DAS ELEKTRONISCHE MAGAZIN 4/86

# INPUT 64

Infos · News · Programme · Unterhaltung · Tips

LISP 64:
## Künstliche Intelligenz auf dem 64er mit dem INPUT64-LISP-Interpreter

LIFE:
## Simulation von Wachstumsprozessen

IC-Tabelle:
## Integrierte Schaltkreise werden durchsichtig

64er Tips
Mathe mit Nico

Dokumentation und Bedienungshinweise

INPUT-ein Magazin aus dem Verlag Heinz Heise GmbH, Postfach 61 04 07, 3000 Hannover 61


## Ab 4/85 auch auf Diskette-

direkt vom Heise-Verlag, INPUT-Vertrieb,
Postfach 610407, 3000 Hannover 61

für 19.80 DM inkl. Versandkosten + MwSt.-

Nur gegen V-Scheck!

## Das Geschenk:

**INPUT 64 V . Disk im Sixpack**

Die Ausgaben 4/85 bis 9/85 der Disketten-Version bekommen Sie ab sofort zum Paketpreis von 90 DM.

Jetzt bestellen, **24.80 DM** sparen!

Direkt beim Verlag:
(Inclusive Porto und Verpackung) **Nur gegen V-Scheck!**

**Verlag Heinz Heise GmbH · Postfach 610407 · 3000 Hannover 61**

## Kurs komplett

**Jetzt als Sampler:**
**Die Serie BITS & BYTES IM VIDEO-CHIP**

Alle Folgen des Kurses auf Kassette und Diskette. Eine grundlegende Einführung in die Programmierung des Video-Chip, mit Exkursen in die Binärarithmetik, Programmiertips und so weiter.

Überarbeitet und um einen Teil zur Multicolor-Grafik erweitert.

Kassette 17.80 DM
(mit SuperTape-Lader und Sicherheitskopie auf der Rückseite)

Diskette 24.80 DM

Direkt beim Versand: (inclusive Porto und Verpackung)

Nur gegen V-Scheck!

**Verlag Heinz Heise GmbH · Postfach 610407 · 3000 Hannover 61**

# INHALTSVERZEICHNIS

INPUT 64

4/86

## *Liebe 64er-Besitzer(in)!*

*Konventionen sind dazu da, gebrochen zu werden. Und außerdem erfordern große Dinge viel Raum. Trivial? Finden wir auch. Wir wollten Ihnen auch nur sagen, daß das Vorwort diesmal aus Platzgründen (LISP!) entfällt.*

*Viel Spaß mit INPUT 64*

### Auf einen Blick: INPUT 64 - Betriebssystembefehle

| | |
|---|---|
| Titel abkürzen | CTRL und Q |
| Hilfsseite aufrufen | CTRL und H |
| Inhaltsverzeichnis aufrufen | CTRL und I |
| Farbe für Bildschirm-Hintergrund ändern | CTRL und F |
| Rahmenfarbe ändern | CTRL und R |
| Bildschirmausdruck | CTRL und B |
| Programm sichern | CTRL und S |

Laden von Kassette mit LOAD oder SHIFT + RUN/STOP
Laden von Diskette mit LOAD "INPUT*",8,1
Ausführliche Bedienungshinweise finden Sie auf Seite 29

## Leser fragen. . .

### INPUT BASIC, AUTO und Arrays

*Wenn unter INPUT-BASIC (Ausgabe 1/86) in folgender Schleife das Text-Array N$(i) auf den Bildschirm gebracht werden soll, funktioniert die Übergabe des dritten Parameters im Text-Befehl (Schriftabstand) nicht mehr.*

```
140 FOR I = 1 TO 2
150 TEXT0,P(I),8,N$(I)
160 NEXT I
```

*Ein Fehler von INPUT-BASIC oder von mir?:*
*(L.Kutscha, Frankfurt)*

Leider ein Fehler von INPUT-BASIC. (Es gibt eben nur zu 99,5 Prozent fehlerfreie Software . . .) Wird dem Text-Befehl als auszugebender String eine indizierte Variable übergeben, wird die Schrift zu breit. Man kann diesen Effekt durch folgende Änderung in Zeile 150 leicht umgehen:

```
150 ZW$ = N$(I):TEXT0,P(I),8,N$(I)
```

Die indizierte Variable N$(I) wird vor dem Ausdruck einem normalen String zugewiesen, der dann fehlerfrei auf dem Bildschirm erscheint. Apropos 99,5-prozentige Fehlerfreiheit: der AUTO-Befehl arbeitet nur bis zur Zeilennummer 32767 korrekt! (d. Red.)

### Redaktion fragt

*Betrifft Flugsimulator JETFLIGHT (Ausgabe 10/85): Wie schnell schaffen Sie es eigentlich von Hannover nach Frankfurt? Schreiben Sie uns doch kurz 'ne Postkarte, der Schnellste wird durch namentliche Nennung hier im Heft geehrt. Bitte nicht mogeln!: (d. Red.)*

### Textchaos bei MiniGraphik

*In der Befehlserklärung zu dem Programm MiniGraphik (Ausgabe 1/86) hat der Druckfehlerteufel massiv zugeschlagen. Die einzelnen Befehle und die entsprechenden Erläuterungen wurden auseinandergerissen. Hier noch einmal der Text in der richtigen Reihenfolge: (d. Red.)*

Bereich festlegen: SYS51218,n (SYS2642,n)
Dieser Befehl muß (!) als erster eingegeben werden. n

definiert den Anfang des Speicherbereich für die acht Sprites (also die Sprite-Zeiger), und zwar entsprechend der aus der Programmierung der Zeiger 2040 bis 2047 bekannten Rechenvorschrift: Adresse gleich Pointer-Inhalt mal 64, hier also n*64. In der Version am BASIC-Anfang sind dafür 512 Byte freigehalten worden, durch n = 33 wird genau dieser Adressbereich belegt. In der Version ab $C800 sollte der VIC-Adressbereich auf Anfangsadresse $8000 (32768 dez) gelegt werden. (Siehe Listing 1, Zeilen 1 und 2). n = 0 belegt dann die Adressen $8000 folgende mit Sprite-Definitionen.

Einschalten: SYS51200,x,y,g (SYS2624,x,y,g)
Schaltet die Mini-Graphik mit den Koordinaten x und y ein. Der Punkt 0/0 ist oben links. Beim Einschalten muß die Graphik vollständig auf den Bildschirm passen, ansonsten kommt es zu einem 'Illegal Quantity Error'. g bestimmt die Größe. g = 0: Normal-Darstellung; g = 1: Maxi-Darstellung.

Ausschalten: SYS51203 (SYS2627)
Schaltet die Graphik aus, ohne andere Parameter zu verändern.

Löschen: SYS51206 (SYS2630)
Löscht den Inhalt von Sprite-Blocks und Mini-Graphik.

Farbe setzen: SYS51209,c (SYS2633,c)
Setzt die Farbe c für jeden gesetzten Punkt in der Graphik.

Invertieren: SYS51212 (SYS2636)
Invertiert das Graphik-Bild.

Punkt setzen: SYS51215,x,y,m (SYS2639,x,y,m)
Setzt einen Punkt in das Graphikfeld mit den Koordinaten x und y im Mode m. m = 0: Punkt löschen; m = 1: Punkt setzen; m = 2: Punkt invertieren. Der Nullpunkt ist oben links. x-Maximum: 95; y-Maximum: 41.

Text einblenden: SYS51221,x,y,m,ab, text (SYS2645, x,y,m,ab, text )
Setzt den Text text an die Koordinaten x/y im Modus m (siehe oben) mit Zeilenabstand ab. x/y bezieht sich auf die linkere, obere Ecke des ersten Zeichens in text , m beträgt normalerweise 8.

Priorität setzen: SYS51224,p (SYS2648,p)
Legt Graphik/Text-Priorität fest. p = 0: Graphik hinter Schrift; p = 1: Graphik vor Schrift.

Drucken: SYS51227 (SYS2651)
Erzeugt eine Hardcopy der Graphik. Zur Druckeranpassung siehe unten.

Bewegen: SYS51230,r,s (SYS2654,r,s)
Bewegt die Graphik in Richtung r um s Schritte. Für r gilt die Zuordnung:

| | |
|---|---|
| 0 = links/oben | 1 = oben |
| 2 = rechts/oben | 3 = rechts |
| 4 = rechts/unten | 5 = unten |
| 6 = links/unten | 7 = links. |

Joystick: SYS51233,g (SYS2657,g)
Die Graphik folgt den Bewegungen eines Joy-Sticks in Port 2. g gibt die Geschwindigkeit an, zulässig sind Werte von 1 bis 255. (1 = schnell) Betätigen des Feuerknopfs beendet den Befehl. Falls Sie keinen Joy-Stick besitzen: die Tasten a,z,. und / entsprechen den Richtungen aufwärts, abwärts, links und rechts.

## Rudi überlistet

*... wollen Sie nicht für das Spiel RUDI DIE RATTE Spiele-Pokes veröffentlichen? Ich habe POKE 3731,99 und 3766,99 herausgefunden. Damit werden die Anzahl der Figuren hochgesetzt.*
*(tel. Anfrage)*

Übung macht den Meister. Ein guter Tip, um sich in die Spielbedienung einzugewöhnen. Geht natürlich nur, wenn Sie das Spiel vorher auf eigenem Datenträger abgespeichert haben. Sollten Ihnen für andere Spiele unseres Magazins ähnliche Tips einfallen, geben wir diese gerne an unsere Leser weiter. (d. Red.)

## Bezugsquelle - Turboass

*Ich habe Ihren Testbericht über Turboass gelesen und würde diesen gerne bestellen. Wer vertreibt diesen Assembler?* *(Leserbrief)*

Turboass wird vertrieben von:
Omikron Software
Erlachstr. 15
7534 Birkenfeld 2
Tel: 0 70 82/53 86
Zum Preis von 139.-DM erhalten Sie Handbuch, Diskette und Dongle. (d. Red.)

## Minidat maximal

*Ich habe das Programm MiniDat auf Diskette überspielt. Es funktioniert nur teilweise! Folgende Befehle gehen nicht:*

*f - finden Zeichenkette*

*n - neuen Namen festlegen*

*s - saven auf Diskette geht nur das erste Mal. Nach Erweiterung der Datei kann man nicht mehr saven.*

*s $,dv - Saven unter Namen auf Gerät dv*

*@ Com - Disketten-Befehle*

*f1/f3 - Blättern zurück*

*Frage: Kann man MiniDat auch Listen?*
*(Herr Paudler)*

*...Was bewirken die Markierungen für Zeilen-/Seitenende? Zum Beispiel im Magazin 000/013.*
*(Herr Wagner)*

Zugegeben, die Bedienungsanleitung im Beiheft ist ein wenig zu kurz ausgefallen, doch ganz so schlimm, wie es hier den Anschein hat, ist es nun doch nicht. Das größte Mißverständnis verursachte wohl die Bezeichnung '$'. Gemeint ist

hiermit eine Abkürzung der Commodore-Bezeichnung für String-Variablen; also '$' entspricht "text ". So funktionieren auch die Befehle 'f', 'n', 's':

f - Beispiel: f "test " findet alle Seiten, auf denen die Zeichenkette 'test' vorkommt. Weitersuchen mit SPACE-Taste.

n - Beispiel: n "datei01 " nennt die MiniDat-Datei, an der Sie gerade arbeiten, in 'datei01' um.

s - Beispiel: s "datei02 ",1 speichert die Datei unter dem Namen 'datei02' auf Kassette. Mit . . .,7 wird im SuperTape-Format abgespeichert, wenn SuperTape vorher geladen und gestartet wurde, mit . . .,8 würde auf Diskette geschrieben. Entsprechendes gilt auch für den Befehl 'l'. Beispiel: l "inhalt 85* ",8 lädt die Datei 'inhalt 85' von Diskette, die Sie übrigens innerhalb von INPUT 64 mit 's' auch auf eigenen Datenträger abspeichern konnten.

So lassen sich auch weitere Dateien anlegen: Setzen Sie im Disketten-Betrieb s " @:datei02 ",8, wird damit die Datei 'datei02' überschrieben; aber Vorsicht, das DOS-Betriebssystem der Disketten-Station kann dabei Fehler machen. Besser zum Beispiel:

@s:datei02 löscht die Datei 'datei02' und anschließend s "datei02 ",8 schreibt die neue Version wieder auf die Disk. Wechseln Sie die Diskette, können Sie mit @ i die Diskette initialisieren. @ sendet die bekannten DOS-Befehle an die Disketten-Station (siehe Floppy-Handbuch). So erscheint nach @ $ die Directory der Diskette. 'f1/f3' ist leider vom Druckfehler-Teufel erwischt worden. Richtig ist:
f1/f2 - Blättern vor und zurück
Der Fehler-Teufel geht inzwischen leider mit der Zeit und hat sich auch in das Programm eingeschlichen. Haben Sie mit c9,9,1 weiße Schrift auf braunem Hintergrund eingestellt, verfärbt sich der Bildschirm-Hintergrund, nach ←&e und Hin- und Herblättern mit f1/f2, orange. Sie können den 'Bug' eigenhändig verjagen. Laden Sie MiniDat von Ihrem Datenträger, geben Sie folgende POKES ein:

POKE 8227,42:POKE 8661,42:POKE 8235,15

und speichern Sie MiniDat wieder auf Ihrem Datenträger ab. Jetzt haben Sie die Farben voll im Griff.
Das Zeichen '&' bedeutet: Beide Tasten gleichzeitig drücken (zum Beispiel '←' und 'a').
Die Markierungen für Zeilen/Seitenende lassen sich mit dem Befehl 'm' verändern, um Fremddateien einlesen zu können.

Beispiel:
m 010/255 würde eine Datei erwarten, bei der die Zeilen mit CHR$(10) und die Seiten mit CHR$(255) abgeschlossen wurden.
Ein Listen des Programms ist nicht möglich, da es sich um ein Maschinen-Programm handelt.

---

**Wettbewerbsgewinner:**

# LISP64

Wenn wir den Software-Markt recht überblicken, handelt es sich bei diesem Programm des Monats um eine Weltneuheit. Wer vor 5 Jahren prophezeit hätte, daß eine Implementation von LISP auf einem Heim-Computer überhaupt möglich ist, wäre wahrscheinlich als Spinner abgetan worden. Peter Feldtmann (Jahrgang '63) hat das LISP-System während seines Informatik-Studiums an der Uni Hamburg entwickelt. "In Ergänzung zum nüchternen Studium begann ich bald – nachdem ich am uni-eigenen Großrechner die Bekanntschaft mit der Sprache LISP gemacht hatte und sofort fasziniert war – mit der Planung und Entwicklung eines LISP-Interpreters für meinen häuslichen Rechner, den C 64. Dabei mußte ich feststellen, daß Implementations-Hinweise in der Literatur recht dünn gesät und speziell für kleinere Rechner überhaupt nicht vorhanden waren. So war ich weitgehend auf mich selbst angewiesen, und nach und nach entstand dann etwas, was jetzt LISP64 heißt" beschreibt Peter Feldtmann die Entstehungsgeschichte des Systems.

Ehe Sie die ersten Gehversuche mit dieser Sprache beginnen, ein paar Anmerkungen zu den naheliegenden Fragen:
LISP – was ist das eigentlich?
LISP – was soll das überhaupt?
LISP – woher kommt das?

LISP steht als Abkürzung für "LISt Processing language" (frei übersetzt: Listen-Verarbeitungs-Sprache) und ist eine der ältesten existierenden Programmier-Sprachen. Sie wurde bereits 1958 am MIT (Massachusetts Instituts of Technology, die Hochburg der theoretischen Computer-Forschung schlechthin) entwickelt. LISP ist sozusagen die "Mutter" verschiedener neuerer Programmier-Sprachen (Modula, Small-Talk etc.), die sich die Struktur und Systematik von LISP zunutze gemacht haben. LOGO wurde sogar in LISP geschrieben. LISP ist die Sprache, ohne die die Entwicklung der "Künstlichen Intelligenz" nicht möglich gewesen wäre und gewinnt nach einem zwanzigjährigen Außenseiter-Dasein zunehmend auch an Breitenbedeutung.

