lieh, daß das Kriterium des Einwilligungserfordernisses den einzigen Ansatzpunkt für eine sinnvolle Handhabung des Kopierverbots gem. § 53 IV. S.2 UrhG bietet. Letzteres führte zumindest Teile der Literatur dazu, im Abschluß eines Softwareüberlassungsvertrags durch den Berechtigten gleichzeitig die konkludente Vergabe einer Lizenz[60] im Sinne einer konkludenten Einwilligung zur Vervielfältigung des Datenverarbeitungsprogramms in den Arbeitsspeicher bei bestimmungsgemäßer Benutzung zu sehen[61]. Dieser Lösungsweg soll aber nach Ansicht der Vertreter dieser Meinung in den Fällen versagen, in denen der Berechtigte einen entgegenstehenden Willen ausdrücklich geäußert hat[62], weil dann die Möglichkeit des Rückgriffs auf eine konkludente Einwilligung verschlossen bleibt. Ob das Einwilligungserfordernis indes wirklich so problembeladen ist, soll im nachfolgenden für die unterschiedlichen im Rahmen der Benutzung von Computersoftware entstehenden Vervielfältigungen im einzelnen untersucht werden.

60) Vgl. Röttinger, a.a.O., IuR 1986, 12, 16; Lehmann, a.a.O., NJW 1988, 2419, 2420; Bunte/Graf vWestphalen, Großkommentar zum AGB-Gesetz, Band III 2.Aufl. 1985, S.40.1 Rdn.1
61) Vgl. Schricker/Loewenheim, a.a.O., § 53 Rdn. 40; Lehmann, NJW 1988, 2419, 2420
62) Vgl. Schricker/Loewenheim, a.a.O., § 53 Rdn. 40

CLIPPER - eine Alternative zu dBASE III+?

Michael König

Herr Richter a. AG Ulm H. Hoffmann hat in IuR 6/88 und 7/88[1] das relationale Datenbanksystem dBASE III+ anhand seines persönlichen Rechts-Informations-Systems dargestellt.

Vorliegender Beitrag soll keine Kritik dessen Ausführungen, sondern vielmehr eine Ergänzung hinsichtlich der Frage darstellen, ob zum Erwerb und der Verwendung von dBASE nicht bessere Alternativen existieren.

Der Verfasser hat für sein Dissertationsvorhaben ebenfalls etwas ähnliches wie ein persönliches Rechts-Informations-System entwickelt, das allerdings die banalere Bezeichnung „Literatur-Verwaltung" trägt. Nachdem die erste Version dieses Programms unter dBASE III+ entstanden war und sich dabei dessen Grenzen zeigten, erfolgte die Weiterentwicklung konsequent unter CLIPPER. Unter Berücksichtigung der nicht unerheblichen Anschaffungskosten des Programmpaketes dBASE III+ und der hohen Überlegenheit von CLIPPER sollten potentielle Käufer von dBASE erwägen, gleich „Nägel mit Köpfen zu machen" und den um maximal DM 400,— teureren CLIPPER, Version Sommer 1987, anzuschaffen.

WAS IST CLIPPER ?

CLIPPER stammt von der US-amerikanischen Firma Nantucket, die vor einigen Jahren von einer Anzahl „abtrünniger" Ashton-Tate-Mitarbeiter - bekanntlich der Herstellerin von dBASE III+ - gegründet wurde. Grund hierfür waren unterschiedlich Vorstellungen dieser Programmierer und der entscheidenden Ashton-Tate-Instanzen über die Konzeption und Features von dBASE, insbesondere, ob dBASE zu einem Compiler weiterentwickelt werden oder ein Interpreter bleiben sollte[2] also gerade die Unterschiede zwischenCLIPPER und dBASE, die nach Auffassung des Verfassers den Vorteil von CLIPPER ausmachen.

INTERPRETER - COMPILER

Die Begriffe „Compiler" und „Interpreter" stehen für unterschiedliche Verfahrensweisen bei der Umsetzung des in einer höheren Programmiersprache geschriebenen Programmes - Quellcode/-programm (sourcecode) - in den vom Computer unmittelbar ausführbaren Maschinencode - Objektcode/-programm (objeetcode). Daher können Interpreter nicht mit Compilern verglichen werden, ohne die grundsätzlichen Unterschiede zu berücksichtigen[3].

Wenn ein Quellprogramm über einen Interpreter zum Laufen gebracht werden soll, so muß zunächst der Interpreter geladen werden. Hierauf wird der Quellcode - entweder en block oder Zeile für Zeile - eingelesen. Dieser Vorgang ist mit dem Einlesen eines Textes in ein Textprogramm unmittelbar zu vergleichen, da der Quellcode auch nur einen Text darstellt - wenngleich auch nur für den verständlich, der die betreffende Programmiersprache beherrscht. Danach werden die eingelesenen Teile des Quellprogramms Zeile für Zeile analysiert, Befehl für Befehl intern in den ausführbaren Maschinencode übersetzt und sogleich ausgeführt[4]. Da auch beim mehrfachen Abarbeiten derselben Anweisungen - z. B. in einer Schleife - diese immer wieder neu übersetzt werden müssen, ist die Verarbeitungsgeschwindigkeit relativ gering. Diese wird durch im Quellcode befindliche Kommentare[5], die ebenfalls eingelesen

