" geschehen, werden ihre Inhalte zuerst vorsorglich gelöscht. Das übernimmt die Prozedur "delkndva:". In unserem Beispiel müßte das nicht unbedingt ein Unterprogramm machen, aber in Hinsicht auf Erweiterbarkeit lohnt sich der Aufwand schon. Folgt der Aufruf von "ediknd%:", inklusive der Übergabe eines Textes. Wechseln wir dorthin und sehen, was passiert.

ediknd%: ()

In "ediknd%: (TEXT)" wird ein Psion-Dialog benutzt, um Eingaben zu realisieren. Dialoge beginnen immer mit dem einleitenden Befehl "dINIT". "dINIT" kann man einen String zuordnen, der dann im Kopf des Dialoges angezeigt wird.

Das Ende wird grundsätzlich durch den Befehl "DIALOG" gebildet. Dazwischen kann es recht unterschiedlich aussehen. Konkret haben wir es bei "dEDIT " mit einer Eingabemöglichkeit für Strings zu tun. Die hier auf den Befehl folgenden Parameter sind:

- die Variable, die die Eingabe aufnehmen soll, und
- ein Text, der den Nutzer informiert, was er eingeben ("editieren") soll.

Die Eingaben werden erst übernommen, wenn der Dialog regelgerecht beendet wird (drücken von <Enter>). Im Falle von <Esc> werden die verwendeten Variablen nicht in ihrem Wert verändert - bei uns bleiben sie also leer.

"dBUTTONS" bietet eine Button-Ansicht, die über die Art informiert, wie der Dialog verlassen werden kann. Das ist keine Pflicht - sieht aber gut aus und ist äußerst benutzerfreundlich. Die Zahlen-Angaben hinter den einzelnen Texten werden vom Psion automatisch in einen Beschriftungstext umgesetzt: 27 als "Esc" und 13 als "Enter". So wird die Benutzung optisch unterstützt. Aber aufpassen! Wenn der Dialog durch <Enter> beendet wird, ist der Rückgabewert in die lokale Variable "erg%" auch "13" (wie bei "dBUTTONS" angegeben), <Esc> gibt aber "0" zurück, nicht etwa "27"! Das Ergebnis des Dialoges wird schließlich an das aufrufende Programm "wahl%: ()" mittels "RETURN erg%" zurückgegeben.

Noch einmal wahl%:

Das verhält sich nun je nach dem zurückgegebenen Ergebnis:

- Die allgemeinen Variablen "name\$" und "tel\$" werden erneut geleert und letztendlich zu "main:" zurückgekehrt, oder
- der Eintrag durch Übertragen der Variableninhalte auf die Datenbankvariablen "A.name\$" und "A.tel\$" der Datenbank zur Verfügung gestellt und endgültig mit APPEND dort eingebracht. Am Ende wird wieder einmal der Bildschirm mit den neuen Ergebnissen aufgefrischt.

Nachdem wir nun etwas in der Datenbank stehen haben, können wir die Inhalte auch ändern oder löschen. Bevor wir jedoch alles wieder durch Löschen vernichten, werden wir es zuvor mit "waha%:" ändern.

wahl%:

Nach der langen Erläuterung zu "wahl%:" sind Sie gut gewappnet, um sich diese Routine selber zu Gemüte zu führen. Der einzige Unterschied: Der Befehl "UPDATE" löscht den aktuellen Datensatz und ersetzt ihn durch die frische Eingabe. Voila!

wahl%:

Ähnlich sollte Ihnen auch die Löschroutine "wahl%:" verständlich sein. Der neue Befehl, der für das Öffnen und Schließen des Mülleimerdeckels und den lautlosen Tod des aktuellen Datensatzes verantwortlich zeichnet, lautet hier "ERASE". Achten Sie einmal darauf, wie hier die Buttons im Dialog beschriftet sind und wie die Auswertung vorgenommen wird.

Damit haben Sie es für heute einmal mehr geschafft. War doch alles garnicht so schwierig, oder? Sie haben nun alle Kenntnisse, um die Datenbank auf Ihre speziellen Belange zuzuschneiden. Versuchen Sie eine Ergänzung der Felder durch Hinzufügen von Vornamen, Strasse, PLZ und Ort. Sie werden sehen, es ist garnicht so kompliziert.

Den Programmtext finden Sie auf unserer Web-Site <http://www.palmtop-pro.com/> in der Download-Sektion. Dort haben wir für die werdenden Profis unter den Lesern auch noch den Quellcode eines vergleichbaren Programmes hinterlegt, das bereits etwas umfangreicher ist und fortgeschrittenere Programmieretechniken verwendet.

Die Möglichkeit, die die Datenbanken so effektiv verwenden läßt (das Suchen) , lernen Sie im nächsten Workshop kennen. Sie sind doch sicherlich wieder dabei?