LISP ist (meistens) eine Interpreter-Sprache wie zum Beispiel BASIC. Das System befindet sich also ständig in einer Warteschleife und wartet auf Eingaben des Benutzers, wertet diese aus, um entweder eine Fehlermeldung auszuspucken oder zur Freude des Benutzers seine Anweisungen auszuführen, dann wieder zu warten und so fort. Damit hören die Parallelen zu BASIC auch schon auf. Denn LISP ist eine "ganz andere" Spache. Vor allem ganz anders als BASIC. Denn:

## LISP im Allgemeinen . . ..

In LISP gibt es keine formalen Unterschiede zwischen Programmen und Daten. Daher können Programme andere Programme erzeugen oder sich selbst verändern. Dies ist eine der Voraussetzungen für die bequeme Programmierung selbstlernender Systeme.

In LIPS sind Datenstrukturen beliebiger Tiefe möglich, ohne daß diese Struktur vorher festgelegt sein muß, wie beispielsweise bei Arrays in BASIC.

In LISP lassen sich Befehle selbst definieren. Der Interpreter behandelt nach einer Benutzer-Definition den neuen Befehl, als wär's ein Teil von ihm.

In LISP ist rekursive Programmierung möglich.

Das heißt, aus einer Funktion heraus kann ebendiese Funktion aufgerufen werden.

In LISP herrscht die "Polnische Notation". An diesem Begriff soll auch gleich einiges des oben Angepriesenen konkreter entwickelt werden:

Normalerweise schreibt man eine Funktion bzw. den Operator zwischen die Argumente. Also: 3 + 4. (Infix-Notation) Wer schon einmal ein FORTH-Programm gesehen hat, kennt die "umgekehrt polnische Notation": 3 4 + – der Operator steht nach den Argumenten. (Sinngemäß also Postfix-Notation, dieser Ausdruck ist aber ungebräuchlich.) Diese polnische Notation – so benannt nach einer in Polen ansässigen Logiker-Schule – nicht umgekehrt heißt dann: + 3 4; der Operator steht vor den Argumenten (Prefix-Notation).

## . . . und Besonderen

Der Ausdruck "+ 3 4" ist fast schon ein fertiges LISP-'Programm'. Es fehlen noch die Klammern, die in dieser Sprache jeden Ausdruck einschließen: (+ 3 4). Und das System kennt nicht das Zeichen "+", sondern möchte es ausgeschrieben haben: (PLUS 3 4). Ergebnis: 7. Falls Sie beim gewohnten "+" bleiben möchten, definieren Sie sich diesen Befehl selbst:

```
(DE + (X Y)
(PLUS X Y)
)
```

DE heißt "Jetzt wird eine neue Funktion definiert", danach folgt, durch Leerzeichen getrennt, der Name der neuen Funtion, in diesem Fall "+". In den Klammern stehen die Argumente, die unserer Funktion beim Aufruf übergeben werden, und in der nächsten Zeile, was mit ihnen geschehen soll. Die letzte, aus Gründen der Übersichtlichkeit so allein stehende, Klammer schließt die Definition ab. Der Interpreter liefert den Namen der neuen Funktion zurück: +. (Wenn Sie unvermutet viele Klammern eingeben müssen, um die Definition abzuschliessen, ist meist irgendetwas an der Logik faul . . .) Wollen Sie nur zwei Zahlen zusammenzählen , tut's nach dieser Definition der Befehl (+ 3 4). (Mit der Funktion PLUS können eigentlich beliebig viele Argumente verarbeitet werden, es geht also auch: (PLUS 3 4 6 128 34). Das Beispiel sollte aber zunächst möglichst einfach den DE-Befehl klarmachen.) Das Ergebnis 7 wird übrigens an das Terminal ausgegeben, ohne daß irgendwo eine PRINT-Anweisung steht! Diese 'automatische Ausgabe' ist LISP-typisch: nach der Eingabe von (PRINT (PLUS 3 4)) steht die Ziffer 7 zweimal untereinander auf dem Bildschirm.

Nächstes Beispiel: Rekursion wird in allen bekannten Lehrbüchern am Beispiel der Fakultäts-Funktion abgehandelt. Dieser Tradition wollen wir uns auch hier nicht entziehen. Also:

Definition 1 – Die Fakultät einer natürlichen Zahl n (geschrieben n!) ist das Produkt aller natürlichen Zahlen von 1 bis n. So ist
7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040

Definition 2 – kürzer und rekursiv:
n! = n * (n-1)!
Im Klartext: "Die Fakultät einer Zahl n ist das Produkt von n und der Fakultät von n-1". Damit sich die Katze nicht in den Schwanz beißt, muss dem ständigen Selbstaufruf ein klares Ende gesetzt werden. Falls n = 0, soll gelten n! = 1. Diese 'Abbruchbedingung' nennt sich im Informatiker-Jargon "Rekursions-Basis". Die vollständige rekursive Definition heißt somit:
Wenn n = 0, dann n! = 1
Sonst immer: n! = n*(n-1)!

Diese Definition läßt sich leicht in eine LISP-Funktion umsetzen:

```
(DE FAKU (N)
[COND [(ZEROP N) 1]
[T (TIMES N (FAKU (SUB1 N]

]
)
```

In der ersten Zeile wieder der DE-Befehl, gefolgt vom neuen Funktionsnamen und dem zu übergebenden Argument. COND ist, flapsig ausgedrückt, das IF-THEN-ELSE von LISP, und bezieht sich auf eine Reihe von Bedingungs-Aktions-Paaren, die alle LISP-gemäß in Klammern stehen. Die erste Bedingung ist (ZEROP N), umgangssprachlich "Prüfe, ob N gleich Null ist!". Ist dies der Fall, diese Bedingung also wahr, dann soll der Wert 1 zurückgegeben werden. Ansonsten geht es zur nächsten Bedingung (eine Zeile tiefer). Dort steht im Bedingungsteil ein schlichtes T. Das T steht in LISP für den Wahrheitswert "true" (wahr). Durch das Einsetzen von T im Bedingungsteil dieser Zeile wird der zugehörige Ausführungsteil immer (!) bearbeitet, weil die Bedingung eben ausdrücklich auf wahr gesetzt wurde. Man hätte auch schreiben können: N = N, 3 = 3 oder einen anderen Ausdruck, der in jedem Fall wahr ist. Die daraufhin auszuwertende Funktion ist (TIMES N (FAKU (SUB1 N))). TIMES bildet das Produkt beliebig vieler Argumente. Die Argumente sind N und die Funtion (FAKU (SUB1 N)).

Eines der Argumente ist also ein neuer Aufruf der FAKU-Funktion, der als Argument (SUB1 N) übergeben wird. (SUB1 N) ist wiederum ein LISP-Ausdruck und bedeutet "Ziehe von N Eins ab".

Die letzte, eckige Klammer schließt alle noch geöffneten Klammern. Das System gibt FAKU zurück und "weiß" künftig, daß beispielsweise (FAKU 6) gleich 720 ist.

Zum Schluß noch ein Wort zu den Begriffen ATOM und LISTE. Dies sind die unter LISP verfügbaren Daten-Formen. Ein Atom ist wie in der Physik unteilbar, es kann ein numerisches Atom sein (eine Zahl), eine nicht durch Leerzeichen getrennte Zeichenfolge oder ein String. Eine Liste beginnt mit der obligatorischen "(" und enthält beliebig viele Elemente. Die Elemente sind entweder Atome oder wiederum Listen. Abgeschlossen wird jede Liste mit ")". Zum Beispiel ist

```
'(BANANE APFEL BIRNE)
```

eine Liste mit drei Elementen, jedes dieser Elemente ist ein Atom. Diesen Atomen – und das ist einer der Clous von LISP – können Eigenschaften mit bestimmten Werten zugeordnet werden. Zum Beispiel:

```
(PUTPROP 'BANANE 'FORM 'KRUMM)
```

Dadurch wird dem Atom BANANE unter der Eigenschaft FORM der Wert KRUMM zugeordnet. Dies geht – um auf die Datenstrukturen beliebiger Tiefe zurückzukommen – beliebig oft:

```
(PUTPROP 'BANANE 'GESCHMACK
'GUT)
(PUTPROP 'BANANE 'KONSISTENZ
'WEICH)
```

und kann – ohne sich um Felder, Indizes oder ähnliches kümmern zu müssen – genauso einfach wieder abgeholt werden:

(GETPROP 'BANANE 'GESCHMACK) ergibt:
GUT
(GETPROP 'BANANE 'KONSISTENZ) ergibt:
WEICH

und so fort. Dies geht auch in die Tiefe:

```
(PUTPROP 'BANANE 'FARBE 'GELB)
(PUTPROP 'GELB 'HAEUFIGKEIT 'OFT)
```

und entsprechend:

```
(GETPROP (GETPROP 'BANANE 'FARBE)
'HAEUFIGKEIT)

= OFT
```

Überlegen Sie einmal, wie einfach man dadurch baumstrukturierte Datenbänke oder ein Wörterbuch anlegen kann!

Das Hochkomma vor "Banane" und so weiter ist übrigens das Kürzel für "Quote" und bedeutet, daß der Interpreter nicht nach dem Wert von "Banane" suchen, sondern diesen Begriff als solchen nehmen soll.

## Das System

Dies soll als Vorgeschmack auf die Beschäftigung mit LISP reichen. Wir können natürlich innerhalb des Magazins keinen LISP-Lehrgang anbieten! Falls Sie in diese Sprache einsteigen wollen, müssen Sie sich (mindestens) eins der an anderer Stelle in diesem Heft besprochenen LISP-Bücher zu Gemüte führen!

Unsere Rolle 'beschränkt' sich darin, Ihnen ein komplettes LISP-System zur Verfügung zu stellen; um LISP zu lernen oder – falls Sie es schon können – auf dem 64er anzuwenden. Da dieses Systen sehr umfangreich ist, wird LISP64 in mehreren Teilen veröffentlicht.
In dieser Ausgabe erhalten Sie den LISP-Interpreter, einige kleine Beispielprogramme, die den Einstieg erleichtern sollen, und eine richtige LISP-Anwendung: Ein Programm zum symbolischen Differenzieren. (Unmathematischer ausgedrückt: dieses Programm kann ableiten.) Das ist schon mehr als ein einsatzfähiges LISP-System.

Zu einer wirklich guten LISP-Implementation gehören außerdem noch ein TRACE-Modul (zur Fehlererkennung und Behandlung) und ein Listen-Editor, mit dem vorhandene Funktionen oder Listen komfortabel verbessert oder geändert werden können. Tracer und Editor finden Sie in der nächsten Ausgabe, sowie drei Befehlserweiterungen: eine zur Mengen-, eine zur Array-Verarbeitung und MACROS.LSP. Letzteres enthält so nützliche Dinge wie REPEAT, FOR und andere Kontrollstrukturen. Damit ist das LISP-System komplett. Als Zugabe veröffentlichen wir eine Ausgabe später zwei größere, LISP-typische Anwenderprogramme.

## Abspeichern auf eigenen Datenträger

Den LISP-Interpreter selbst können Sie wie üblich durch CTRL und s auf Ihren eigenen Datenträger überspielen. Die mitgelieferten Programme (auch die Befehlerweiterung MACROS.LSP) sind sequentielle Files (auf Diskette als USR-Programme abgelegt). Diese werden aus dem Auswahl-Menue im Modul abgespeichert. Laden der Programme vom eigenen Datenträger (das heißt unter LISP auch: Einfügen in die OBLIST, also Einbinden in den Interpreter) erfolgt unter LISP64 durch:
(LOAD ga "name")
ga ist die Geräteadresse. Zum Beispiel:
(LOAD 7 "HANOI.LSP")

## Beispielprogramme...

Die "Befehlserweiterung" MACROS.LSP ist unten beschrieben. Drei kleine Programme sollen den Einstieg in die LISP-Programmierung erleichtern:

TOH.LSP steht für das bekannte orientalische Spiel "Towers of Hanoi". Es geht dabei darum, einen Turm aus Scheiben absteigender Größe (die kleinste Scheibe liegt oben) von einem Startfeld unter Benutzung eines Hilfsfeldes auf ein Endfeld zu schichten. Eine größere Scheibe darf niemals auf eine kleinere gelegt werden (Bild 2). Der theoretische Ansatz zur rekursiven Lösung dieses Problems ist in der einschlägigen Literatur schon häufig beschrieben worden, sehr allgemeinverständlich unter anderem im c't-Special 1 "Software Knowhow" (Heise-Verlag).

HANOI.LSP behandelt das gleiche Problem, gibt aber gleichzeitig die Anzahl der Schritte aus. Interessanter ist aber – und deswegen auch zwei Programme zum selben Problem – die unterschiedliche Realisierung im Vergleich zu TOH. Dieses Programm wurde dem Buch "Einführung in das Programmieren in LISP" von C.M. Hamann (de Gruyter Verlag) entnommen.

PERM.LSP permutiert eine Liste mit beliebig vielen Elementen. Eine Demonstration liefert die Eingabe von (DEMO). PERM selbst wird ausgeführt durch (PERM Liste), zum Beispiel (PERM '(HASE KATZE IGEL FUCHS)). PERM ruft auf PERM1, PERM2 und TAUSCH. PERM1 und PERM2 sind zwei sich gegenseitig aufrufende Funktionen. Dies ist zwar nicht gerade der einfachste Algorithmus, aber sehr instruktiv für die LISP-Programmierung.

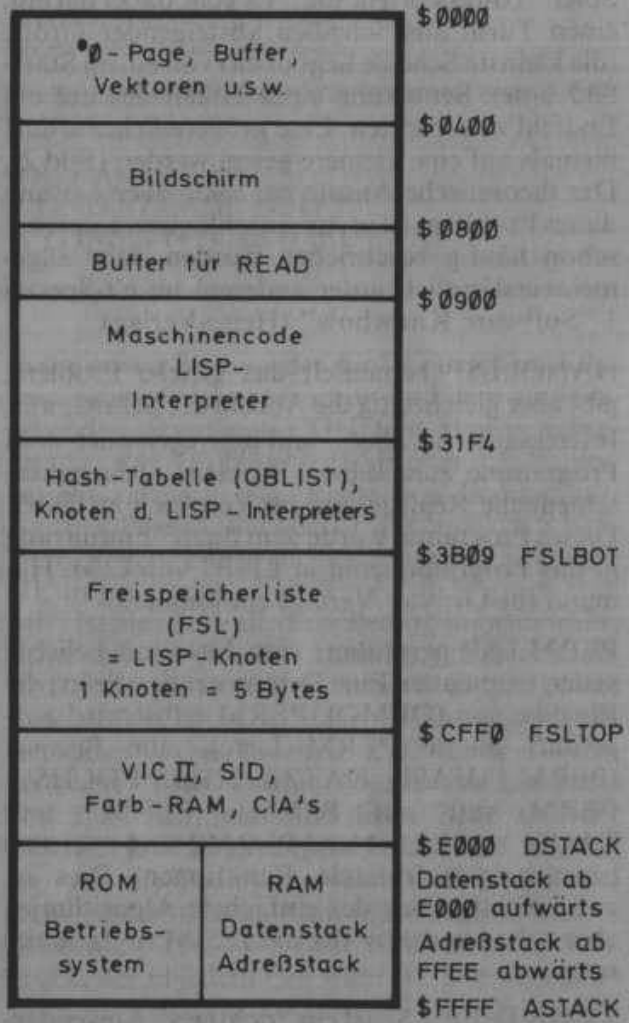
SYMB-DIFF.LSP ist ein "richtiges" Anwenderprogramm in LISP. Diese Funktion erstellt – wahlweise – die erste bis fünfte Ableitung einer beliebigen Funtion. Der Aufruf erfolgt mit (DO), dann wird hinter dem "F(X) =" die zu differenzierende Funktion erwartet. Achten Sie bei der Eingabe auf die LISP-typische Notation und die trennenden Leerzeichen. Also nicht eingeben:

(X§F5),
sondern
(X §F 5)
Außerdem gilt: INFIX-Notation (LISP untypisch), keine Punkt-vor-Strich-Regel. Zum Beispiel wird die Funktion
$4X+5X^2+8SIN(2X)$
eingegeben als:

((4 * X)+(5 * (X §F 2))+(8 * (SIN (2 * X]

Eine Erklärung der Arbeitsweise werden Sie hier im Heft vergeblich suchen. Denn, wie oben schon gesagt, wir können Ihnen zwar ein LISP-System, aber keinen LISP-Lehrgang liefern.

**Bild 1**




Speicher-Konfiguration prinzipiell veränderbar durch folgende Adressen:

$0906 (dez. 2310) FSLTOP
$0908 (dez. 2312) DSTACK
$090A (dez. 2314) ASTACK

FSLBOT ($0904/dez. 2308) darf nicht verändert werden!