1) IuR 88,253ff, 310ff
2) Mantz/Holzer, Grundlagen der Programmierung mit Clipper, 1987, 10
3) vgl. Mierzowsky, Leinen los!, c't 6/86, 28
4) Daubach, Clipper, Tips und Tricks, 1987, 11
5) Kommentare im Quellcode dienen der Erläuterung des Programmaufbaus. Sie ermöglichen es dem Programmierer, auch nach längerer Zeit Programmänderungen mit vertretbarem Aufwand durchzuführen. Vgl. Mantz/ Holzer a.a.O., 19

Michael König ist Rechtsanwalt in Frankfurt.

und als solche erkannt werden müssen, weiter herabgesetzt. Ein weiterer Nachteil ist darin zu sehen, daß der Programmierer bei Vermarktung ö.ä. des Programms den Quellcode und damit das dem Programm zugrundeliegende Know how - im Gegensatz zu der Überlassung nur des Maschinencodes eines kompilierten Programmes[6] - herausgeben und damit preisgeben muß. Weiterhin eröffnet die Überlassung des Quellprogramms dem Anwender die Möglichkeit, selbständig Manipulationen an diesem vorzunehmen, was insbesondere hinsichtlich der Gewährleistung Probleme bereiten kann[7]. Nachteilig für den Vertrieb des Programms kann sich auswirken, daß jeder Anwender den Interpreter besitzen muß, denn da der Hersteller des Interpreters diesen selbst vertreiben will, hat dies - je nach Programmiersprache - eine nicht unbeträchtliche Verteuerung des Programmes zur Folge[8]. Letzlich bietet die Erstellung umfangreicher Applikationen in einer Interpreter-Sprache die hervorragende Möglichkeit, heimtückische „Zeitbomben" in das Programm einzubauen: Der Interpreter kann natürlich nur diejenigen Programmanweisungen auf ihre korrekte Syntax überprüfen, die er während des Programmlaufs übersetzt. Gerade bei einem umfangreichen Programm können einzelne Programmteile bei den Testläufen infolge der Erforderlichkeit exotischer Kombinationen von Eingabewerten unübersetzt und damit ungeprüft bleiben. Die Konsequenz ist offenkundig: Etwa vorhandene Syntaxfehler werden erst in der praktischen Anwendung festgestellt und können beim Anwender erheblichen Schaden durch Datenverluste anrichten. Die Erstellung von Progammen in einer Interpreter Sprache erfordert also einen erheblichen Aufwand beim Testen[9], was wiederum negative Auswirkungen auf den Verkaufspreis haben dürfte.

Ein Compiler übersetzt dagegen den Quellcode in einem oder mehreren Durchgängen in einen Zwischencode, der nach einer weiteren Bearbeitung - dem Linken[10] - als der ausführbare Maschinencode, also ein selbständig lauffähiges Programm, vorliegt[11]. Dabei wird jeder Teil des Quellcodes nur einmal übersetzt, so daß z.B. Programmschleifen in dieser Form erhalten bleiben und lediglich beim Programmlauf als Maschinencode wesentlich schneller abgearbeitet werden. Durch die einmalige Umwandlung in ein Objektprogramm kann der Programmierer sein Produkt weitergeben ohne befürchten zu müssen, sich durch Preisgabe seines Know-hows[12] eine weitere Vermarktung seines Produkts selbst zu vereiteln. Auch die übrigen, oben dargestellten Nachteile eines Interpreters werden vermieden: weder ist eine eigenmächtige Änderung des Programms durch den Anwender möglich möglich[13], noch benötigt der Anwender das Interpreterprogramm, denn das kompilierte Programm ist ebenso selbständig lauffähig wie z.B. der Compiler - oder Interpreter - selbst[14]. Schließlich ist auch die Gefahr unentdeckter Programmierfehler, insbesondere Syntaxfehler, geringer, da der Quellcode während des Kompilierens vollständig übersetzt und auf die korrekte Syntax überprüft wird[15].

Ein Compiler ist jedoch einem Interpreter in mancher Hinsicht auch unterlegen. Hier fallen zwei Nachteile ins Gewicht: Zum einen ist ein Compiler naturgemäß nicht in der Lage, interaktive Kommandos und Befehle zu verstehen. Es ist also z.B. nicht möglich, im Dialog den CLIPPER-Compiler anzuweisen, eine bestimmte Datenbankdatei zu eröffnen („use ..."), einen bestimmten Datensatz zu suchen („locate for ...") und diesen anzuzeigen („display"). Es müßte stattdessen ein im wesentlichen aus diesen Anweisungen bestehendes „Programm" geschrieben und kompiliert werden. Der andere Nachteil ist, daß jede auch noch so geringe Änderung des Quellcodes einen neuen Compilerlauf erfordert. Dies kann - je nach Programmumfang, Compiler und verwendetem Computersystem - mehrere Minuten oder gar Stunden dauern. Man kann also nicht „eben mal" ein Programm entwerfen oder die Wirkung bestimmter Befehle erproben[16], was sich infolge der erforderlichen Planung und Strukturierung allerdings auch auf die Qualität der Erzeugnisse positiv auswirken kann[17].