REM Applikation fuer Psion Workabout, Serie 3/3a/3c/3mx, Siena, Serie 5

```

PROC startup:      REM Startupperoedur
    GLOBAL filenam$(130), basenam$(10), name$(20), tel$(20)
    REM [nur S5/ Courier 13pt fett: FONT 268436068,1]
    iniFile:
    main:
ENDP

PROC iniFile:
    basenam$="\MEINE_DB"
    TRAP MKDIR(basenam$)
    filenam$ = basenam$ + "\" + "KUNDEN.ODB"
    IF NOT EXIST(filenam$)
        TRAP CREATE filenam$, A, name$, tel$
    ELSE
        TRAP OPEN filenam$, A, name$, tel$
    ENDIF
    showknd:
ENDP

PROC main:
    GLOBAL taste%, mod%
    DO
        taste%=GET : mod%=KMOD
        IF taste%=290      REM Menue-Taste
            showmen%:
        ELSEIF (mod% AND 8)      REM Psion-Taste mitgedruekt
            DoMenu:(taste%)
        REM [S5: die letzten beiden Zeilen voellig ersetzen durch:]
        REM ELSEIF (mod% AND 4)      REM Strg-Taste mitgedruekt
        REM taste%=taste% + 96      REM spez. Anpassung fuer S5
        REM DoMenu:(taste%)
        ELSEIF (taste% AND $FF00) REM nur noch taste%-Werte ueber 255
            IF taste%=259      REM Pfeil nach links
                rollen:(taste%)
            ELSEIF taste%=258      REM Pfeil nach rechts
                rollen:(taste%)
            ENDIF
        ENDIF
    UNTIL 0
ENDP

PROC showmen%:
    LOCAL keycode%
    mINIT
        mCARD "Datei","Beenden",%x REM [S5: mCARD "Datei","Beenden",%e]
        mCARD "Kunden","Neu",%n,"Aendern",%a,"Loeschen",%l
    keycode% = MENU
    DoMenu:(keycode%)
ENDP

PROC DoMenu:(k%)
    LOCAL k$(1)
    LOCAL menuops$(10)
    menuops$="xnal"      REM [S5: menuops$="enal"]
    k$=CHR$(k% AND $FF)
    IF LOC(menuops$,k$)
        @%("wahl"+k$):
    ENDIF
ENDP

```

```

PROC wahl%:
    LOCAL erg%
    USE A
    delkndva:
    erg% = ediknd%("Kunden anlegen")
    IF erg% = 0
        delkndva:
    ELSEIF erg% = 13
        A.name$ = name$ : A.tel$ = tel$
        APPEND : GIPRINT "Hinzugefuegt"
        showknd:
    ENDIF
ENDP

PROC wahl%:
    LOCAL erg%
    USE A
    erg% = ediknd%("Kundendaten aendern")
    IF erg% = 0 OR erg%=27
        delkndva:
    ELSEIF erg% = 13
        A.name$ = name$ : A.tel$ = tel$
        UPDATE : GIPRINT "Geaendert"
        LAST : showknd:
    ENDIF
ENDP

PROC ediknd%:(title$)
LOCAL erg%
DO
    dINIT title$          REM [wg. Schriftgröße am WA: dINITS title$]
    dEDIT name$,"Kundenname:"
    dEDIT tel$,"Telefonnummer:"
    dBUTTONS "Nicht speichern",27,"Speichern",13
    erg% = DIALOG
UNTIL erg% = 0 OR erg%=13
RETURN erg%
ENDP

PROC wahl%:
    LOCAL erg%
    USE A
    IF COUNT > 0
        dINIT "Kunden loeschen"
        dTEXT "", "Wirklich loeschen?"
        dBUTTONS "JA",%j,"Nein",%n
        erg% = DIALOG
        IF erg% = %n
            RETURN
        ELSEIF erg% = %j
            ERASE : POSITION 1 : showknd:
        ENDIF
    ENDIF
ENDP

PROC wahlx%:          REM [S5: PROC wahlx%:]
    USE A : CLOSE : STOP
ENDP

PROC delkndva:
    name$ = "" : tel$ = ""
ENDP

```

```

PROC showknd:
    LOCAL c%
    CLS : db2knd:
    REM [WA/SN:wg. Bildschirmgröße Font ändern: FONT 1,0]
    PRINT "  K U N D E N D A T E N"
    PRINT "  -----"
    PRINT "Kundenname: "+CHR$(9)+name$
    PRINT "  Telefon: "+CHR$(9)+tel$
    DO
        PRINT
        c% = c% + 1
        UNTIL c% = 9          REM [WA: c%=6]
    IF COUNT <> 0
        PRINT "Datensatz ";POS;"/";COUNT
    ELSE
        PRINT "Datensatz 0/0"
    ENDIF
    PRINT
    PRINT "<-" + " Zurück/Vor " + "->" + " od.Menue-Taste"
    gBORDER 0
ENDP

PROC db2knd:
    USE A : name$ = A.name$ : tel$ = A.tel$
ENDP

PROC rollen:(keycode%)
    IF COUNT > 1
        IF taste% = 259      REM Pfeil nach links
            IF POS = 1
                GIPRINT "1. Datensatz"
            ELSE
                BACK : showknd:
            ENDIF
        ELSEIF taste% = 258   REM Pfeil nach rechts
            IF POS = COUNT
                GIPRINT "Letzter Datensatz"
            ELSE
                NEXT : showknd:
            ENDIF
        ENDIF
    ENDIF
ENDP

```