## Details der Implementation

Die Speicherplatzbelegung von LISP64 ist in Bild 1 dargestellt. Da das System im RAM liegt, ist es natürlich gegen unbedachte POKE-Befehle nicht geschützt! Damit LISP64 auch für Datasetten-Benutzer vernünftig handhabbar ist, wurde das schnelle Lade- und Speicherverfahren SuperTape im System installiert. Sie brauchen also SuperTape nicht vorher laden!

Es gelten folgende Festlegungen:

a) Zahlen:
32-Bit-Integer, vorzeichenbehaftet. Damit ist der Zahlenbereich von $-2^{31}$ bis $2^{31}-1$ darstellbar, das heißt: -2**147**483**648 bis +2**147**483**647.

Strings:
in Hochkommata (") eingeschlossene Zeichenketten von maximal 128 Zeichen Länge.

Literals:
Zeichenketten bis maximal 80 Zeichen Länge, keine Leerzeichen oder syntaktische Zeichen (;.()[]"") enthaltend. Literals sind zum schnellen Zugriff in einer hash-adressierten Liste organisiert – daher auch die "merkwürdige" Anordnung bei (OBLIST).

b) Systemfunktionen und System-Literals (VALUE, EXPR, NIL ...) können keine Eigenschaftslisten besitzen.

c) Der von Zeit zu Zeit auftauchende weiße Stern in der rechten oberen Bildschirmecke ist das Zeichen des tätigen "Garbage Collectors" (GC).

d) LISP64 ist ein EVAL-Lisp, das heißt: erstes Element = Funktion, Rest = Argumente. Funktionen können sein:
- Systemfunktionen (CAR, PLUS...)
- LAMBDA-Ausdruck
- NLAMBDA-Ausdrücke (Argumente werden nicht ausgewertet)
- LABEL-Ausdrücke
- durch DE, DF, DM definierte Funktionen
- VALVE eines Literals (SETQ FIRST 'CAR), (FIRST '(A B)) = a

e) Beim Definieren neuer Funktionen gilt:
in der Form
(DE Name Varlist ...) bzw.
(DF Name Varlist...)
bedeutet:

Varlist = NIL / Varlist = () – Funktion erwartet keine Argumente.
Varlist = Atom (non-NIL) – Funktion hat beliebig viele Argumente, Varlist enthält die Liste mit diesen Argumenten.
Varlist = Liste von Literalen: jedes Literal ist lokale Variable und nimmt beim Funtionsaufruf den Wert des zugehörigen Arguments an.

f) Funktionskörper und COND-Klauseln sind grundsätzlich vom PROGN-Typ, das heißt, Ergebnis ist der Wert der Ausführung des letzten von beliebig vielen, nacheinander ausgewerteten Ausdrücken.

## Fehlermeldungen

Die Fehlermeldungen werden nach dem Muster:

[Fehlermeldungs-Text] [:Term]
[IN name]
erzeugt. Der Zusatz [Term] erscheint nur, wenn "Term" ungleich NIL ist; der Zusatz [Name] nur , wenn der Fehler während der Ausführung einer benutzerdefinierten Definition "Name" eintritt.

Nach Ausgabe einer Fehlermeldung sowie der Erzeugung eines Programmabbruchs (durch RUN/STOP) befindet sich das LISP-System wieder in der Interpreterschleife. Lokale Variablen (von Funktionen, LAMBDA-Ausdrücken, PROGs) behalten ihre zum Zeitpunkt des Fehlerfalls gültigen Werte. Zur Fehleranalyse kann man jetzt mit der Funktion (UNBIND) jeweils die letztgültigen Variablen-Bindungen zurücknehmen, zum Beispiel eine rekursive Funktion Schritt für Schritt zurückverfolgen. Die Funktion (RESET) setzt alle benutzten Variablen wieder auf ihre globalen Werte zurück und sollte vor dem Neustart einer Funktion nach einem Fehlerfall ausgeführt werden.

### Die Fehlermeldungen im einzelnen:

EMPTY STACK
Term = NIL. Es wird ein Zugriff auf den leeren Datenstack versucht; meist bei Aufruf einer Funktion mit fehlenden Argumenten, zum Beispiel ((CONS)

STACK OVERFLOW
Term = NIL. Der Datenstack ist voll, zum Beispiel bei einer nicht-terminierten Rekursion.

UNBOUND VARIABLE
Term = Variablen-Name. Es wird versucht, den Wert einer Variablen zu ermitteln, die weder global (durch SETQ) noch lokal (in Funktion, LAMBDA-Ausdruck, PROG) mit einem Wert belegt wurde.

UNDEFINED FUNCTION
Term = Funktionsname. Es wird versucht, eine noch nicht mit DE, DF oder DM definierte Funktion aufzurufen.

NON-NUMERIC ARGUMENT
Term = Argument der numerischen Funktion. Es wird versucht, eine numerische Operation mit einem nicht-numerischen Wert (zum Beispiel NIL) durchzuführen, beispielsweise (PLUS 3 'A).

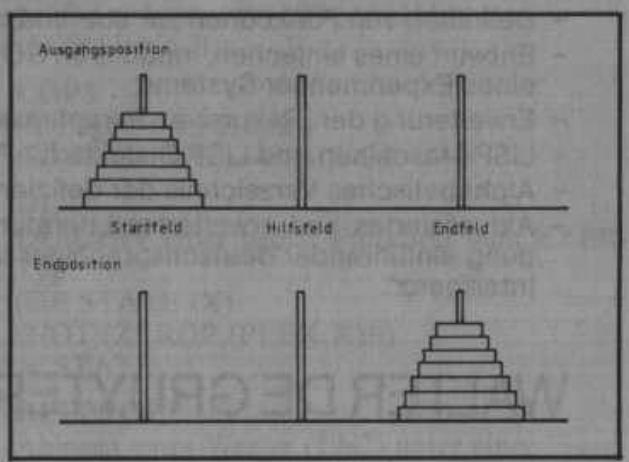
READ-ERROR
Term = NIL. Beim Einlesen (READ, READL) eines LISP-Ausdrucks wird ein syntaktischer Fehler festgestellt, zum Beispiel (A..B).

I/O ERROR # n
+++ I/O ABORTED +++
Beim Arbeiten mit den Eingabe-/Ausgabe-Funktionen (OPEN, INPUT, und so weiter) tritt ein Fehler auf. "n" hat die Bedeutung wie im C64-Betriebssystem.

1 = too many files open
2 = file open
3 = file not open
4 = file not found
5 = device not present
6 = not input file
7 = not output file
8 = missing filename
9 = illegal device number

MISSING PARAMETER
Term = OPEN. In einem OPEN-Befehl fehlen die nötigen Angaben, zum Beispiel Filenummer, Gerätenummer.

## Bild 2



# LISP64 – Der Befehlssatz

Die Befehle selbst sind jeweils fett gedruckt, danach folgt die Erklärung und anschließend ein oder mehrere Beispiele.

*****
Kommentarkennzeichnung
(* BEISPIEL 1)

**ABS**
Absolutbetrag
(ABS -4) = 4

**ADD1**
Inkrementieren (Zahl + 1)
(ADD1 4) = 5

**AND**
Und-Verknüpfung von Wahrheitswerten
(AND NIL NIL T NIL) = NIL
(AND T T T) = T

**APPEND**
Aneinanderhängen von Listen
(APPEND '(A B) '(C D) '(E F)
= (A B C D E F)

**APPLY**
Anwenden einer Funktion auf die Argumentliste
(APPLY 'PLUS '(3 4)) = 7

**APPLY***
Anwenden einer Funktion auf die Argumente
(APPLY* 'PLUS 3 4) = 7

**ASC**
Ermittlung des ASCII-Codes
(ASC "A") = 65

**ASSOC**
Suchen (EQ-Vergleich) in Assoziationsliste
(ASSOC 'X '((X.A) (Y.B))) = (X.A)

**ATOM**
prüft auf atomar
(ATOM 'A) = T

**CAAR**
(CAR(CAR. . . ))
CAAR '((A B)C)) = A

**CADR**
(CAR (CDR. . . ))
(CADR '(A B C D)) = B

**CALL**
Maschinenprogrammaufruf, A/X/Y/ST in Adressen 780-783
(CALL 65487)= NIL

**CAR**
Erstes Element einer Liste
(CAR '(A B)) = A

**CDAR**
(CDR(CAR. . . ))
(CDAR '((A B) C)) = (B)

**CDDR**
(CDR(CDR. . . ))
(CDDR '(A B C D)) = (C D)

**CDR**
Liste ohne erstes Element
(CDR '(A B C))= (B C)

**CHAR**
wandelt ASCII-Code nach Zeichen
(CHAR 65) = "A"

**CLOSE**
schliessen einer mit OPEN eröffneten Datei
(CLOSE 1)= T

**COMPL**
Komplementbildung
(COMPL 13) = -14

**COND**
bedingter Ausdruck:
(COND
(b1 a11 a12 . . .)
(b2 a21 a22 . . .)
. . .
(bn an1 an2 . . .))
wenn eine Bedingung bi ≠NIL dann ai1 ai2 . . . auswerten
(COND ((EQ (READ)'J)'JA)
(T 'NEIN))

**CONS**
konstruieren von S-Expr (Listen)
(CONS 'A 'B) = (A.B)
(CONS 'A '(B C)) = (A B C)

**CONSP**
prüft auf Liste = (NOT(ATOM. . .))
(CONSP '(A.B)) = T

**COPY**
Erzeugen einer Kopie
COPY '(A B C)) = (A B C))

**DE**
Definieren einer neuen Funktion vom EXPR-Typ
(DE STATE (X)
(NOT (ZEROP (PEEK X))))
= STATE

**DEFPROP**
Ablegen eines Wertes (1982) unter einer Eigen-

schaft (BAUJAHR) auf der Eigenschaftsliste des Atoms (AUTO) FEXPR ! (s.a. PUTPROP)
(DEFPROP AUTO BAUJAHR 1982)
= 1982

**DF**
Definieren einer neuen Funktion vom FEXPR-Typ
(DF TEST (X Y)
(CONS (LIST X) (LIST Y)))
= TEST

**DIFFERENCE**
Differenzenbildung
(DIFFERENCE 10 3) = 7

**DIR**
Inhaltsverzeichnis der Diskette lesen
(DIR)

**DISK**
Befehl zur Floppy senden
(DISK "S:FILE1") = T

**DM**
Definieren einer MACRO-Funktion
(DM NCONS L
(LIST 'CONS (CADR L) NIL))
= NCONS

**EQ**
prüft auf Atom- oder Zeigergleichheit
(EQ 'A 'A) = T
(EQ 4 5) = NIL

**EQUAL**
prüft auf Strukturgleichheit
(EQUAL '(A B C) '(A B C)) = T

**ERROR**
Fehlermeldung erzeugen, Rücksprung auf top-level
(ERROR '(DIVISION DURCH 0))

**EVAL**
evaluieren eines S-Expr
(EVAL '(PLUS 3 4)) = 7

**EXIT**
Verlassen des LISP-Systems
(EXIT)

**GC**
Garbage Collector aufrufen, ermittelt Anzahl freier Knoten
(GC) = Nr.

**GETCHAR**
ein Zeichen lesen
((GETCHAR) = "A")

**GETDEF**
ermitteln der Funktions-Definition (property) eines Atoms
(GETDEF TEST)
= (DF TEST (X Y) (CONS . . .))
= 1982

**GETPROP**
Ermitteln des Wertes (81) einer Eigenschaft (BAUJAHR) auf der Eigenschaftsliste eines Atoms (AUTO)
(GETPROP 'AUTO 'BAUJAHR)
= 81

**GETPROPLIST**
ermitteln der Eigenschaftsliste eines Atoms
(GETPROPLIST 'AUTO)
= (BAUJAHR 1982 . . .)

**GO**
in PROG: Sprung auf Marke
(GO LOOP1)

**GREATERP**
Zahlenvergleich auf größer
(GREATERP 5 3) = T

**HBYTE**
höherwertiges Byte einer Adresse
(HBYTE 1030) = 4

**INPUT**
legt alle Eingaben (READ, GETCHAR usw.) auf das mit (OPEN 1 . . .) eröffnete File
(INPUT 1)

**LAST**
letztes Element einer Liste
(LAST '(A B C)) = C

**LBYTE**
niederwertiges Byte einer Adresse
(LBYTE 1030) = 6

**LENGTH**
Anzahl der Elemente einer Liste
(LENGTH '(A B C)) = 3

**LESSP**
Zahlenvergleich auf kleiner
(LESSP 3 5)= T

**LINE**
maximal Zeilenlänge für Ausgaben
(LINE 80) = 80

**LIST**
bilden einer Liste
(LIST 'A 'B 'C) = (A B C)

**LOAD**
Laden einer mit SAVE abgespeicherten LISP-Datei von Atom-Eigenschaftslisten. SuperTape ist integriert!
1 = Laden von Kassette
7 = SuperTape-Laden von Kassette
8 = Laden von Diskette
(LOAD 8 "FILE1") = (. . .)
(LOAD 7 "FIL*") = (. . .)

**LOGAND**
logisches (bitweises) Und
(LOGAND 12 4) = 4

**LOGOR**
logisches (bitweises) Oder
(LOGOR 8 4) = 12

**LOGXOR**
logisches (bitweises) Exklusiv-Oder
(LOGXOR 12 4) = 8

**MAP**
sukzessives Anwenden einer Funktion auf eine Liste, deren CDR, CDDR und so weiter.
(MAP 'PRINT '(A B C)) =
(A B C)
(B C)
(C)
NIL

**MAP2CAR**
sukzessives Anwenden einer zweistelligen Funktion auf die jeweils stellengleichen Element-Paare zweier Listen
(MAP2CAR 'CONS '(A B) '(1 2))
= ((A.1)(B.2))

**MAPC**
sukzessives Anwenden einer Funktion auf die Elemente einer Liste
(MAPC 'PRINT '(A B C)) =
A
B
C
NIL

**MAPCAN**
entspricht dem Ausdruck (APPLY 'NCONC (MAPCAR. . .))
(MAPCAN '(LAMBDA (X)
(LIST X X)) '(A B C))
= (A A B B C C)

**MAPCAR**
sukzessives Anwenden einer Funktion auf die Elemente einer Liste; Ergebnisse in einer Liste sammeln
(MAPCAR '(LAMBDA (X)
(LIST X X) '(A B C))
= ((A A) (B B) (C C))

**MAPLIST**
sukzessives Anwenden einer Funktion auf eine Liste, deren CDR, CDDR usw; dabei Ergebnisse in einer Liste sammeln
(MAPLIST 'PRINT '(A B C)) =
(A B C)
(B C)
(C)
((A B C) (B C) (C))

**MEMBER**
Element in einer Liste suchen wenn gefunden, Liste ab Element als Ergebnis
(MEMBER 'B '(A B C)) = (B C)

**MINUS**
Vorzeichenwechsel
(MINUS 4) = -4

**MINUSP**
Zahlenvergleich auf negativ
(MINUSP -4) = T

**MSG**
'Message' ausgeben: T gibt CR aus, Zahlen geben Leerstellen (spaces) an
(MSG T "TEXT" 4 "TEXT2 ") =
TEXT TEXT2 NIL

**NCONC**
destruktives Aneinanderketten von Listen
(NCONC '(A B) '(C D) '(E F)) =
(A B C D E F)

**NCONC1**
destruktives Verlängern einer Liste um ein Element
(NCONC1 '(B C) 'A) = (B C A)

**NORMAL**
Aufhebung eines INPUT/OUTPUT-Befehls: Eingabe von Tastatur, Ausgabe auf Bildschirm
(NORMAL) = T

**NOT**
Negation
(NOT NIL)= T

**NTH**
Liste ab N-tem Element
(NTH '(A B C D) 3) = (C D)

**NULL**
prüft auf leere Liste = NIL
(NULL '()) = T

**NUMBERP**
prüft auf numerisches Atom

(NUMBERP 4) = T

**OBLIST**
Ausgabe der Objektliste mit allen Atomnamen
(OBLIST)

**OPEN**
Eröffnen einer Eingabe- oder Ausgabedatei
(OPEN1 4) für Drucker
(OPEN 1 8 2 "TEST,S,W") = T

**OR**
Oder-Verknüpfung von Wahrheistwerten
(OR F F T F) = T
(OR NIL NIL F) = NIL

**OUTPUT**
leitet alle Ausgaben auf ein mit (OPEN 1 . . .) eröffnetes File um
(OUTPUT 1)

**PACK**
erzeugen eines neuen Atomnamens oder Strings durch Zeichenverkettung

(PACK '(A B C D)) = ABCD
(PACK '("TE","ST",44))
= "TEST44"