dBASE III+ - CLIPPER

dBASE stellt einen Interpreter dar, wobei dBASE zugleich auch die Bezeichnung für diese Interpreter-Sprache ist. Die Formulierung von Hoffmann, die in dieser Programmiersprache geschriebenen Programme dürften nicht mit dem eigentlichen Datenbank-Programm verwechselt werden[18], ist indes irreführend. Der Eindruck, dBASE sei ein zum selbständigen Einsatz gedachtes Datenbankprogramm, ergibt sich aus der Möglichkeit, interaktiv Anweisungen geben zu können. Es handelt sich jedoch um dieselben Anweisungen, die auch in dBASE-Programmen verwendet werden. Da es sich bei dBASE um eine spezielle Entwicklung zur Verwaltung von Daten handelt sind die verfügbaren Befehle im Vergleich mit anderen Programmiersprachen in dieser Hinsicht besonders

6) Entgegen der wohl noch überwiegenden, nichtsdestoweniger irrigen Auffassung kann die Überlassung eines Objektprogramms deshalb nicht als Überlassung eines Know-hows verstanden werden, weil das Programm lediglich unter Zuhilfenahme des Know-hows erstellt wurde, letzteres im Objektcode jedoch nicht in dem Sinne repräsentiert ist, daß es der Erwerber selbständig nutzen oder verstehen könnte. So auch Dörner/Jersch, Die Rechtsnatur der Software-Überlassungsverträge, IuR 88, 137, 141 sowie Hoeren, Softwareüberlassung an der Schnittstelle zum Urherber- und Vertragsrecht, GRUR 88, 340, 347; a.A. z.B. Bömer, Die Pflichten im Computersoftwarevertrag, 1988, 42ff, der trotz der Feststellung, daß der Erwerber des Maschinenprogrammes keinen Einblick in das Know-how erhält, die Einräumung eines Nutzungsrechtes im Rahmen eines Know-how-Lizenzvertrages bejaht, 44ff, 110f mwN; Gaul, Know-how-Schutz bei der Softwareerstellung, CR 88, 841, 842f. Es soll hier jedoch nicht weiter auf diesen Streit eingegangen werden, da zumindest bei der Überlassung des Quellcodes auch nach der hier vertretenen Auffassung ein programmiertechnisches Know-how durch die mit der Herausgabe des Quellcodes verbundene, zwangsweise Offenlegung der Programmalgorithmen mitgeteilt wird.

7) vgl. hierzu Bömer a.a.O., 110f

8) Zu diesen Nachteilen vgl. Daubach a.a.O., 12ff; s. auch Mantz/Holzer a.a.O., 10.

9) vgl hierzu Daubach a.a.O., 15

10) Die technischen Einzelheiten interessieren hier nicht weiter; es genügt zu wissen, daß dieser sog. Linker idR zum Lieferumfang des Compilers gehört und auch schon oft als Teil der betriebssystemnahen Software vorhanden ist.

11) Bei dieser knappen Darstellung bleiben Zwischenformen, wie z.B. das Run-Time-Modul von Ashton Tate oder der Pseudocompiler Foxbase+, unberücksichtigt. Vgl. hierzu Daubach a.a.O., 13; Dahlmann, Zweimal mehr als das Vorbild, Chip 10/88, 264f; Mierzowsky, Familienzuwachs, c't 7/87, 68ff; ders., Leinen los!, c't 6/86, 28.

12) vgl. FN 6

13) Unberücksichtigt bleibt die Möglichkeit, unmittelbar im Maschinencode Änderungen vorzunehmen, da dies nur durch erfahrene Assembler Programmierer und auch durch diese nur sehr eingeschränkt erfolgen kann.

14) vgl. Daubach a.a.O., 14f

15) vgl. FN 9

16) vgl. Mierzowsky a.a.O.

17) vgl. Daubach a.a.O., 16

18) Hoffmann a.a.O, 256. Ähnlich auch Simpson,Arbeiten mit dBASE III, 1986, 9, der dBASE III als ein Datenbank-Verwaltungssystem bezeichnet.

mächtig[19]. Nichtsdestoweniger kann auch in einer anderen Interpreter Sprache - z. B. BASIC - interaktiv eine Datenbankverwaltung durchgeführt werden, wenngleich dies auch eine nahezu unzumutbare Tortur darstellte[20]. Dennoch würde niemand BASIC als ein Datenbankprogramm bezeichnen. Der Grund, warum dBASE als ein schwierig handzuhabendes Programm ohne anwenderfreundliche Benutzeroberfläche gilt[21], ist gerade in der Meinung zu suchen, es handele sich um ein zur unmittelbaren Verwaltung von Daten bestimmtes Endanwenderprogramm. Wie sich aus dem Beispiel Hoffmanns[22] ergibt, erfordert die interaktive Verwaltung von Datenbanken schon bei an sich banalen Suchanweisungen sehr komplizierte und fehlerträchtige Befehlsungetüme, die zu allem Überfluß bei jeder weiteren Abfrage noch einmal in gleicher oder ähnlicher Form eingegeben werden müssen. Versteht man dBASE dagegen als eine spezielle Programmiersprache zur Entwicklung von Datenbank-Verwaltungsprogrammen, so stellt sich deren Programmierung mit dBASE als im Vergleich mit z. B. BASIC oder PASCAL ausgesprochen einfach und unkompliziert dar. Hoffmann führt jedoch zu Recht aus, daß dBASE durch die neue Benutzeroberfläche ASSIST wesentlich einfacher zu bedienen ist - der Anwender muß sich nicht mehr die Befehle merken, sondern wählt sie aus den Menüs aus[23].