**PDEF**
übersichtliche Ausgabe einer Funktionsdefinition = (PP (GETDEF. . .))
(PDEF TEST) =
(DF TEST
(XY) . . . )

**PEEK**
Inhalt einer Speicherzelle
(PEEK 1) = 54

**PLUS**
Addition beliebig vieler Werte
(PLUS 1 2 3 4) = 10

**POKE**
Adresse mit Wert besetzen
(POKE 650 128) = 128

**PP**
Übersichtliche Ausgabe einer Liste ("pretty print")
(PP '(AB(CD(E)))) =
(AB
(CD
(E)))

**PRIN1**
Ausgabe ohne CR, Strings ohne Hochkomma (")
(PRIN1 '(A "B" C)) = (A B C)

**PRINC**
Ausgabe ohne CR und ohne syntaktische Zeichen
(PRINC '(A ("B") ((C))) =
ABC

**PRINL**
Ausgabe ohne syntaktische Zeichen, CR als Zeilenabschluß
(PRINL '(A(B C) (D))) =
ABCD

**PRINT**
Ausgabe eines S-Expr, CR als Zeilenabschluss
(PRINT '(A "B" (C))) =
(A "B" (C))

**PROG**
PROGram-Anweisung: - lokale Variable (X)
- beliebig viele Anweisungen
= evaluierbare S-Expr
- Marken = atomare Namen
Sprung auf Marken mit GO, Verlassen des PROGs mit RETURN
(PROG(X)
M1 (SETQ X (READ))
(COND((EQ X 'QUIT)
(RETURN 'OK)))
(PRINT (EVAL X))
(GO M1))

**PROG1**
beliebig viele S-Expr evaluieren, Wert des ersten zurückgeben
(PROG1 (CAR '(A B))
(SETQ X (CDR X)))
(CDR'(AB)))
= A

**PROGN**
beliebig viele S-Expr evaluieren, Wert des letzten zurückgeben
(PROGN (CAR '(A B))
(SETQ X (CDR X))
(CDR '(A B)))
= (B)

**PUTPROP**
Ablegen eines Wertes (81) unter einer Eigenschaft (BAUJAHR) auf der Eigenschaftsliste eines Atoms (AUTO); EXPR!
(PUTPROP 'AUTO 'BAUJAHR 81)
= 81

**QUOTE**
Quotierung: Unterdrückung der Auswertung eines S-Expr
(QUOTE (A B C)) = (A B C)

(QUOTE A) = 'A = A

**QUOTIENT**
Quotientenbildung, siehe auch REMAINDER
(QUOTIENT 12 4) = 3

**RANDOM**
Zufallszahlenerzeugung im angegebenen Bereich
(RANDOM 10 20) = 14

**READ**
Einlesen eines S-Expr
(READ) = (A B C)

**READCH**
Einlesen eines Zeichens
(READCH) = "A"

**READL**
Einlesen beliebig vieler S-Expr und sammeln in einer Liste
(READL) = ((A B)(C D E))

**REMAINDER**
Rest bei Quotientenbildung
(REMAINDER 13 3) = 1

**REMOB**
Entfernen von Atomen aus der Objektliste
(REMOB CASR TEST TEST1)

**REMOVE**
Entfernen eines Elements aus einer Liste
(REMOVE 'A '(A B A C A A D))
= (B C D)

**REMPROP**
Entfernen einer Eigenschaft des Atoms
(REMPROP 'AUTO 'BAUJAHR)
= BAUJAHR

**RESET**
nach Fehlerfall/Break: Zurücksetzen der Variablen auf ihre globalen Werte
(RESET) = NIL

**RETURN**
Verlassen eines PROGs mit Wertrückgabe
(RETURN 'OK) = OK

**REVERSE**
Umkehrung der Reihenfolge der Listenelemente
(REVERSE '(A B C)) = (C B A)

**RPLACA**
destruktives Ersetzen des CARs einer Liste
(RPLACA '(A B) 'C) = (C B)

**RPLACD**
destruktives Ersetzen des CDRs einer Liste
(RPLACD '(A B) '(C)) = (A C)

**SASSOC**
suchen (EQUAL-Vergleich) in einer Assoziationsliste
(SASSOC '(A B) '(((A B) C)
((D E) F)))
= ((A B) C)

**SAVE**
Abspeichern der Eigenschaftslisten von Atomen in einer Liste
(SAVE 8 "FILE" '(FABFIB ..))
(SAVE 7 "FILE2" FUNCS)

**SET**
Wertzuweisung an Variable
(SET 'X '(A B)) = (A B)

**SETQ**
Wertzuweisung an Variable ohne deren Evaluierung = (SET (QUOTE . . .) . . .)
(SETQ X '(A B)) = (A B)

**SPACES**
Ausgabe von Leerzeichen
(SPACES 10)

**ST**
Einlesen des Floppy-Status
(ST) = (0 OK 0 0)

**STRINGP**
prüft auf Zeichenkette
(STRINGP "ABCD") = T

**SUB1**
Dekrementieren (Zahl - 1)
(SUB1 4) = 3

**TAB**
Tabulatorfunktion
(TAB 10)

**TERPRI**
'terminal printing': CR ausgeben
(TERPRI) = NIL

**TIMES**
Produktbildung
(TIMES 2 3 4) = 24

**UNBIND**
bei Fehlerfall/Break: Aufhebung der letzten Variablen-Bindungen
(UNBIND) = NIL

**UNPACK**
Auflösung eines Atomnamens oder Strings in einzelne Zeichen
(UNPACK 'ABCD) = (A B C D)
(UNPACK "A4") = ("A" 4)

**WAITCHAR**
wartet, bis Zeichen eingegeben wird
(WAITCHAR) = "A"

**ZEROP**
Zahlenvergleich mit 0
(ZEROP 0) = T

**EXPR**
Der unter der Eigenschaft EXPR auf der Eigenschaftsliste eines Atoms abgelegte LAMBDA-Ausdruck gilt als Funktionskörper, siehe auch DE

**F**
F = False: Wahrheitswert 'falsch' = NIL

**FEXPR**
Der unter der Eigenschaft FEXPR auf der Eigenschaftsliste eines Atoms abgelegte NLAMBDA-Ausdruck gilt als Funktionskörper, siehe auch DF

**LABEL**
benennt eine Funktion, die rekursiv benutzt wird, temporär (nur für die Ausführung) mit einem Namen in der Form
((LABEL Name LAMBDA-Ausdruck) Argumente)
wobei im LAMBDA-Ausdruck
(LAMBDA Variablenliste Ausdruck1 Ausdruck2 . . .)
unter den Ausdrücken die Funktionsaufrufform

(Name . . .)
vorkommen darf (Rekursion).

**LAMBDA**
In der Form
((LAMBDA Variablenliste Ausdruck1 Ausdruck2 . . .)
Argument1 Argument2 . . .)
werden zunächst die Argumente evaluiert und mit den Variablen der Variablenliste gebunden (wenn die Variablenliste ein Atom < > NIL ist, wird eine Liste mit allen evaluierten Argumenten an dieses Atom gebunden).
Danach werden die Ausdrücke ausgewertet; der Wert des letzten Ausdrucks ist der Wert des gesamten LAMBDA- Ausdrucks.

**MACRO**
Der unter der Eigenschaft MACRO auf der Eigenschaftsliste eines Atoms abgelegte NLAMBDA-Ausdruck gilt als Funktions-Körper vom MACRO-Typ; siehe auch DM

**NIL**
bezeichnet sowohl ein Atom [(ATOM NIL) = T] als auch die leere Liste [(NULL NIL) = T] und den Wahrheitswert 'falsch'

**NLAMBDA**
In der Form
((NLAMBDA Variablenliste Ausdruck1 Ausdruck2 . . .)
Argument1 Argument2 . . .)
werden die Argumente NICHT ausgewertet, alles weitere siehe LAMBDA

**T**
Wahrheitswert 'true' = wahr

**'**
'Ausdruck ist eine abkürzende Schreibweise für (QUOTE Ausdruck)

**[]**
Die 'Superklammern' erleichtern das Eingeben von verschachtelten Klammerausdrücken: wird ein Ausdruck mit der Klammer '[' begonnen, so schliesst eine spätere Klammer ']' alle noch offenen Klammern '(' bis zur '[' (einschliesslich). Wird eine Klammer ']' ohne zugehörige '[' benutzt, dann werden alle noch offenen Klammern '('geschlossen. Beispiele:

(COND[(ATOM X)(RETURN (LIST X (CAR X]

[T (PRINT (CDR X])

wird zu

(COND((ATOM X)(RETURN (LIST X (CAR X))))

(T (PRINT (CDR X))))

(A B (C D (E) (F (G H]

wird zu

(A B (C D (E) (F (G H))))

**Taste f5**
Druckvorgang anhalten

**Taste f7**
Druckvorgang fortsetzen

**RUN/STOP-Taste**
'BREAK': Evaluierungsprozess abbrechen und Rücksprung auf LISP-top-tevel

## Listing HANOI.LSP

```
(de hanoi
  (n)
  (reset-toh n)
  (setq step 0)
  (transfer-toh 'a
     'c
     'b
    (length a)) t)
nil

t

(de countup
  (n)
  (reverse(countdown n)))
nil

(de countdown
  (n)
  (cond((eq n 0)
        nil)
    (t(cons n
        (countdown(sub1 n))))))
nil

(de reset-toh
  (n)
  (setq c nil)
  (setq b nil)
  (cond((zerop n)
       (setq a
         (countup 4)))
    (t(setq a
        (countup n)))))
nil

(de transfer-toh
  (from to spare n)
  (cond((zerop n)
       (print(list from
             'leer)))
    ((eq n 1)
      (movedisk-toh from to))
    (t(append(transfer-toh from spare
           to
           (sub1 n))
         (movedisk-toh from to)
         (transfer-toh spare to from
           (sub1 n))))))
nil

(de movedisk-toh
  (from to)
  (print(list 'schritt
      (setq step
        (add1 step))
      'bewege
      (car(eval from))
       'von from
       'nach to))
  (cond((null(eval from))
       (print(list from
             'leer)))
    ((or(null(eval to))
      (lessp(car(eval from))
        (car(eval to))))
       (set to
         (cons(car(eval from))
           (eval to)))
       (set from
         (cdr(eval from))))
    (t(print(list 'cant
           'bewege
           (car(eval from))
           'nach
           (car(eval to))))))
nil
```

## Listing PERM.LSP

```
(de demo nil
  (msg t
    "erzeugung von permutationen" t
    "----------------------------" t t)
  (msg "(perm '(1 2 3 4)) = " t t)
  (perm '(1 2 3 4))
  (msg t t
    "(perm '(apfel banane citrone)) = "
 t t)
  (perm '(apfel banane citrone)))
nil

(de perm
  (l)
  (print l)
  (perm1 l
    (cdr l) l) t)
nil

(de perm1
  (l l1 l2)
  (cond((atom l1)
       nil)
    (t(perm1 l
        (cddr l2)
        (cdr l2))
      (perm2 l l1 l2))))
nil

(de perm2
  (l l1 l2)
  (cond((atom l1)
       nil)
    (t(tausch l1 l2)
      (print l)
      (perm1 l
        (cddr l2)
        (cdr l2))
      (tausch l1 l2)
      (perm2 l
        (cdr l1) l2))))
nil

(de tausch
  (x y z)
  (setq z
    (car x))
  (rplaca x
    (car y))
  (rplaca y z))
nil
```

## Listing TOH.LSP

```
(de toh
  (n)
  (bewege 'links
     'rechts
     'mitte n))
nil

(de bewege
  (von nach stift n)
  (cond((zerop n)
        t)
       (t(bewege von stift nach
           (subl n))
         (print(list 'lege
                'scheibe
                'von von
                'nach nach))
         (bewege stift nach von
           (subl n)))))
nil
```

---

**News**

# Sechs LISP-Bücher im Vergleich

Sucht man als interessierter Laie eine gute Einführung in LISP, dann teilt sich das Angebot an Büchern zwanglos in zwei Kategorien: die deutschen und die guten.

Zwei dieser Bücher sind von deutschen Professoren verfaßt, die den scheinbar unbezähmbaren Drang haben, immer wieder von den letzten und größten Dingen zu reden, als da sind: gesellschafts-, wissenschafts- , metamathematische-, sprach- und was-immer-sie-wollen-theoretische Probleme. Wo man doch eigentlich bitte erst mal bloß wissen will, was zum Teufel eine Liste ist und was CAR und CDR bedeutet!

Doch schön der Reihe nach.An erster Stelle der deutschen Bücher möchte ich LISP von Stoyan/Görz nennen. Wie seine beiden Konkurrenten ist es als Einführung gänzlich ungeeignet. Mir kam es zum ersten Mal vor gut einem Jahr in die Finger, als ich bereits jahrelang mit LISP gearbeitet hatte und auch wußte, wie man einen LISP-Interpreter baut. Dennoch bereitete es mir Mühe, bei der ersten Lektüre zu verstehen, wovon die beiden denn eigentlich reden. Und sie schaffen es, nicht nur unverständlich zu bleiben, sondern auch einen abstoßend schauderhaften Professorenstil zu schreiben.

## Die Deutschen: Schwer verdaulich . .

Meine Reaktion auf diese erste Leseerfahrung: das Buch erstmal zur Seite zu legen. Sollten Sie ihm in einer Buchhandlung begegnen und eine Einführung in LISP suchen, dann empfehle ich Ihnen, es mir gleichzutun. Selbst bei leidlichen Englischkenntnissen sind die meisten der im folgenden noch besprochenen englischsprachlichen Einführungen vermutlich leichter zu lesen und erfüllen ihren Einführungs-Zweck besser. Dennoch sollte man das Buch nicht aus seinem Bewußtsein streichen. Denn von den verfügbaren deutschen Büchern ist es das beste.

Deswegen, weil die Autoren tiefe Einblicke in die 'Mechanik' von LISP selbst erlauben (Bau eines LISP-Interpreters und Compilers). Und deswegen, weil sie eine gute Zusammenschau der modernen Strömungen in der Informatik bieten und die wichtigsten zeitgenössischen Programmierstile – das prozedurale Programmieren (der BASIC-, FORTRAN-, Pascal-, COBOL-way of life), das funktionale oder applikative Programmieren (der angestammte Stil in LISP), das deklarative Programmieren nach Prolog-Art und das objektzentrierte Programmieren a la Smalltalk – einüben (und das alles in LISP). Im 64er-LISP kann man viele der Beispiele allerdings nur nachvollziehen, wenn man weiß, wie man die zugrundegelegten Sprachkonstrukte aus TCL-LISP darin nachbilden kann.

Zum letzten deswegen, weil sie einen guten überblick über die wichtigsten Techniken der KI-Programmierung bieten (z.B. Pattern Matching, Produktionssysteme, Backtracking-Algorithmen (klingt doch toll, oder?) (NB, lieber Säzzer: die vielen Klammern sollstu (Sic!) bitte stehenlassen, das erinnert nämlich an LISP!)).

Zwar gibt es etliche amerikanische Bücher, in denen das alles auch – und meist besser geschrieben – zu finden ist, aber man muß es sich da eben aus mehreren, zum Teil recht kostspieligen Quellen zusammenklauben, die zumeist auch noch schwer aufzutreiben sind. Wen es zu diesen Quellen drängt, der wird die ausgezeichnete Bibliographie des Buches von Stoyan/Görz zu schätzen wissen. Fazit: Anfänger, laß die Finger davon! Wer aber die Liebe zu LISP in sich entbrannt fühlt – und welcher fühlende Mensch würde sich nicht in diese wundervollste aller

Programmiersprachen verlieben! – und darob den Dingen auf den Grund gehen will und in seinem übermächtigen Wissensdurst auch einen schröcklichen Stil nicht scheut, dem kann man das Buch wohl empfehlen.

Jedenfalls mehr als das Konkurrenzprodukt von Herrn Schefe. Das Buch von P. Schefe nennt sich Einführung in die Informatik ; ich würde es eher Abschreckung vor der Informatik nennen, und besonders weh tut mir, daß es dieses Ziel mit LISP und Prolog zu erreichen versucht. Inhaltlich ist das Buch eigentlich gar nicht so übel, doch Herr Schefe befleißigt sich eines Stils, der in jedem Nebensatz und Komma verrät, daß er Kant, Hegel, Adorno und alles, was schwer verdaulich ist, gelesen hat und mindest ebenso schwer verdaulich daherschreiben kann wie die. Aber schließlich soll Wissenschaft ja auch keinen Spaß machen, denn sonst wär's keine.

Anders als das Buch von Stoyan/Görz habe ich dieses Buch nach den ersten hundert Seiten nicht mehr ganz durchgelesen, sondern nur Stichproben gezogen. Dabei bin ich auf so gut wie nichts gestoßen, was nicht auch bei den anderen – und besonders den amerikanischen – Autoren zu finden wäre. Auch Schefe behandelt LISP, Prolog und schließlich Smalltalk, er macht das meist mit konventionelleren Beispielen (Suchen, Sortieren) als Stoyan/Görz, die viel mehr auf KI-Techniken eingehen. Auch seine LISP-Beispiele erfordern oftmals eine sehr moderne Implementation. Schefe bemüht sich ebenfalls um eine integrative Darstellung der gegenwärtigen Strömungen in der Informatik, und vielleicht ist ihm das ja auch gelungen, aber es ist halt so furchtbar zäh zu lesen. . .

Das dritte deutsche LISP-Buch – Ch. Hamann hat es verfasst – ist , weil verständlich geschrieben, noch am ehesten anfängertauglich, wenn auch nur eingeschränkt. Die Einschränkungen ergeben sich einmal deswegen, weil es mit Interlisp arbeitet, welches nur auf Großrechnern zu haben ist und einige persönliche Eigenheiten aufweist, die LISPs für kleine Maschinen nicht unbedingt haben und um die man herumarbeiten muß (aber dazu darf man kein Anfänger mehr sein).

Dann besteht der eigentliche Einführungsteil aus äußerst mageren 30 Buchseiten, ehe der Autor mit seinem einen Großbeispiel loslegt (zu dem komme ich gleich). Da LISP ja nun radikal anders als andere Sprachen ist, reichen diese 30 Seiten beileibe nicht aus, um sich mit der andersartigen Denkweise vertraut zu machen.

Drittens wird in diesem Buch zwar in LISP programmiert, nicht jedoch im LISP-Stil. Der Autor benutzt ausgiebig das von Theoretikern verteufelte PROG zusammen mit SETQ (der seiteneffektbehafteten und deswegen üblen Wertzuweisung) und programmiert so in einer Sprache von heute mit dem Stil von gestern: Rekursion scheint ihm ein Greuel. BASIC- Hacker werden sich darin schnell zurechtfinden, aber das ist, finde ich, nicht der Sinn der Sache: man lernt ja nicht eine Sprache wie LISP, um damit munter die Sünden der Vergangenheit weiterpraktizieren zu können!

Nach den ersten 30 Seiten Einführung kommt dann das eben angesprochen Großbeispiel (auf 300 Seiten). C. Hamann zeigt, wie man eines der Lieblingsspielzeuge der KI programmiert: die Blockwelt. Dabei manipuliert ein intelligenter Roboter unterschiedlich geformte Bauklötzchen, die er übereinanderstapeln, verschieben, in Kisten stecken etc. kann. Das ist ganz lustig, bloß: es handelt sich dabei um ein breitgewalztes Übungsbeispiel aus Winston/Horn – siehe unten –, das diese beiden Autoren in einem Kapitel (durchaus verständlich; das geht) abhandeln.

Ich finde, das ist zu wenig. Zwar kann man bei Hamann sehen, wie in LISP ein größeres Programmierprojekt durchgezogen wird. Aber man sieht nicht, wie das ein LISP-Programmierer tun würde, der auf der Höhe der Zeit ist und über die fortschrittlichen Techniken der LISP- Programmierung verfügt; deshalb ist der Wert des Buches zweifelhaft.

## . . . aber gehaltvoll

Eines möchte ich bei all der eben geäußerten Kritik doch festhalten. Inhatlich verfügen alle drei Bücher (meiner Meinung nach absteigend in der hier aufgeführeten Reihenfolge) über beträchtlichen Tiefgang. Das läßt sich sich keinesfalls vergleichen mit den seichten Produkten, die von manchen Computer-Verlagen den Hobbyisten für teures Geld angedreht werden. Die drei obigen Bücher fallen da in eine ganz andere Kategorie: die Leute wissen allesamt, wovon sie reden.

Aber sie können das nicht vermitteln. Ich würde ja nicht so rummäkeln, wenn es nicht auch glänzende Beispiele gäbe dafür, wie man beides unter einen Hut bekommen kann: Fachliche Kompetenz und packenden, sowohl didaktischen als auch eben nicht in den höheren Blödelton (den man leider auch allzuoft antrifft) verfallenden Schreibstil.

Womit wir bei den Amis angelangt wären. Die hier vorgestellten amerikanischen LISP-Bücher sind allesamt so gut, daß Du es bereuen wirst, lieber Leser, deine Englich-Hausaufgaben immer bloß abgeschrieben zu haben, weil Du sie jetzt nicht lesen kannst!

Anders als bei der Besprechung der deutschen Bücher sagt die Reihenfolge der Vorstellung nichts über die Qualität aus. Jedes von ihnen ist überzeugend.

Das Buch von T. Hasemer ist besonders für die interessant, deren LISP nicht alle Raffinessen der moderneren Implementationen aufweist. T. Hasemer weiß um die Nöte des Anfängers und führt ihn sehr einfühlsam in die zu Beginn befremdlich erscheinende Denkweise von LISP ein. Dann weiß er aber auch um die Nöte der Besitzer kleiner Maschinen, die nicht gerade mit einem Megabyte Speicher gesegnet sind. Das ist bei LISP ein Problem, denn die Sprache ist speicherplatzgefräßig. Deshalb werden viele Leser die Tips zu Beginn und im Anhang des Buches begrüßen, die der Autor zur Auswahl eines LISPs für kleine Maschinen zu geben hat.

Wie es bei einem LISP-Buch naheliegt, verfolgt der Autor eine Gliederung in zwei Teile: die reine LISP-Einführung, in der die Grundlagen gelegt werden, und eine Vertiefung, in der Beispiele aus der KI herangezogen werden. Die LISP-Einführung beinhaltet auch einen Teil, in dem die Speicherorganisation von LISP besprochen wird; für das Verständnis einiger 'chirurgischer' Funktionen (RPLACA, RPLACD) ist das wichtig. Außerdem befriedigt es die Neugier derer, die wissen wollen, wie's gemacht wird (denen lege ich jedoch ganz besonders das Buch von J. Allen ans Herz).

## Die Amerikaner: Mit einem Augenzwinkern

Ein Kapitel habe ich in diesem Teil besonders interessant gefunden. Für einige – aber, zum Lob des Autors muß man es erwähnen: ausgesucht wenige – Beispiele benutzt T. Hasemer das PROG-Feature (das ich ja bereits bei C. Hamann gegeißelt habe). Nun gibt es LISPs, deren Entwickler darauf verzichtet haben, diese umstrittene Funktion in den Sprachumfang mit aufzunehmen (muLISP ist dafür ein Beispiel). Doch nicht verzweifeln; der Autor zeigt nämlich, wie man mit den angestammten LISP- Mitteln (Rekursion und Lambda-Konversion) eine PROG-Funktion selberbauen kann! Das gewährt nicht nur wichtige Einblicke in die Arbeitsweise des Interpreters, sondern macht auch klar, daß rekursives Programmieren mächtiger als das übliche sequentielle ist (denn in diesem kann man jenes nachbauen, umgekehrt aber nicht, wenn Sie verstehen, was ich meine. . .)

Der zweite Teil bringt die KI-Beispiele: ein Datenbasis-Programm, die unvermeidliche Eliza und schließlich ein Produktionssystem. Hasemer ist so ehrlich, darauf hinzuweisen, daß auf einem Spielzeugrechner natürlich keine echten KI- Programme möglich sind. Aber die Beispiele, die er bringt, bedienen sich der Techniken, die auch KI- Profis anwenden, und sie machen obendrein Spaß.

Das erste Beispiel demonstriert dies gleich augenfällig. Hier wird ein semantisches Netz aufgebaut, in dem beliebige Fakten gespeichert und auf vielfältige Weise wieder abgefragt werden können. Solche Dinge realisiert der KI-Profi, indem er sich (in LISP) eine eigene Spezialsprache zimmert, mit der sich das Problem besser angehen läßt. Auch Hasemer verfolgt diesen Trick; zusammen mit dem Leser entwickelt er die Sprache LSOLO, baut also einen Interpreter – so daß Sie somit auch die Grundzüge des Interpreterbaus mitbekommen.

Das letzte Beispiel (Produktionssysteme) ist recht aktuell; denn dies ist die Technik, die hinter den vieldiskutierten Expertensystemen steckt. In Hasemers Beispiel ist alles drin: eine Wissensbasis mit Fakten und Regeln, ein Regel-Interpreter ( Inference Engine nennt man das auf Fremdländisch), der mit symbolischer Mustererkennung arbeitet (Pattern Matching) und schließlich die Kontrollstrukturen zur Steuerung des Interpreters.

Daß ich das Buch für sehr gelungen halte, habe ich schon gesagt. Dazu trägt auch bei, daß es Übungsaufgaben mit gut kommentierten Lösungen enthält; für das Selbststudium ist das arg wichtig. Im Anhang gibt der Autor die Definition jeder verwendeten Funktion an; wenn also der eigene LISP-Dialekt von dem im Buch zugrundegelegten abweicht, dann kann man sich hier mit Hasemer synchronisieren.

Als einziger Kritikpunkt (dafür werde ich ja schließlich bezahlt) fällt mir höchstens ein, daß das Kapitel über den Selbstbau von PROG etwas früh (bereits auf Seite 50) kommt. Die Materie ist nicht trivial, und an dieser Stelle muß sich ein Anfänger schon abmühen, um alles restlos zu verstehen. Aber dabei hilft ihm die aus-

gezeichnete Darstellung; sollte er es nicht gleich auf Anhieb kapieren, so macht's auch nichts. Keines der Kapitel hängt vom Verständnis dieses Abschnitts ab.

Fazit: uneingeschränkt empfehlenswert, besonders dann, wenn man seine LISP-Maschinen für 250 Tsd. Dollar grad eben verliehen hat und so auf den alten C64 angewiesen ist!

DIE (mit großen Buchstaben!) LISP-Einführung schlechthin ist jedoch immer noch der Winston/Horn. Das Buch hat Standards gesetzt und noch ist kein anderes in Sicht, das ihm den Rang streitig machen könnte. Einer der Autoren, Patrick Henry Winston, leitet das KI-Laboratorium des MIT (Massachusetts Institute of Technology); beide Institutionen sind so berühmt, daß mir bereits beim Schreiben vor Ehrfurcht die Finger zittern! Spaß beiseite; das MIT ist wirklich hochberühmt (Technologie-Hochburg und so), und das zurecht. Unter anderem verdankt ihm die Menschheit ja die Erfindung von LISP, und alleine das würde schon ausreichen...

## ... wissenschaftlich ganz vorn

Das Buch ist mit einem Augenzwinkern geschrieben. Den Autoren macht ihr Thema Spaß und sie wollen, daß dieser Funke auch auf die Leser überspringt. Darum machen sie es ihnen an keiner Stelle schwerer als unbedingt nötig, ohne sie jedoch wie unmündige Idioten zu behandeln. Nichts ist schwieriger, als diese stilistische Leichtigkeit auch bei zunehmend komplexerem Stoff beizubehalten; Winston/Horn gelingt's.

Auch hier liegt wieder die bereits bei Hasemer angetroffene Zweiteilung vor: LISP-Einführung und KI-Einführung. Winston/Horn gehen jedoch weiter, wesentlich weiter, als der andere Autor. Sowohl die Behandlung von LISP als auch die der KI- Techniken ist tiefschürfender; während Hasemer auf eine Hobbyisten- Leserschaft abzielt, schreiben Winston/Horn eben für Informatik-Studenten. Und da mischt sich der erste (und einzige) Wermutstropfen in die Freude über dieses ansonsten ausgezeichnete Produkt. Es ist als Uni-Lehrbuch gedacht und vermittelt deswegen auch all das, was zur Zeit an vorderster Front der LISP- Programmierung - und eben auch im KI-Labor des MIT – zum Handwerkszeug gehört. Es setzt dazu voraus, daß der Leser Zugang zu einer Maschine mit Common LISP hat. Zwar gibt es für IBM-Rechner und kompatible mit mindestens 512 KB eine Common LISP-Implementation (für satte 1800 DM!; vgl. den Test in c't 3/86), doch nur die wenigsten von uns werden diese Voraussetzung erfüllen. So kann man vieles, was die beiden Autoren darstellen, nicht praktisch nachvollziehen, oder man muß um die Common-LISP- Spezifika herumprogrammieren, was Kenntnisse vorausssetzt, die ein Anfänger einfach noch nicht hat.

Was LISP betrifft, so ist im Winston/Horn alles enthalten, was man sich wünschen kann (mit Ausnahme der Details einer LISP-Implementation). Die ersten 9 Kapitel machen den Leser mit der 'traditionellen' und der modernen Technik der LISP-Programmierung (die sich von der traditionellen vor allem durch den vermehrten Einsatz von Macros und den Gebrauch von anderen als den angestammten Datenstrukturen – den Listen – unterscheidet). Damit ist ein solides Fundament für die Auseinandersetzung mit der KI gelegt.

Die elf KI-spezifischen Kapitel behandeln unter anderem die Blockwelt (vgl. das Buch von C. Hamann), die Techniken der Syntaxanalyse bei natürlicher Sprache mithilfe der ATNs (das steht für Augmented Transition Network ) und die Optimierung dieser Verfahren durch den Bau spezieller ATN-Compiler, die Prinzipien der Expertensysteme, die Implementation spezieller Sprachen zur Wissensrepräsentation (Regel- und Frame-Systeme), das objektzentrierte Programmieren, den Aufbau des LISP-Interpreters und und und... Zu jedem Kapitel gibt es Aufgaben mit Lösungen.

Manche interessiert neben den Anwendungen von LISP auch, wie ein LISP-Interpreter gebaut wird. Das Buch, von dem alle abschreiben, die etwas über dieses Thema zu sagen haben, ist der Allen. Anders als die anderen Bücher empfehle ich das Buch ausdrücklich nicht für Anfänger; dazu ist es zu schwer. Aber nicht wegen seines Stils: auch Allen kann schreiben. Vielmehr setzt es eine Stufe höher an, als dies LISP-Einführungen tun. Man könnte es nennen Wie baue ich eine LISP-Maschine? und was es sonst noch Wissenswertes in der Informatik gibt...

Auch Winston/Horn, Hasemer und Stoyan Görz informieren über das Innenleben des LISP- Interpreters, indem sie die dafür gängige Methode anwenden: sie programmieren in LISP einen LISP-Interpreter. Das ist einfach, denn LISP verfügt schon von Haus aus über Datenstrukturen, die hinreichend mächtig und flexibel

für ein solches Unterfangen sind, eben die Listen. Aber wer die nackte Maschine vor sich hat, die nichts als Bytes kennt und daraus ein LISP hochziehen soll, der steht vor ganz anderen Problemen. Diese Probleme löst Allen; da zur Implementation von LISP einige trickreiche Datenstrukturen samt zugehöriger Algorithmen nötig sind, wird dieses Thema ausführlichst behandelt. Somit kann man das Buch auch als Einführung in das Gebiet Algorithmen und Datenstrukturen lesen, wenn auch das Lieblingsthema dieses Informatik-Bereichs, die Sortiererei, nicht allzu breiten Raum einnimmt.

## Fazit

Wer einen LISP-Interpreter bauen will, der auf der Höhe der Zeit ist, der kommt ohne den Allen nicht aus. Er muß sich aber klar darüber sein, daß er für das Buch (Import!) tief in die Tasche greifen muß.