Weil dBASE einen Interpreter darstellt wird der Quellcode erst während der Ausführung in die Maschinensprache übersetzt. Die Nachteile sind wie oben dargestellt:

- langsamer Programmablauf
- umständliche Bedienung, da zunächst erst dBASE geladen werden muß
- Geheimhaltung des Quellcodes ist nicht möglich
- Anwender muß den Interpreter besitzen
- Anwender kann das Programm eigenmächtig ändern
- Aufwendige Fehlersuche

dBASE-spezifisch kommt noch hinzu, daß der Texteditor zur Erstellung des Programms und zur Bearbeitung der MEMO-Felder lediglich eine Kapazität von ca. 4000 Zeichen hat. Die Möglichkeit, diesen per Definition in einem Systemfile durch ein beliebiges Textsystem zu ersetzen, stellt nur eine notdürftige Improvisation dar, weil das entsprechende Textprogramm zunächst erst geladen werden muß.

CLIPPER stellt dagegen einen Compiler dar, also ein Programm, das den Quellcode in die Maschinensprache übersetzt. Während des Compilerlaufes wird das Programm auf fehlerhafte Befehlssyntax geprüft und entsprechende Fehler angezeigt. Dies ist z. B. bei der Programmierung verschachtelter Schleifen von Vorteil, da hier vergessene „end"-Befehle sofort angezeigt werden - bei dBASE führt ein solcher „Codierfehler" zu merkwürdigem Fehlverhalten, das oft nur nach stundenlangem Suchen aufgeklärt werden kann. Während sich CLIPPER früher - 1985 - im wesentlichen darauf beschränkte, die Compilierung von dBASE-Programmen zu ermöglichen, stellt es mittlerweile eine im Verhältnis zu dBASE sehr umfangreiche und mächtige Programmiersprache speziell zur Entwicklung relationaler Datenbanksysteme dar. Als Nachteil dieser Spezialisierung fehlen CLIPPER, insbesondere jedoch dBASE, eine Reihe von Befehlen, die in anderen Programmiersprachen im Verhältnis zu dBASE/CLIPPER besondere Möglichkeiten bieten - diesen Mangel verspürt schmerzhaft selbstverständlich nur derjenige Anwender, der bereits in anderen, universellen Programmiersprachen entsprechende Programme entwickelt hat. Bei CLIPPER kann dieser Mangel jedoch durch die Erstellung und einfache Integration von ergänzenden Funktionen in C und ASSEMBLER behoben werden.

Die Vorteile von in CLIPPER geschriebenen und compilierten Programmen sind also in concreto:

- Schneller Programmablauf
- Einfacher Programmaufruf
- Umfangreicher und mächtiger Befehlssatz
- Geschützter Quellcode, da nur der Objektcode herausgegeben zu werden braucht
- Programm selbständig lauffähig
- „Automatische" Überprüfung der Befehlssyntax

Dabei ist CLIPPER insofern kompatibel zu dBASE III+, als sich nach relativ gerinfügigen Änderungen auch unter dBASE erstellte Programme compilieren lassen. Änderungen sind im wesentlichen bei der Verwendung der interaktiven Befehle von dBASE wie „append", „edit", „create", „browse" u.ä. erforderlich, da ein Compiler selbstverständlich keine interaktive Befehle kennt. Diese Änderungen sind jedoch relativ leicht durchzuführen, wenngleich auch das Handbuch hierbei keine allzugroße Hilfe darstellt. Es existieren jedoch einige, wenn auch nicht billige, Publikationen, die den Umstieg bzw. Einstieg erleichtern[24]. Allerdings sind Quellcodes, die den umfangreicheren Sprachumfang von CLIPPER auch nur zum Teil ausschöpfen, unter dem Interpreter dBASE nicht lauffähig, da dieser die CLIPPER-spezifischen Befehle nicht kennt.

Die oben aufgeführten prinzipiellen Nachteile eines Compilers - keine Möglichkeit des unmittelbareren, sofortigen Laufs des Quellcodes sowie der interaktiven Befehlseingabe - entfallen bei der neuesten Version des CLIPPER annähernd vollständig.

Während früher das Compilieren und Linken zu einer zeitraubenden Angelegenheit werden konnte, ist bei CLIPPER Sommer 1987 eine beachtliche Geschwindigkeitssteigerung festzustellen. Unter Verwendung des Turbo-Linkers TLINK von Borland anstelle des mitgelieferten Overlay-Linkers PLINK86+ oder des MS-Linkers LINK[25] lassen sich sogar recht umfangreiche Applikationen in weniger als einer Minute compilieren und linken[26]. Selbstverständlich muß der Quellcode zuvor mit einem beliebigen Texteditor als ASCII-File CLIPPER zur Verfügung gestellt worden sein. Dies ist jedoch insgesamt nicht umständlicher als das Einbinden eines Textsystems in dBASE.

19) vgl. Mantz/Holzer a.a.O., 9; Hoffmann a.a.O., 312f
20) Währnd unter dBASE mit einem „locate for ...-Befehl ein bestimmter Datensatz automatisch eingelesen werden kann, muß unter BASIC manuell solange jeder einzelne Datensatz eingelesen und inhaltlich ausgewertet werden, bis der gewünschte Datensatz gefunden ist.
21) Hoffmann a.a.O., 256
22) Hoffmann a.a.O., 313
23) Hoffmann a.a.O., 256
24) z. B. Daubach a.a.O.; Mantz/Holzer a.a.O.
25) Bis vor kurzem war dieser Linker noch im Betriebssystem MS-DOS enthalten.
26) Auf dem 10-MHz-XT mit V20 und 30-MB-Festplatte des Verfassers benötigten bei einem Quellcode von ca. 60 KByte und ca. 2100 Programmzeilen das Compilieren etwa 40 sec. und das Linken ca. 15 sec.

In der neuen Version bietet CLIPPER das Utility DBU, das ähnlich wie die ASSISTent-Benutzeroberfläche von dBASE III+ - die Errichtung und Bearbeitung von Datenbanken erlaubt. DBU liegt als Quellcode vor, der auch ganz oder zum Teil in eigene Applikationen eingebaut werden kann. DBU kann auch allein compiliert und gelinkt werden und stellt danach eine brauchbare Alternative zur Entwicklung eines speziellen Datenbankprogrammes dar, sofern man auf den Komfort einer auf die individuellen Bedürfnisse zugeschnittenen Entwicklung keinen Wert legt. Dennoch ermöglicht DBU nicht das interaktive Arbeiten mit CLIPPER, denn es stellt selbst nur ein in CLIPPER geschriebenes Programm dar.

Einzelne CLIPPER-spezifische Befehle

Dieser Beitrag versteht sich nicht als eine Einführung in CLIPPER - hierfür ist dessen Befehls- und Funktionsumfang viel zu umfangreich. Es sollen lediglich schlaglichtartig einige der wesentlichen Vorteile gegenüber dBASE dargestellt werden. Da der Einsatz von CLIPPER das Erlernen dieser Programmiersprache und das Erstellen eigener Programme voraussetzt - eine interaktive Datenverarbeitung, wie in dem Beitrag von Hoffmann in den Vordergrund gestellt, ist mit Ausnahme der Verwendung des vorerwähnten Hilfsprogrammes DBU nicht möglich - ist die Bedeutung der Vorteile ohne Beschäftigung mit der Programmiersprache CLIPPER nicht unbedingt sofort erkennbar. Einfache Datenbankprogramme können jedoch schon in relativ geringer Zeit erstellt werden, wenngleich sichere und mit kommerziellen Programmen vergleichbare Eigenentwicklungen doch einen erheblichen Zeit- und Programmieraufwand erfordern. Hierbei wird für die eigentlichen Datenbankfunktionen infolge der mächtigen Befehle wesentlich weniger Zeit benötigt als für die Erstellung ansprechender Ein- und Ausgabemasken sowie die Verhinderung von Bedienungs- und Eingabefehlern. Aus diesem Grund hat der Verfasser bei der Erstellung seines Programms auf aufwendige Bildschirmmasken, wie sie z. B. das Programm Hoffmanns zieren[27], verzichtet.

MEMO-Felder[28]

MEMO-Felder dürfen bis 64 KByte groß sein. Dies entspricht etwa 32 DIN-A-4-Seiten - genug für jeden Langtext. Der Inhalt des MEMO-Feldes kann ohne Schwierigkeiten zusammen mit den übrigen Daten angezeigt und editiert werden (vgl. Abb. 1). Dabei kann die Größe und Lokalisierung des MEMO Feld-Fensters frei bestimmt werden. In CLIPPER ist hierfür ein relativ einfacher Texteditor integriert, der mit Hilfe einiger Programmiertricks unter Ausnutzung CLIPPER-spezifischer Befehle z. B. um eine einfache aber wirkungsvolle Silbentrennung erweitert werden kann. Das MEMO-Feld kann - im Gegensatz zu dBASE - wie eine gewöhnliche String-Variable behandelt werden. Dadurch ist also z. B. eine Volltext-Recherche ohne weiteres möglich und auch infolge der Schnelligkeit der compilierten Programme praktisch verwendbar. So benötigt das vom Verfasser entwickelte Programm für das Durchsuchen von ca. 300.000 KByte Langtext, der auf ca. 370 Datensätze verteilt ist, nach einem Begriff weniger als 30 Sekunden [29]. Zum Vergleich: Das ebenfalls sequentielle Durchsuchen des Stichwortfeldes (ca. 370 x 65 Byte) dauert weniger als 5 Sekunden. Infolge dieser kurzen Suchzeiten hat der Verfasser in seiner Programmentwickung auf das Anlegen mehrerer, indizierter Stichwortfelder verzichtet und sich mit einem einzigen Feld für die Aufnahme sämtlicher Stichworte begnügt (Abb. 1). Diese Suchzeiten lassen sich bei entsprechender otpimierender Programmierung sicherlich noch verkürzen.