Wer als Besitzer einer kleinen Maschine ernsthaft LISP betreiben und sich in die KI einarbeiten will, der sollte sich zuerst das Buch von Hasemer durchlesen. Wenn er dann fit genug in LISP ist, kann er zur Vertiefung und Ausweitung seiner Kenntnisse den Winston/Horn durcharbeiten. Es sollte dann auch möglich sein, den eigenen LISP- Dialekt an den im Buch verwendeten anzupassen, wo dies nötig ist. Es ist ja gerade eine der wundersamen Eigenschaften von LISP, daß sich andere Sprachen und deswegen auch andere LISP-Dialekte in ihm nachbilden lassen! Allerdings sehe ich ein, daß bei amerikanischer Literatur zweierlei Hindernisse zu überwinden sind: die Sprachbarrierere und das Bestreben mancher Buch-Importeure, sehr schnell sehr reich zu werden. Letzteres kann man umgehen, wenn man in der Buchhandlung darauf besteht, direkt bei den europäischen Verlagsvertretungen zu beziehen! Wem diese Hürden zu hoch sind, der sollte sich den Stoyan/Görz zulegen, in dem man viele der hier behandelten Themen wiederfindet.

**Die besprochenen Bücher:**

H. Stoyan, G. Görz: LISP. Eine Einführung in die Programmierung. Berlin, 1984. Springer Verlag. 358 Seiten, DM 49.- ISBN 3-540-13158-2.

P. Schefe: Informatik - Eine konstruktive Einführung. LISP, Prolog und andere Konzepte der Programmierung. Mannheim, 1985. Bibliographisches Institut. 403 Seiten, DM 38.00. ISBN 3-411-03107-7.

C-M. Hamann: Einführung in das Programmieren in LISP. Berlin, 1985. De Gruyter. 343 Seiten, DM 45.- ISBN 3-11-0103257.

T. Hasemer: A Beginner's Guide to LISP. Reading, 1984. Addison-Wesley. 256 Seiten. $17.60. ISBN 0-201-14634-7.

P. H. Winston, B.K.P. Horn: LISP. Reading, 1984. Addison-Wesley. 434 Seiten. $31.70. ISBN 0-201-08372-8.

J. Allen: Anatomy of LISP. New York, 1978. McGraw - Hill. 446 Seiten. ISBN 0-07-001115-X.

*Peter Rosenbeck*

# Pferderennbahn

Ein richtiges Pferd kriegt man heutzutage nur noch auf dem Dorf (oder dem Fernsehschirm) zu sehen. Außer, man geht zum Pferderennen. Und schon sind wir beim Thema. Dieser Sport, bei dem die Pferde in sehr großem Maße beteiligt sind, wird von diesem Spiel imitiert.

Dabei geht es darum, daß Sie eine solche Rennveranstaltung besuchen, und zu diesem Zweck bis zu 500 DM einsetzen wollen.

Nachdem Sie das Programm geladen und gestartet haben, geben Sie die Anzahl der Mitspieler (maximal sechs) und ihre Namen ein. Auf dem folgenden Bild sehen Sie drei Spalten. In der Spalte Platz/Sieg, tragen Sie ein P für Plazierung oder ein S für Sieg ein. Dabei sollte bedacht werden, daß Plazierung (Sie bekommen das Doppelte Ihres Einsatzes zurück, falls sich das von Ihnen gesetzte Pferd unter den ersten drei befindet) die Chancen auf einen Gewinn vergrößert, bei Sieg aber der mögliche Gewinn das Sechfache Ihres Einsatzes beträgt.

Auf welches Pferd, ob auf Platz oder Sieg und wieviel Sie wetten, bleibt ganz Ihnen und Ihren Mitspielern überlassen. Ansonsten erklärt sich das Spiel von selbst.

## Redaktionelle Mitteilung

Das Spiel "Der große Preis" mußte auf eine der nächsten Ausgaben verschoben werden, da noch nicht alle Fragen im Zusammenhang mit dem Urheberrecht des ZDF vollständig geklärt sind. (d.Red.)

## Mathe mit Nico

Diesmal holt sich Nico seine Zinsen von der Bank ab. Dahinter stehen natürlich die neuen mathematischen Aufgaben, die sich mit der Zinsrechnung beschäftigen.

Selbstverständlich steht Ihnen auch diesmal die Rechenseite zur Verfügung, auf der Sie alle Rechen- Operationen ausführen können, die Sie auch im Direkt-Modus zur Verfügung haben. Diese Rechenseite können Sie immer dann aufrufen, wenn Sie zu einer Eingabe aufgefordert werden. Der Aufruf erfolgt mit der Taste R und RETURN.

## Hilfsprogramm

### Kalkultool

Das Tool erlaubt die Einbindung von Formeln innerhalb von BASIC-Programmen. Sie haben somit die Möglichkeit, über den GET-, INPUT- oder einen INKEY-Befehl vom Anwender Ihres Programms einen Rechenvorgang oder eine komplette Formel eingeben zu lassen. Diese Zeichenfolge übergeben Sie im Programm an eine String-Variable. Der SYS-Aufruf des Rechen-Tools weist das Ergebnis dann einer Variablen zu. Auf diesem Weg kann das Ergebnis entweder auf dem Bildschirm ausgegeben oder innerhalb des Programm, beispielsweise in grafischen Darstellungen, weiterverarbeitet werden.

Aufruf:
SYS 51200,A$,ER,FE
A$: Sollte die eingegebene Formel enthalten.
ER: Hier steht das Ergebnis als Real-Zahl
FE: Enthält nach der Rechnung eine Fehlerzahl:
11 Syntax-Fehler, fehlerhafte Eingabe
14 Falscher Wert
15 Overflow, Ergebnis zu groß
16 Speicher Überlauf
18 Falscher Index
20 Teilen durch Null
22 falscher Zahlentyp
25 Formel zu komplex
27 undefinierte Funktion

Beispiel-Programm:

```
100 REM BEISPIEL
110 INPUT "Formel ";A$
120 SYS 51200,A$,ER,FE
130 IF FE<>0 THEN PRINT "Fehlernummer:";FE:GTO110
140 PRINT "Ergebnis ";ER
150 END
```

Sie können das Tool in zwei Versionen innerhalb des Moduls mit CTRL+S abspeichern. Die Adresse des SYS-Aufrufs ist in der Version am BASIC-Anfang natürlich nicht mehr 51200, sondern **-2761**

## ChipList

Für Hardware-Konstrukteure bieten wir ein Programm, mit dem Sie sich die Pin-Belegung und Funktionsbeschreibung einzelner integrierten Schaltkreise anschauen können. Das Programm enthält einige typische Standardbausteine der 7400er- und 4000er-Serie, sogenannter TTL-Bausteine (Transitor Transistor Logik).
Das Programm ist menuegesteuert und erlaubt die direkte Anwahl eines speziellen Typs, sowie das Durchblättern der gesamten Liste. Außerdem können Sie nach bestimmten Merkmalen suchen. Sie brauchen nur ein Stichwort oder eine Zeichenkette einzugeben und erhalten eine Liste aller IC-Typen, die das gesuchte Kriterium erfüllen.

Menueauswahl:
- A Anzeige einzelner IC-Typen
- S Suchen über Eingabe eines Stichwortes
- D Schaltet den Drucker ein oder aus
- ← Rückkehr ins Menue innerhalb einer Auswahl

Untermenue:
Auf jedem angezeigten 'Blatt' können Sie folgende Punkte aufrufen:
- T Direktauswahl eines bestimmten IC-Typs
- + Vorwärtsblättern
- - Rückwärtsblättern
- ← Zurück ins Hauptmenue

Haben Sie über "D" den Drucker eingeschaltet, wird die Anzeige vom Bildschirm auf den Drucker umgelenkt. Sie können nach Anwahl von "D" folgenden Parameter zur Steuerung des Druckers angeben:
- Geräteadresse
"4", "5" oder "6" (6 = Plotter)

In Ihrem Drucker- oder Plotterhandbuch finden Sie die entsprechenden Hinweise.

Da das Programm in BASIC geschrieben wurde, können erfahrene Programmierer weitere Kenndaten in die DATA-Zeilen aufnehmen und so die Liste erweitern. Außerdem kann bei Bedarf die Druckersteuerung ab Zeile 1740 geändert werden. Im Programm wird der Bildschirm ausgelesen und auf den Drucker geschickt. Bei der Darstellung der Pin-Belegung wird mit CHR$(8) und anschließendem CHR$(15) vor der nächsten

Zeile, der Zeilenabstand auf 7/72 Inch verringert.
Denken Sie bei Änderungen daran, daß der BASIC-Anfang verschoben ist! Vor dem Programm liegen PRINTAT und INKEY. Sie müssen nach einer Änderung mit POKE 44,8 und POKE 43,1 den BASIC-Anfang wieder zurücksetzen. Das eigentliche Programm kann nur nach RUN gelistet werden. Logisch oder?

# 64er Tips

## Kassetten-Tips

Viele Leser benutzen, besonders wenn sie zu den Einsteigern gehören, die Kassetten-Ausgabe unseres Magazins. Selbst wer bereits über ein Floppy-Laufwerk verfügt, mag im einen oder anderen Fall trotzdem auf die Kassette zurückgreifen, um beispielweise Sicherheitskopien anzulegen, zumal man in Verbindung mit SuperTape vernünftige Lade- und Speicherzeiten erreichen kann.

### Wie werden die Daten auf Band gespeichert?

Leider ist es nicht ohne weiteres möglich, die Inhalte eines Datenbandes sichtbar zu machen. Beim Schreiben auf Kassette sendet der Rechner über die Leitung zur Datasette einzelnen Impulse unterschiedlicher Länge. Diese werden anschließend von der Elektronik der Datasette in magnetische Schwingungen umgewandelt, die auf der Magnetschicht des Bandmaterials aufgezeichnet werden. Hierbei werden die einzelnen Bits nach folgendem etwas kompliziertem Muster umgewandelt:
Der Rechner erzeugt drei verschieden lange Signale. Jeweils vier solcher Signale werden für eine Information verwendet:

..: 2mal kurz, –: 2mal mittel, = =;2mal lang
. . . . Endemarkierung (4mal)
..– Null (1 BIT = 0)
–.. Eins (1 BIT = 1)
= =– vor jedem BYTE (8 BIT)
= =.. vor einem Markierungsblock

Beispiel:
Der Buchstabe "I" wird auf Band geschrieben.
"I" = CHR$(73)

= =– –.. ..– ..– –.. ..– ..– –.. ..– –.. ..–

BYTE 1 0 0 1 0 0 1 0 1
BIT 0. 1. 2. 3. 4. 5. 6. 7. Parity

Sie können sich vorstellen, wie wichtig dabei die gleichmäßige Bewegung des Bandes am Tonkopf ist. Schwankungen haben unweigerlich zur Folge, daß die Elektronik im Gerät die Informationen auf dem Band falsch interpretiert.

### Wie sendet der Rechner die Daten?

Die Frequenz, die Zahl der Schwingungen pro Sekunde, wird vom Rechner durch die eingebauten Timer-Bausteine festgelegt. Genau an dieser Stelle greifen Verfahren ein, die die etwas müde Lade- oder Speicherzeit des normalen Commodore-Betriebs aus den Träumen reißen. SuperTape erhöht die Frequenz so weit, daß sogar die erheblich schnellere Floppy überholt werden kann. So erklärt sich auch, warum Kassetten, die mit SuperTape beschrieben wurden, mit dem normalen Commodore-Format nicht mehr gelesen werden können. Die Datasette überträgt zwar Daten, aber der Rechner schlägt einen anderen Takt, sodaß keine Verständigung zustande kommt.

Verantwortlich für die Steuerung ist ein Baustein im Rechner, der CIA1 ab Adresse $DC00 (56320). Dieser Baustein regelt mit seinem Äquivalent CIA2 die Ein-Ausgabevorgänge und die Tastaturabfrage im Rechner. Der 'Complex Interface Adapter' betreut auch die RS232-Schnittstelle am User-Port. Beide Bausteine enthalten zwei unabhängige Timer und eine 24-Stunden Uhr, die innerhalb diese Magazins im Modul c't-Uhr zu Demonstrationszwecken benutzt wird. Der CIA1 ist für den Kassetten-Betrieb, Tastaturabfrage, Joysticks und Paddles zuständig. CIA2 wird für den RS323-Betrieb genutzt und hält im Rechner Kontakt zum Videoram und zur seriellen Schnittstelle (Floppy oder Drucker).

Diese Bausteine sind gegen Fremdspannungen und Kurzschlüsse sehr empfindlich. Eine elektrostatische Aufladung an den Händen über den Joystickport auf den CIA gebracht reicht! Dummerweise liegt der Aus-Schalter direkt neben den Anschlußstiften der Joystickports. Sollten Sie ausversehen CIA1 erwischt haben, bemerken Sie das am teilweisen Ausfall der Tastatur oder des Kassetten-Betriebs. CIA2 verweigert bei Defekten die Zusammenarbeit mit Drucker oder Floppy. Manchmal bemerkt man solche Störungen jedoch erst bei Benutzung des User-Ports. Mir war es nach verschiedenen Experimenten mit den Paddle-Anschlüssen am Joystick-Port gelungen, CIA1 kurzzuschließen. Anschließende Joystick-Kunststücke zeigten keine Wirkung

mehr. Beim Rechner eines Freundes hüllte sich der angeschlossene Akustikkoppler in Schweigen. Durch Messungen am Userport wurde CIA2 geliefert. Einen neuen Baustein aufzutreiben, ist ausgesprochen schwierig, ganz zu schweigen von der nervenaufreibenden Auslöterei der Chips. Nur wer vom Schicksal besonders begünstigt wurde, finden gesockelte CIAs in seinem Rechner. Einem Laien kann da nur ein guter Fachmann helfen!

**Woher weiß der Rechner, was er liest?**
Von den Konstrukteuren des C64 wurde zur Verständigung mit der Datasette echte Kopfarbeit geleistet.

In den RAMs der Rechner wurde ein sogenannter Puffer direkt vor dem Bildschirmspeicher freigehalten. In diesen wird der Header (Kopf) einer jeden Datei beim Kassetten-Betrieb abgelegt. Dieser 191 Byte lange Block enthält Informationen über die Datenart, ob Maschinen- oder BASIC-Programm oder sequentielle Datei, den Namen und die Start- und Endadresse. Diese Informationen wurden beispielsweise bei den Programmen Kassetten-Direktory (4/85) und TapeCopy (7/85) ausgenutzt. Da der Name höchstens 16 Zeichen lang sein darf, kommen insgesamt 20 Bytes zusammen, der Rest bleibt im Header leer.

Wer mit einem Monitor umzugehen weiß, kann sich das anschauen. Wird nach der Meldung 'Found' der Ladevorgang abgebrochen, findet man ab der Adresse 828 ($033C) den jeweiligen Block.

**Headerformat**

| Byte Nr | Inhalt |
|---|---|
| 0 | Headertyp |
| 1+2 | Start-Adresse |
| 3+4 | Endadresse |
| 5-20 | Dateiname |

Im Beiheft INPUT 64 4/85 finden Sie die Beschreibung des Headers in Tabellenform.
SuperTape verwendet ebenfalls einen Header der gleichen Länge, jedoch sind die Daten hier in der Reihenfolge Name, Sekundäradresse, Startadresse, Filelänge abgelegt. In INPUT 64 4/85 ist auch SuperTape beschrieben.

**Wie arbeitet der Rechner mit Kassetten?**
Im Kassetten-Betrieb nimmt Ihnen der Rechner einige Arbeit ab. Das Betriebssystem teilt dem CIA1 mit, ob Daten gelesen oder geschrieben werden sollen. Dann wird der Timer (eine Art Stopuhr) gesetzt und dem CIA-Chip mitgeteilt, wo die Daten von Kassette hereinkommen und der Timer gestartet.