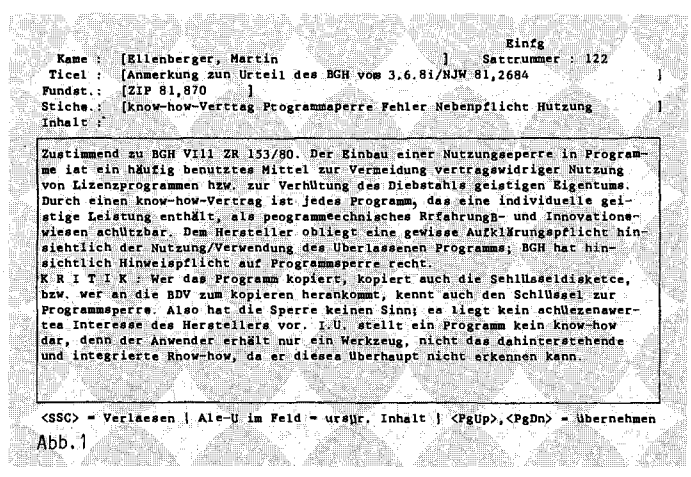

Abb. 1

Weiterhin existieren eine Vielzahl von Spezialfunktionen zur Verarbeitung von MEMO-Feldern, die z. B. auch das Speichern der MEMO-Felder als ASCII-Files oder das Einlesen beliebiger Textdateien in MEMO-Felder ermöglichen. Dadurch kann das von Hoffmann angesprochene Manko der Nichttransportabilität der MEMO-Felder[30] ohne großen Programmieraufwand beseitigt werden. Durch neue String-Funktionen wird außerdem z. B. eine Umwandlung des ASCII-Textes in das WordStar-Format ermöglicht.

Browse

CLIPPER kennt den interaktiven dBASE-Befehl „browse" zum schnellen Scannen der Datenbank nicht. Als überreichliche Kompensation ist dafür die Funktion „dbedit()" verfügbar, die ungleich mehr Möglichkeiten bietet als „browse". So können sämtliche Felder der Datenbank in beliebigem Format zugleich angezeigt werden, wobei verschiedene Optionen weitere Einzelheiten der Darstellung bestimmen (vgl. Abb. 2). Diese Funktion stellt auch das Rückgrat der oben erwähnten Utility DBU dar.

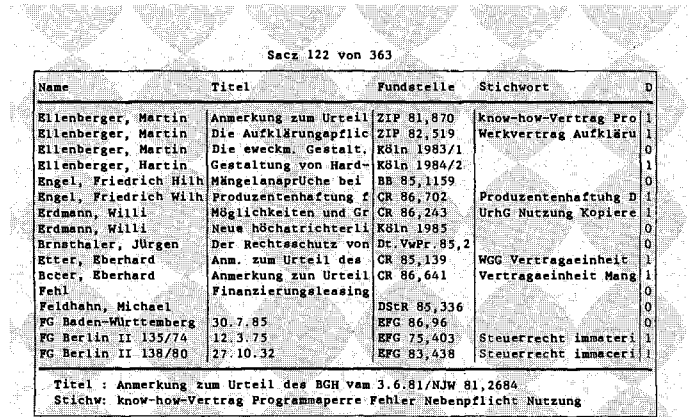

Abb. 2

27) Hoffmann a.a.O, 254, 257
28) Siehe zunächst die Erläuterungen Hoffmanns a.a.O., 258
29) Ermittelt auf einem 10-MHz-XT mit V20-CPU und Festplatte mit 60ms Zugriffszeit.
30) Hoffmann a.a.O., 315f
31) Hoffmann a.a.O., 316

Windows

Seit einiger Zeit ist es Mode geworden, Dateneingaben und -ausgaben in sog. Windows vorzunehmen (vgl. Abb. 3). Ohne spezielle Funktionen ist dies nicht oder nur mit sehr großem Programmieraufwand möglich. CLIPPER enthält nun zwei Funktionen, durch deren Einsatz der Inhalt exakt spezifizierter Abschnitte des Bildschirms „gespeichert" und wiederhergestellt werden kann. Zur Erzeugung eines Windows bedarf es dadurch nur mehr dreier Befehle.

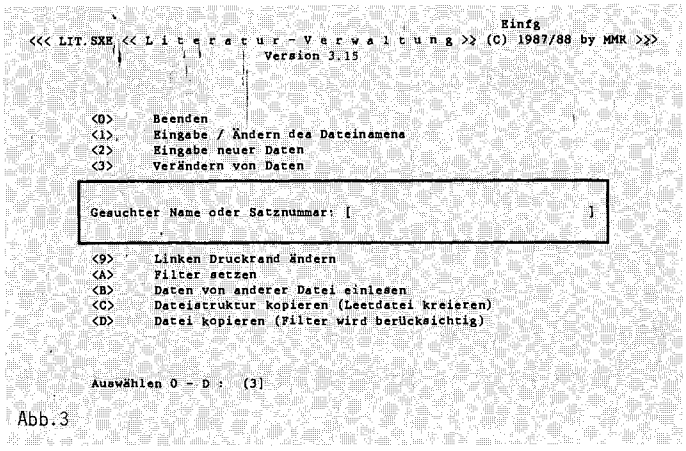

Abb. 3

Index-Dateien

Bislang waren im Gegensatz zu den Daten-Dateien die Index Dateien von CLIPPER zu denen von dBASE nicht kompatibel, jedoch in der Verarbeitung wesentlich schneller. Dies stellte allerdings keinen Nachteil dar, da Index-Dateien zu jeder Zeit ohne Aufwand vom Programm generiert werden können. In der neuen Version können nun auch nach Wahl dBASE-kompatible Index-Dateien verwendet werden - allerdings unter Verlust des CLIPPER-typischen Geschwindigkeitsvorteils.

§ - Zeichen

Für Juristen besonders wichtig ist, daß CLIPPER im Gegensatz zu dBASE[31] sowohl das §-Zeichen als auch sämtliche Umlaute in sämtlichen Datenbankfeldern einschließlich des MEMO-Feldes akzeptiert und darstellt.

Eigene Funktionen

Versierte Programmierer können eigene Routinen in anderen Programmiersprachen schreiben - C, ASSEMBLER - und als Funktionen in die Programme einbinden. Dies ist aus dem Grunde vorteilhaft, weil CLIPPER trotz aller Mächtigkeit und Schnelligkeit der Programmiersprache selbstverständlich nicht für jede denkbare Anforderung eine passende Funktion bereitstellt. Diese kann vom Anwender durch eine selbst zu entwickelnde Routine ersetzt und beim Linken eingebunden werden.

Debugger

In CLIPPER-Programme kann ein leistungsfähiger Debugger zur Fehlersuche „eingebaut" werden, dessen Aufruf aus dem laufenden Programm heraus möglich ist.

Fehlerfunktionen

Während Fehler beim Programmablauf von dBASE im Ergebnis zum Programmabsturz - und ggfs. Datenverlust - führen, kann bei CLIPPER ein individuelles Fehlersystem integriert werden, das eine Rückkehr in den normalen Programmlauf ermöglicht.

Hilfefunktion

Jedes Programm kann mit einer kontext-abhängigen integrierten Hilfefunktion versehen werden, die auch unerfahrenen Anwendern eine sachgerechte Bedienung ermöglicht (vgl. Abb. 4). Diese Hilfefunktion wird - wie es mittlerweile bei vielen professionellen Programmen zum Standard geworden ist - mit der Funktionstaste F1 aufgerufen.

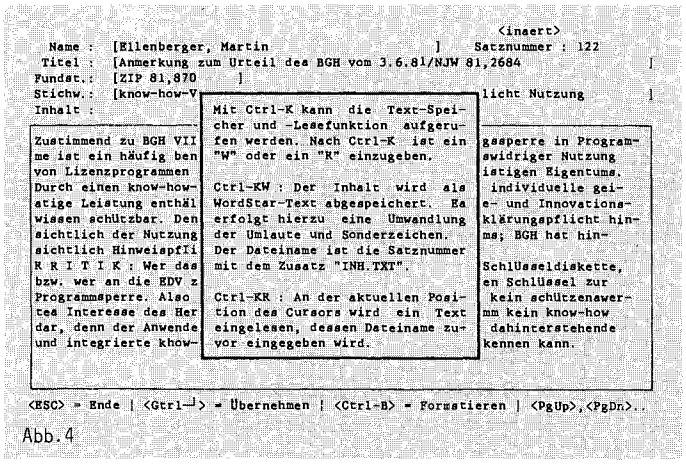

Abb. 4

DOS-Datei-Funktionen

Ein großer Vorteil von CLIPPER und dBASE ist, daß man sich nicht um die Spezifizierung der Dateiparameter zu kümmern braucht, wie es bei anderen Programmiersprachen der Fall ist. Andererseits wäre es gelegentlich hilfreich, wenn man Dateien selektiv byteweise einlesen oder schreiben könnte. Diese niedrigstufigen DOS-Funktionen bietet CLIPPER in der neuesten Version.

Array-Variable

Array-Variable sind Variable, die zu einem Array desselben Namens gehören und durch einen Index gekennzeichnet sind. Wenn z. B. das Array mit „declare test10" definiert ist, können die 10 zu diesem Array gehörenden Variablen mit den Bezeichnungen test1, test2, test3, ... test9, test10 angesprochen werden. Da als Index eine andere Variable fungieren kann, ist der Vorteil offensichtlich. In der neuesten Version von CLIPPER kann das Array 4096 Variable umfassen, zwar weniger, als in anderen Programmiersprachen, jedoch besser als überhaupt nicht vorhanden.

Keyboard-Funktionen

Durch eine spezielle Funktion kann das Drücken einer beliebigen Taste des Keyboards simuliert werden - der betreffende Tastenwert wird in den Tastaturpuffer übertragen. Bei der nächsten Abfrage der Tastatur - z. B. einer Menüauswahl - wird dieser simulierte Tastendruck ausgewertet. Andere Funk-

tionen stellen fest, welche Taste zuletzt betätigt wurde oder welche als nächste betätigt wird. Selbstverständlich hat CLIPPER keine hellseherischen Fähigkeiten, sondern schaut nur nach, welcher Tastenwert im Tastaturpuffer bereitliegt. Durch diese Funktionen ist es möglich, Teile des Programmablaufes bei bestimmten Bedingungen zu automatisieren, ohne daß der Anwender weitere Anweisungen zu geben hätte. Dies ermöglicht eine Erhöhung des Bedienungskomforts und der Einheitlichkeit der Programmbedienung, wie es unter dBASE - sofern überhaupt - nur unter ganz erheblichem Programmieraufwand und damit verbundener Verlangsamung des Programmablaufs erreicht werden könnte.