**Wie werden Fehler festgestellt?**
Während des Lesens werden die gelesenen Informationen gegengeprüft. Bemerkt das Betriebssystem einen Fehler, erfahren Sie es durch den berüchtigten Load Error . Das Betriebssystem benutzt die Adresse 144 ($90), das sogenannte Statusbyte "ST", um genauer mitzuteilen, welcher Zustand bei der Kassettenoperation auftrat. Diese Adresse kann von BASIC direkt über PRINT ST angesprochen werden. Für Maschinen-Programmierer steht die Routine $FFB7 (READST) zur Verfügung, die den Wert im Akku übergibt.

| ST | Bedeutung | BIT |
|---|---|---|
| 4 | Zu kurzer Block | 2 |
| 8 | Zu langer Block | 3 |
| 16 | Lesefehler | 4 |
| 32 | Prüfsummenfehler | 5 |
| 64 | Dateiende | 6 |
| 128 | Bandende (EOT) | 7 |

Treten mehrere Fehler gleichzeitig auf, wird die Summe der einzelnen Werte in ST abgelegt. BIT-Kundige werden bemerken, daß die Werte genau der Potenz-Reihe von Zwei entsprechen. Die Blocklänge muß, wie Sie inzwischen wissen, immer genau 192 Zeichen betragen.
Lesefehler erkennt das System dadurch, daß die Datenblöcke zweimal hintereinander auf das Band geschrieben werden. Tritt beim Lesen des zweiten Blocks eine Abweichung gegenüber dem ersten auf, wird ein Fehler festgestellt.
Die einzelnen Bytes werden beim Schreiben auf Band der Reihe nach zu einer Prüfsumme verrechnet - für ganz Neugierige: über ein EXOR, was logisch 'exkulsiv Oder' entspricht. Das Ergebnis ist 1 (wahr), wenn die beiden verglichenen Bits ungleich sind, sonst ist das Ergebnis 0.

| AND | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| OR | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| EXOR | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

(Für Tüftler: EXOR läßt sich durch eine Verknüpfung von AND, OR und NOT ersetzen, in einer BASIC-Zeile!)

Beim Lesen wird die Prüfsumme ermittelt und mit der vom Band gelesenen verglichen. So lassen sich Übertragungsfehler feststellen.
Bei sequentiellen Dateien (vergleiche auch IN-

PUT 3/86) wird über ST = 64 festgestellt, daß das Ende der Datei erreicht wurde.

Ein Bandende wird leider nur dann erkannt, wenn tatsächlich ein EOT-Block auf Band geschrieben wurde.

In den 64er Tips finden Sie ein paar interessante anschauliche Hinweise und Beispiele für den Umgang mit Kassetten und natürlich wieder ein kleines Beispielprogramm zum Abspeichern (mit CTRL+S) und Rumprobieren. Solange Sie nicht gerade die neusten Top-Hits der Musikparade auf Kassette eingelegt haben, dürften Sie einige interessante Daten wiederfinden.

---

### Simulation auf dem C64

## Life

Vor mehr als 15 Jahren entwickelte John Horton Conway das Simulationsspiel Life. Die Grundidee dieser Simulation ist schnell beschrieben: Als Spielbereich dient ein zweidimensionales Spielfeld, das aus quadratischen Gitterplätzen besteht. Jedes dieser Plätze kann von einer Zelle belegt werden. Leben bzw sterben dieser Kolonie wird durch bestimmte Regeln bestimmt. Ausschlaggebend ist die Anzahl der maximal acht Nachbarzellen. Hat eine Zelle mehr als drei, oder weniger als zwei Nachbarn, so stirbt sie an Überbevölkerung bzw. an Isolation. Ist ein freier Platz von genau drei Zellen umgeben, so wird eine neue Zelle geboren. Es ist wichtig das diese Regeln gleichzeitig angewendet werden. Die aus einer festzulegendén Anfangswelt entstehenden Zellmuster überraschen in ihrer Entwicklung. Manche Kolonien breiten sich rapide aus, andere sterben ab und wieder andere wiederholen bestimmte Zyklen.

Inzwischen gibt es auf den unterschiedlichsten Rechnern Implementationen. Auch für den 64er ist dieses Programm nicht neu. Wir können Ihnen aber ein Programm anbieten, das zum einen den Generationsaufbau sehr schnell berechnet, und zum anderen Ihnen die Möglichkeit einräumt, die Regeln zu verändern. Innerhalb von INPUT 64 sind nur die Kassetten- und Disketten-Operationen abgeschaltet. Dafür können Sie aber die HardCopy-Funktion mit CTRL und B benutzen. Wenn Sie das Programm mit CTRL und S auf Ihren Datenträger überspielt haben, sind diese Optionen selbstverständlich aktiv. Wenn Sie mit SuperTape arbeiten wollen, müssen Sie SuperTape vorher laden.

Das Programm ist menuegesteuert und kann mit der Tastatur und/oder Joystick (Port 2) bedient werden. Die Joystick-Funktionen werden durch die CRSR-Tasten und RETURN ersetzt. Sie werden erkennen, daß Sie den Generationen unterschiedliche Farben zuordnen können, einen Zwischenspeicher für eine Kolonie benutzen und vor allem die Regeln nach denen sich die Generationen entwickeln, verändern können. Es folgen die Befehle im Editor-Modus:

| Taste | Befehl |
|---|---|
| A | Austausch (Speicher/Bildschirm) |
| S | Bildschirm in Speicher |
| Z | Zurückholen aus Speicher |
| E | Entwicklung starten |
| M | Menü anwählen |
| RUN/STOP | Abbruch der Entwicklung |
| CRSR | Bewegung auf dem Bildschirm |
| RETURN | Setzen oder Loeschen einer Zelle |

Wir können Ihnen bei der Entwicklung Ihrer eigenen Anfangswelt und im fortgeschritten Stadium bei der Festlegung neuer Wachstums-Gesetze nur viel Spaß wünschen.

## Rätselecke

# Nur eine 10stellige Zahl...

### Auflösung des Rätsels aus INPUT 64 1/86

Wie könnt Ihr mir nur so viele schlaflose Nächte bereiten..., und Wielange soll der Rechner denn noch laufen; ich geb' auf..., aber auch Für diese einfache Aufgabe habe ich den 64er gar nicht erst eingeschaltet. Die Lösung ist...

Dieses sind nur einige der vielen Kommentare auf den Postkarten und Briefen, die uns als Reaktion auf die letzte Rätselaufgabe erreichten. Bevor wir nun zur Auflösung kommen, noch eine Anmerkung: Wir erhielten diesmal mehrere Doppel- und Dreifach - Einsendungen. Ein INPUT-Leser hat sich sogar die Mühe gemacht, 20 Postkarten mit derselben Lösungs-Zahl einzusenden. Die Arbeit war in zweifacher Hinsicht umsonst. Erstens war die Zahl falsch, und zweitens kann jeder nur einmal an der Verlosung teilnehmen; wir hätten sowieso 19 Karten beiseite gelegt.

## Die Aufgabenstellung

Sie sollten eine (die) 10stellige Zahl finden, bei der zwei Bedingungen erfüllt sind:

1. Jede Ziffer darf nur einmal vorkommen.

2. Die Zahl, bestehend aus den ersten beiden Ziffern, soll durch Zwei teilbar sein, die Zahl, bestehend aus den ersten drei Ziffern durch Drei...und die ganze Zahl durch Zehn.

Daß es nur eine Lösungs-Zahl gibt, hatten wir Ihnen auch noch gesagt.

## Ein Lösungsweg

Die einfachste Möglichkeit, die gesuchte Zahl zu finden, wäre es, in einer Schleife alle 10stelligen Zahlen auf die zweite Bedingung hin zu überprüfen. Wenn Sie so vorgegangen sind, werden Sie wohl dem zweiten Leserbrief-Auszug aus vollem Herzen zustimmen. Sinnvoller ist es, vor der Überprüfung, die 10stelligen Zahlen einzugrenzen. Fangen wir damit doch einfach mal an:

1. Da die 10stellige Zahl durch Zehn teilbar sein soll, muß die zehnte Stelle eine Null sein.

2. Die Zahl aus den ersten fünf Ziffern soll durch Fünf teilbar sein. Eine Zahl ist nur dann durch Fünf teilbar, wenn die letzte Ziffer entweder eine Null oder eine Fünf ist. Die Null haben wir schon verbraucht , also bleib für die fünfte Stelle nur die Fünf übrig.

3. Eine Zahl ist durch Neun teilbar, wenn sich die Quersumme (die Addition aller Ziffern) durch Neun teilen läßt. Da die Zahl bestehend aus den ersten neun Ziffern als Quersumme in jedem Fall 45 hat, kann es Ihnen egal sein, welchen Wert die neunte Ziffer hat; durch Neun ist die Zahl in jedem Fall teilbar.

4. Die End-Ziffern, die durch eine gerade Zahl geteilt werden sollen (die zweite, vierte, sechste und achte Stelle), müssen mindestens durch zwei teilbar sein, und können somit nur Zwei, Vier, Sechs oder Acht lauten. (Auch hier darf die Null nicht mehr vorkommen.)

5. Nun bleiben für die ungeraden Teiler nur ungeraden End-Ziffern übrig; nämlich die Eins, Drei, Sieben und Neun. (Die Fünf gibt es ja nicht mehr.)

Wenn Sie diese Erkenntnisse nun in eine Tabelle eintragen, erhalten Sie folgende theoretischen Möglichkeiten:

| Stelle | Ziffer | Stelle | Ziffer |
|---|---|---|---|
| 1 | 1,3,7,9 | 6 | 2,4,6,8 |
| 2 | 2,4,6,8 | 7 | 1,3,7,9 |
| 3 | 1,3,7,9 | 8 | 2,4,6,8 |
| 4 | 2,4,6,8 | 9 | beliebig |
| 5 | 5 | 10 | 0 |

Was jetzt übrig bleibt ist eine klassische Permutation der theoretischen Ziffern. Da unsere Rätselaufgabe der Ausgabe 3/85 genau dieses mathematische Problem zum Gegenstand hat, können wir diesmal noch keine Musterlösung anbieten. Aufgeschoben ist aber nicht aufgehoben!

## Der Hauptgewinn

Selbstverständlich wollen wir Ihnen aber die gesuchte Zahl nicht vorenthalten; sie lautet 3816547290. Aus dem Berg der vielen hundert richtigen Einsendungen hat eine Glücksfee die Gewinner ermittelt. Den ersten Preis - und damit ein INPUT 64 Jahresabonnement - hat Harald Fleischmann aus Hallstadt gewonnen. Die Gewinner der Buchpreise werden von uns direkt benachrichtigt. Allen Gewinnern unseren herzlichen Glückwunsch. Aber auch bei den vielen anderen Einsendern wollen wir uns für die rege Teilnahme bedanken.

# c't-Uhr

## Betrifft Echtzeituhr in c't 4/86

Hier sollte Ihnen die Software zur Bedienung der Echtzeituhr vorgestellt werden. In allerletzter Minute zeigte sich jedoch, daß die c't-Uhr aufgrund von Hardware-Problemen im C64 nicht mitspielt. Zwar kann sie von einem Maschinen-Programm ohne Schwierigkeiten ausgelesen und gesetzt werden, doch Versuche in BASIC bei eingebauter Uhr sind zum Scheitern verurteilt. Wir wollten Ihnen keine halbe Sache anbieten und beschlossen deshalb, die Veröffentlichung der Software auf die nächste Ausgabe zu verschieben. Schließlich sollten Sie an der Uhr Freude haben und keine bösen Überraschungen erleben müssen.

Das Hauptproblem ist rein technischer Natur. Im Rechner werden die einzelnen Bausteine durch spezielle Impulse angesprochen (Chip-Selects), so auch das BASIC-ROM, über dem laut Vorschlag in c't 5/86 die Echtzeituhr ihren Platz finden soll. Leider sind diese Impulse so unsauber, daß sich die Uhr in den oberen Bereich des BASIC-ROMS einblendet, was dann im entsprechenden Moment zu Fehlfunktionen führt. Dies geschieht bedauerlicherweise auch dann, wenn die Uhr nicht angesprochen wird.

Wir bleiben jedoch am Ball und erwägen, die Uhr zum Zeichensatz-ROM umziehen zu lassen. In der nächsten Ausgabe erfahren Sie mehr.

### Software-Review

# Turbo Nibbler 4.0

Kennen Sie das? Mit einer ungeschickten Handbewegung zerstört man die Orginal-Diskette der dringend benötigten Software. Eine Sicherheitskopie hat man nicht, der Hersteller hat's fachkundig unterbunden. Briefumschlag, defekte Originaldiskette, Anschreiben an die Firma, Scheck über X DM und Briefmarken, alles sachgerecht zusammengepackt und ab in den Briefkasten. Und jetzt noch ein bißchen warten! Die großen inländischen Softwarehäuser liefern in der Regel schnell, doch das dringend erwartete Auslands-Päckchen hängt tagelang beim Zoll. Wer das schon mal mitgemacht hat, weiß wovon ich rede. Das ganze ist nicht nur ärgerlich, sondern oftmals trifft der ersehnte Ersatz gar nicht mehr ein, und mehrere Hundert Mark, die man ehemals auf den Tisch legen mußte, sind futsch.

Damit diese Fälle sich nicht häufen, bietet die niederländische Firma Eurosystems ihr Backup-Programm Turbo Nibbler an. Mit diesem lassen sich von den meisten geschützten Disketten Arbeitskopien herstellen. Die Version 4.0 ist soeben auf dem Markt erschienen und stellt eine deutliche Verbesserung zu den alten Versionen da. Mußte man bei diesen noch mehrere Parameter von Hand wählen, so entfällt dieses jetzt. Bei Bedarf lassen sich die Defaultwerte für Start-/Endtrack, Anzahl der Schreib-/Leseversuche und Halftracks yes/no im Hauptmenue ändern. Ebenso kann das Kopieren mit zwei Laufwerken gewählt werden. Mit der Funktion 'Disk scannen' kann man sich erst einmal einen Überblick verschaffen, wessen Programmieres Kind der Kopierschutz ist. Dabei werden verschiedene Trackparameter angezeigt. Mit ein wenig Übung läßt sich dann bestimmen, wie man die ersten Kopierversuche durchführt. Sie haben richtig gelesen:: Versuche. Denn nicht bei jeder Software erhält man beim ersten Versuch eine lauffähige Kopie. Und es gibt auch einige hartnäckige Fälle, bei denen man nach ein bis zwei Stunden feststellen muß, daß es gar nicht geht. Das heißt nicht, daß der Turbo Nibbler ein schlechtes Produkt ist, sondern mehr ist – systembedingt über den seriellen Bus – nicht machbar. Doch in den meisten Fällen erhält man in wesentlich kürzerer Zeit die nervenschonende Arbeitskopie. Die gängigsten Schutzmethoden wie READ ERRORs, Killertracks, Spuren 36-41, Halftracks oder die sehr aktuellen Fremdformate stellen kein ernstzunehmendes Hindernis mehr dar. Doch sollte auch die notwendige Kritik nicht fehlen:: Die mitgelieferte Anleitung ist eher dürftig, nicht nur, das die benötigte Lupe nicht beiliegt, sondern auch die verwendeten Begriffe bleiben vielfach unklar. Hilfreich für den ungeübten Nutzer wäre mit Sicherheit eine Beispielbeschreibung. Alles in allem bekommt man für 55 DM eine ausgereifte Software geboten, die sich schnell amortisiert. Zwei ältere Versionen werden gratis mitgeliefert, gerade die Version 2.2 bietet sich, da etwas schneller, für den täglichen Gebrauch an. Nach getanem Tagwerk lassen sich noch schnell ein paar Backups ziehen, die dann im Wandtresor nächtigen können.

*Eurosystems*
*Verlengde Parkweg 6*
*6717 GN EDE*
*Holland*

# Hinweise zur Bedienung

Bitte entfernen Sie eventuell vorhandene Steckmodule. Schalten Sie vor dem Laden von INPUT 64 ihren Rechner einmal kurz aus. Geben Sie nun zum Laden der Kassette *LOAD* und *RETURN* oder *SHIFT* und gleichzeitig *RUN/STOP* bzw. der Diskette *LOAD"INPUT*",8,1* und *RETURN* ein. Alles weitere geschieht von selbst.

Nach der Titelgrafik springt das Programm ins Inhaltsverzeichnis des Magazins. Dieses können Sie nun mit der *SPACE* (Leertaste) durchblättern. Mit *RETURN* wird das angezeigte Programm ausgewählt. Im Fenster unten rechts erhalten Kassettenbesitzer weitere Hinweise ("Bitte Band zurückspulen" und so weiter . . .). Haben Sie bei der Auswahl eines Programms eventuell nicht weit genug zurückgespult, und es wurde nicht gefunden, spulen Sie bis zum Bandanfang zurück. Diskettenbesitzer stellen bitte sicher, daß noch die INPUT 64-Diskette eingelegt ist. Auf der 2. Kassettenseite befindet sich eine Sicherheitskopie. Sollten Sie eventuell mit einem der Programme Ladeschwierigkeiten haben, versuchen Sie es auf Seite 2. Führt auch dies nicht zum Erfolg, lesen Sie bitte die entsprechenden Hinweise im Kapitel "Bei Ladeproblemen"!