Es existieren über die vorstehende Darstellung hinaus eine Vielzahl weiterer Funktionen und Befehle, die es ermöglichen, Programme effektiver und professioneller zu gestalten, ohne daß eine auch nur annähernde Aufzählung und Beschreibung hier möglich wäre. Zuletzt sollte vielleicht darauf hingewiesen werden, daß der Verfasser der Firma Nantucket in keiner Weise verpflichtet ist. Es gibt auf dem Markt noch andere dBASE-ähnliche Compiler[32]. Der Verfasser kennt jedoch aus eigener Erfahrung nur CLIPPER, so daß ein Vergleich mit anderen Compilern gerade nicht vorgenommen wird.

32) vgl. Mierzowsky, Familienzuwachs, c't 7/87, 68ff; vgl auch FN 11

Betriebssystem - einheitliches immaterielles Wirtschaftsgut - nicht rechtskräftiges Urteil des Finanzgerichts Hamburg vom 3. 8. 1988 Az. II 8/86

1. Vorbemerkung:

Das nachfolgend gekürzt wiedergegebene Urteil des Finanzgerichts Hamburg, das noch Gegenstand einer Revisionsentscheidung des Bundesfinanzhofs sein wird, bestätigt bereits bisher in der Literatur überwiegend vertretene Auffassungen. Das Urteil, das zu § 19 des Berlin-Förderungsgesetzes (Investitionszulage) ergangen ist, befaßt sich mit zwei in der Literatur häufig behandelten Fragen:

— der Frage, ob Betriebs-Software ein eigenes, selbständiges Wirtschaftsgut ist,

und

— ob sie ein immaterielles Wirtschaftsgut darstellt.

2. Leitsatz (nicht amtlich, zusammengefaßt)

a) System-Software stellt ein eigenes Wirtschaftsgut dar und ist nicht Bestandteil der Hardware.

b) System-Software ist steuerlich als immaterielles Wirtschaftsgut zu behandeln.

3. Sachverhalt

Der Kläger hatte als Anwender gegen Einmalzahlung das Recht der Nutzung eines Betriebssystems erworben. Für das Nutzungsrecht wurden allerdings gesonderte Lizenzverträge geschlossen. Die Software wurde, soweit ersichtlich, vom Hersteller der Hardware erworben. Der Softwareverkäufer hat unter Hinweis auf sein Urheberrecht dem Kläger lediglich ein gegenständlich beschränktes, nicht übertragbares und nicht ausschließliches Nutzungsrecht an der System-Software einschließlich dem zugehörigen Datenträger eingeräumt.

Der Kläger hatte für diese System-Software Investitionszulage nach dem Berlin-Förderungsgesetz beantragt, die er nur erhalten kann, wenn Software als sogenanntes materielles Wirtschaftsgut, d. h. ein bewegliches, körperliches Wirtschaftsgut anzusehen wäre. Die beklagte Finanzbehörde hat dagegen die Auffassung vertreten, daß System-Software (= Betriebssystem oder System-Steuerungsprogramm) genau wie sogenannte Anwenderprogramme ein immaterielles Wirtschaftsgut darstellt und damit nicht investitionszulagebegünstigt nach dem Berlin-Förderungsgesetz ist.

4. Betriebssystem eigenständiges Wirtschaftsgut

Zunächst mußte das Finanzgericht sich mit der Frage auseinandersetzen, ob ein Betriebssystem überhaupt ein bilanzrechtlich selbständiges, eigenes Wirtschaftsgut ist, oder ob es sich insoweit um einen rechtlich unselbständigen Bestandteil der Hardware handelt. In letzterem Fall wäre der Klage stattzugeben gewesen. Das Finanzgericht setzt sich zu dieser Frage ausdrücklich mit der neueren steuerrechtlichen Literatur, insbesondere mit den Kommentarmeinungen, auseinander. Es vergleicht dazu das Betriebsprogramm mit klassischen Anschaffungsnebenkosten, z. B. den Kosten für Transport oder Fundamentierung einer Maschine, Montagekosten, Anschlußkosten etc. Das Finanzgericht weist darauf hin, daß zu derartigen Nebenkosten, die jedoch nur zusammen mit dem Hauptgegenstand steuerlich zu bilanzieren sind, durchaus auch immaterielle Wirtschaftsgüter gehören können, z. B. Kosten einer Bauplanung, Konzessionskosten etc. Für die in Frage stehenden Betriebssysteme sieht das Finanzgericht dann aber erhebliche Unterschiede darin, daß die Betriebssoftware tatsächlich auch unabhängig von der gekauften Hardware auf anderer Hardware lauffähig ist, daß sie von verschiedenen Anbietern erworben werden kann. Das Finanzgericht hat ausdrücklich darauf hingewiesen, daß, obwohl das Betriebssystem nur für ein bestimmtes Gerät rechtlich aufgrund der Vertragsgestaltung genutzt werden durfte, der Verkäufer der Software ausdrücklich angeboten hatte, gegen Zahlung einer Zusatzgebühr sie auch auf anderen Geräten einzusetzen. Aus diesen Gründen vertritt das Finanzgericht die Auffassung, daß das Wirtschaftsgut Betriebssystem sehr wohl auch ohne das an-