Neben der Programmauswahl mit *SPACE* und dem Ladebefehl mit *RETURN* (im Inhaltsverzeichnis) werden die übrigen 'System-Befehle' mit der Kombination aus *CTRL*-Taste und einem *Buchstaben* eingegeben. Sie brauchen sich eigentlich nur *CTRL* und *H* zu merken (Aufruf der Hilfsseite), denn dort erscheinen die jeweils möglichen weiteren 'System-Befehle'. Nicht immer sind alle Optionen möglich (eventuell werden Sie zu Beginn des Programms auf Einschränkungen hingewiesen). Hier nun alle INPUT 64-Systembefehle:

*CTRL* und *Q* (ab Ausgabe 3/85)

Sie kürzen die Titelgrafik ab; INPUT 64 geht dann sofort ins Inhaltsverzeichnis.

*CTRL* und *H* (ab Ausgabe 1/85)

Es wird ein Hilfsfenster angezeigt, auf dem alle verfügbaren Befehle aufgeführt sind.

*CTRL* und *I* (ab Ausgabe 1/85)

Sie verlassen das Programm und kehren in das Inhaltsverzeichnis zurück.

*CTRL* und *F* (ab Ausgabe 1/86)

Ändert die Farbe des Bildschirm-Hintergrundes (auch im Inhaltsverzeichnis erreichbar).

*CTRL* und *R* (ab Ausgabe 1/86)

Ändert die Rahmenfarbe (auch im Inhaltsverzeichnis erreichbar).

*CTRL* und *B* (ab Ausgabe 4/85)

Sie erhalten einen Bildschirmausdruck – natürlich nicht von Grafikseiten oder Sprites! Angapaßt ist diese Hardcopy für Commodore-Drucker und kompatible Geräte. Das Programm wählt automatisch die richtige Geräteadresse (4, 5 oder 6) aus.

Fortsetzung Seite 30

## Hinweise für Autoren

Falls Sie uns ein Programm zur Veröffentlichung anbieten wollen, beachten Sie bitte folgende Hinweise: Selbstverständlich können Sie uns Ihr Programm nur anbieten, wenn Sie es selbst erstellt haben und das Programm noch nicht veröffentlicht wurde. Ihr Programm sollte in C-64-BASIC oder in 6502/6510-Assembler geschrieben sein. Als Hilfsmittel können Sie die bisher in INPUT 64 erschienenen Tools (PRINT AT, INKEY, Hiresspeed und die Sprite-Befehle) benutzen, wobei Ihr Programm aber insgesamt nicht länger als 100 Blöcke (25 KByte) sein sollte. Das Programm muß auch ohne Floppy lauffähig sein. Floppy-Betrieb optional ist erlaubt und gewünscht. Es gibt außerdem einige, durch das INPUT 64-Betriebssystem bedingte, programmiertechnische Erfordernisse: 1. Belegen Sie nur den Bereich des normalen BASIC-RAM ($0801-$9FFF) und unter dem BASIC-ROM ($A000-$BFFF). 2. Das Programm muß als BASIC-File zu laden und mit RUN zu starten sein. 3. Die CTRL-Taste darf nicht benutzt werden.

Aber auch wenn Ihr Programm zur Zeit diese Anforderungen nicht erfüllt, sprechen Sie uns ruhig an. Bei ausgefallenen Programmentwicklungen sind wir gerne bereit, bei der Anpassung behilflich zu sein. Senden Sie uns Ihr Programm auf Kassette oder Diskette mit einer Programmbeschreibung und notieren bitte auf allen Einzelteilen Ihren Namen und Ihre Anschrift. Sowohl Auto-Start als auch List-Schutz erschweren uns nur die Arbeit! Wir werden deshalb Programme, deren Analyse absichtlich erschwert wurde, zukünftig ungeprüft zurücksenden.

*CTRL* und *S* (ab Ausgabe 1/85)

Wenn das Programm zum Sichern vorgesehen ist, erscheinen weitere Hilfsfenster. Sie haben die Wahl, ob Sie:

| | |
|---|---|
| im Normalverfahren auf Cassette | *C* |
| im SuperTape-Format | *S* |
| auf Diskette | *D* |

sichern wollen.(Die SuperTape-Option ist ab Ausgabe 1/86 realisiert.) Beachten Sie bitte, daß Sie die Programme von Ihrem Datenträger immer als normale BASIC-Programme mit *LOAD "Name ",1* bzw. *LOAD "Name ",8* laden müssen.

## Bei Ladeproblemen:

Schimpfen Sie nicht auf uns, die Bänder sind normgerecht nach dem neuesten technischen Stand aufgezeichnet und sorgfältig geprüft. Sondern: Reinigen Sie zunächst Tonköpfe und Bandführung Ihres Kassettenrecorders. Die genaue Vorgehensweise ist im Handbuch der Datasette beschrieben. Führt auch dies nicht zum Erfolg, ist wahrscheinlich der Tonkopf Ihres Gerätes verstellt. Dieser Fehler tritt leider auch bei fabrikneuen Geräten auf.

Wir haben deshalb ein Programm entwickelt, mit dessen Hilfe Sie den Aufnahme-/Wiedergabekopf justieren können. Tippen Sie das Programm JUSTAGE ein, und speichern Sie es ab. Dieses Programm wertet ein etwa 30 Sekunden langes Synchronisationssignal aus, das sich am Ende jeder Kassettenseite befindet. Starten Sie das JUSTAGE-Programm mit RUN, jetzt sollte die Meldung PRESS PLAY ON TAPE kommen, drücken Sie also die PLAY-Taste. Nach dem Drücken der Taste geht der Bildschirm zunächst wie immer aus. Wird das Synchro-Signal erreicht, wechselt die Bildschirmfarbe; und zwar – bei nicht total verstellter Spurlage – völlig regelmäßig etwa dreimal pro Sekunde. Liegt die Spur des Tonkopfes grob außerhalb der zulässigen Toleranzgrenzen, geschieht entweder nichts oder die Farben wechseln unregelmäßig. Nehmen Sie jetzt einen kleinen Schraubenzieher und werfen Sie einen Blick auf Ihre Datasette. Über der REWIND-Taste befindet sich ein kleines Loch. Wenn Sie bei gedrückter PLAY-Taste durch dieses Loch schauen, sehen Sie den Kopf der Justierschraube für die Spurlage. Drehen Sie diese Einstellschraube. Aber Vorsicht: ganz langsam drehen, ohne dabei Druck auszuüben! Drehen Sie die Schraube nicht mehr als eine Umdrehung in jede Richtung. Nach etwas Ausprobieren wird der Bildschirm gleichmäßig die Farbe wechseln. Zur Feineinstellung lassen Sie das Synchro-Signal noch einmal von Anfang an laufen. Die Schraube jetzt nach links drehen, bis der Farbwechsel unregelmäßig wird. Diese Stellung genau merken, und die Schraube jetzt langsam wieder nach rechts drehen: Der Farbwechsel wird zunächst gleichmäßig, bei weiterem Drehen wieder unregelmäßig. Merken Sie sich auch diese Stellung, und drehen Sie die Schraube nun in Mittelstellung, das heißt zwischen die beiden Randstellungen. Denken Sie daran, daß während der Einstellung kein Druck auf den Schraubenkopf ausgeübt werden darf! Der Tonkopf Ihres Recorders ist jetzt justiert.

Sollte sich auch nach dieser Einstellung INPUT 64 nicht laden lassen, erhalten Sie von uns eine Ersatzkassette. Schicken Sie bitte die defekte Kassette mit einem entsprechenden Vermerk an den Verlag ein (Adresse siehe Impressum).

PS! In der Ausgabe 6/85 haben wir das Programm RECORDER-JUSTAGE veröffentlicht, das die Einstellung des Daten-Recorders zum Kinderspiel macht.

## Listing Justage

```
800 fori=49199to49410:read d:ps=ps+d:poke i,d:next
900 ifps<>24716thenprint"falsch abgetippt - fehler korrigieren!":end
950 print"o.k."
970 sys49338
1000 rem von 49199 bis 49410
1010 data173, 13,220,169,217,174,  4,220,172,  5,220,141, 14,220, 48, 44, 56
1020 data102, 88, 36, 89, 48, 12,144, 10,165, 88,133, 90,169,128,133, 88,133
1030 data 91,192,121,144,  4,224,115,176,  7,169,  0,133, 92, 56,176, 11,165
1040 data 92, 73,128,133, 92, 36, 92, 16, 19, 24,102, 88, 36, 89, 48, 12,144
1050 data 10,165, 88,133, 90,169,128,133, 88,133, 91,104,168,104,170,104, 64
1060 data 96, 36, 91, 16,252,132, 91,165, 90, 96,160,128,132, 89,165, 88,201
1070 data 22,208,250,132, 88,160, 10,132, 89,132, 91, 36, 91, 16,252,132, 91
1080 data165, 90,201, 22,208,226,136,208,241, 32,133,192,201, 22,240,249, 96
1090 data 32,147,252,120, 32, 23,248,165,  1, 41, 31,133,  1,133,192,169, 47
1100 data141, 20,  3,169,192,141, 21,  3,169,127,141, 13,220,169,144,141, 13
1110 data220,173, 17,208, 41,239,141, 17,208,169, 70,141,  4,220,169,129,141
1120 data  5,220, 88, 32,142,192,201, 42,208,249,173, 32,208, 41, 15,168,200
1130 data140, 32,208, 76,237,192,208, 76

ready.
```

# Am 12. Mai '86 an Ihrem Kiosk: INPUT 64 Ausgabe 5/86

**Wir bringen unter anderem:**

### WM'86

Rechtzeitig zur Fußball-Weltmeisterschaft in Mexico: das Begleitprogramm für Ihren 64er. Sämtliche Paarungen der Vorrunde mit Spielorten und Terminen "kennt" das Programm bereits. Viertel-Finale und Halb-Finale werden nach den realen Ergebnissen berechnet.

### MultiTape

Ein Universal-Kopierprogramm für Kassetten. Das Aufzeichnungsformat (Commodore, SuperTape und die anderen gängigen Ladeverfahren) wird beim Lesen vom Programm erkannt, das Schreibformat ist wahlweise SuperTape oder Commodore. Dadurch können Sie jede Kassette in das schnellere SuperTape-Format konvertieren.

### Des LISP's zweiter Teil

Das LISP-Paket aus dieser Ausgabe wird komplett: EDITOR, TRACER, ein Programm zur komfortablen I/O-Steuerung und die Befehlserweiterung MACROS.LSP, die "benutzerfreundliche" Kontroll-Strukturen wie REPEAT, FOR, WHILE, IF enthält.

### Scrolling

Ein Irrgartenspiel mit 18 soft-scrollenden Hires-Bildern, in denen 99 Augen und 16 verschiedene Gesteinsarten in 9999 Sekunden entdeckt werden müssen, damit Sie 500000 Punkte erreichen und die 26 Zielsteine abräumen können. Zuvor müssen Sie in auswegslosen Situationen einige Male mit dem Kopf durch die Wand gehen.

### und außerdem:

Nico macht Voraussagen für's Würfeln, 64er-Tips, ID-Werkstatt. . .

---

## c't Magazin für Computertechnik

**Ausgabe 5/86 – ab 17.4.1986 am Kiosk**

Software-Know-how: Wie funktioniert der Relocator aus INPUT 64 * Fehlerkorrigierende Codes * Das Betriebssystem des Atari ST * Projekt: Solid State Floppy – 256 KByte dauerhaft im Zugriff * Aplle II erhält Interface für ECB-I/O-Karten * Appplikation: 6809-CPU * Prüfstand: Mäuse-Parade * HRG-Karte für PCs * u.v.a.m.

## elrad – Magazin für Elektronik

**Ausgabe 5/86 – ab 28.4.1986 am Kiosk**

Bauanleitung: elSat (5.Teil) – die Satelliten-Direktempfangsanlage ist fertig * elrad-Marktreport: Energie tanken! NiCd-Lader und was sie kosten * Bauanleitung Bühne: Leistungsdimmer 3 kW * Audio: Linkwitz-Filter, Teil 2 * Bauanleitungen Fototechnik: Belichtungsmesser, Netzblitzgerät * Grundlagen: Die elrad-Laborblätter mit MOSFETs * u.v.a.m.

## IMPRESSUM

**INPUT 64**
Das elektronische Magazin

Verlag Heinz Heise GmbH
Bissendorfer Str. 8
3000 Hannover 61
Postanschrift:
Postfach 610407
3000 Hannover 61
Tel.: (05 11) 53 52-0

**Technische Anfragen**
nur dienstags von 9-16.30 Uhr

Postgiroamt Hannover, Konto-Nr. 93 05-308
(BLZ 250 100 30)
Kreissparkasse Hannover, Konto-Nr. 000-01 99 68
(BLZ 250 502 99)

**Herausgeber:** Christian Heise

**Redaktion:**
Christian Persson (Chefredakteur)
Ralph Hülsenbusch
Wolfgang Möhle
Karl-Friedrich Probst
Jürgen Seeger

**Ständige Mitarbeiter:**
Peter S. Berk
Irene Heinen
Peter Sager
Hajo Schulz
Eckart Steffens

**Vertrieb:** Anita Kreutzer-Tjaden

**Redaktion, Anzeigenverwaltung, Abonnementsverwaltung:**

Verlag Heinz Heise GmbH
Postfach 610407
3000 Hannover 61
Tel.: (05 11) 53 52-0

**Grafische Gestaltung:**
Wolfgang Ulber, Dirk Wollschläger

**Herstellung:** Heiner Niens

**Lithografie:**
Reprotechnik Hannover

**Druck:**
Leunisman GmbH, Hannover
CW Niemeyer Hameln

**Konfektionierung:**
Lettershop Brendler, Hannover

**Kassettenherstellung:**
SONOPRESS GMBH, Gütersloh

**INPUT 64** erscheint monatlich.
Einzelpreis DM 14,80
Jahresabonnement Inland Kassette DM 140.-
Diskette DM 198,-
Diskettenversion im Direktbezug: DM 16,80
+ DM 3,- Porto und Verpackung

**Vertrieb (auch für Österreich, Niederlande, Luxemburg und Schweiz):**
Verlagsunion Zeitschriften-Vertrieb
Postfach 5707
D-6200 Wiesbaden
Ruf (0 61 21) 2 66-0

**Verantwortlich:**
Christian Persson
Bissendorfer Str. 8
3000 Hannover 61



Printed in Germany


**ISSN 0177-3771**

**Titelidee INPUT 64**
Titelfoto: Bavaria

Titel – Grafik und – Musik:
Tim Pritlove
Fabian Rosenschein

Betriebssystem:
Hajo Schulz

hier abtrennen

# INPUT 64-Abonnement Abruf-Coupon

Ja, übersenden Sie mir bis auf Widerruf alle künftigen INPUT64-Ausgaben ab Monat

_____

(Kündigung ist jederzeit mit Wirkung ab der jeweils übernächsten Ausgabe möglich. Überzahlte Abonnementsgebühren werden sofort anteilig erstattet.)

Das Jahresabonnement kostet: ○ auf Kassette DM 140,— inkl. Versandkosten und MwSt.

○ auf Diskette DM 198,— inkl. Versandkosten und MwSt.

(Bitte ankreuzen/Nichtzutreffendes streichen.)

Absender und Lieferanschrift

Bitte in jedes Feld nur einen Druckbuchstaben (ä = ae, ö = oe, ü = ue)

Vorname/Zuname

Beruf/Funktion

Straße/Nr.

PLZ Wohnort

_____

Datum/Unterschrift

Von meinem Recht zum schriftlichen Widerruf dieser Order innerhalb einer Woche habe ich Kenntnis genommen. Zur Wahrung der Frist genügt die rechtzeitige Absendung.

Unterschrift _____

Bitte beachten Sie, daß diese Bestellung nur dann bearbeitet werden kann, wenn beide Unterschriften eingetragen sind.

**INPUT 64-Abonnement Abruf-Coupon**

Ich wünsche Abbuchung der Abonnement-Gebühr von meinem nachstehenden Konto. Die Ermächtigung zum Einzug erteile ich hiermit.

Name des Kontoinhabers

Bankleitzahl Konto-Nr.

Geldinstitut Ort des Geldinstituts

Bankeinzug kann nur innerhalb Deutschlands und nur von einem Giro- oder Postscheckkonto erfolgen.

HEISE

Bitte im (Fenster-)Briefumschlag einsenden.
Nicht als Postkarte verwenden!

# INPUT64

**Vertriebsabteilung**
**Verlag Heinz Heise GmbH**
**Postfach 61 04 07**

**3000 Hannover 61**