```
110 REM    ****************************
120 POKE 650,255:Y1$="":Q2=Q2-1:POKE 214,Q1
   :POKE 211,Q2:PRINT"{UP}";RIGHT$(Y3$,Q3)
                        <071>
130 FOR II=1 TO Q3+1                        <123>
140 GET Y2$:IF Y2$=""THEN 140               <130>
150 IF ASC(Y2$)=20 AND II>1 THEN Y1$=LEFT$(Y1$,
   LEN(Y1$)-1):II=II-2:GOTO 250             <253>
160 IF ASC(Y2$)=13 AND II=1 AND Q4=1 THEN GOSUB
    280:GOTO 140                            <104>
170 IF ASC(Y2$)<>13 AND II=Q3+1 THEN GOSUB 280
   :GOTO 140                                <059>
180 IF ASC(Y2$)=13 GOTO 260                 <060>
190 IF ASC(Y2$)<32 OR ASC(Y2$)>93 THEN GOSUB 280
   :GOTO 140                                <224>
200 IF Q5=1 AND ASC(Y2$)=45 AND II=1 GOTO 240
                        <143>
210 IF II<=Q7 THEN Q7=0:Q6=1                <145>
220 IF Q5=1 AND ASC(Y2$)=46 AND Q6=1 THEN Q6=0:
   :Q7=II:GOTO 240                          <035>
230 IF Q5=1 AND ASC(Y2$)<48 OR Q5=1 AND ASC(Y2$
   )>57 THEN GOSUB 280:GOTO 140             <063>
240 Y1$=Y1$+Y2$                             <087>
250 POKE 214,Q1:POKE 211,Q2:PRINT"{UP}";
   LEFT$(Y1$+"T"+Y3$,Q3)+"←";:NEXT II       <004>
260 POKE 214,Q1:POKE 211,Q2:PRINT"{UP}";
   Y1$+LEFT$(Y4$,Q3-LEN(Y1$)+1)             <242>
270 Q1=0:Q2=0:Q3=0:Q4=0:Q5=0:Q6=0:Q7=0
   :POKE 650,0:RETURN:'<--- AUSGANG AUS UP <117>
280 POKE 54296,15:POKE 54277,6:POKE 54278,0
   :POKE 54275,8:POKE 54274,0               <046>
290 POKE 54273,92:POKE 54272,237:POKE 54276,65
   :FOR JJ=1 TO 150:NEXT JJ:POKE 54276,0    <084>
300 RETURN:'<--- AUSGANG AUS PIEP-UP        <061>

310 REM"┌──────────────────────────────────┐
320 REM"| BEISPIEL 1 FUER DEN AUFRUF :     |
330 REM"|                                  |
340 REM"| 10 Y3$=''.........←'':Y4$=''     |
350 REM"|          ''                      |
360 REM"| 20 PRINT''□'':POKE214,5:POKE     |
370 REM"|    211,15:PRINT''□NAME :''       |
380 REM"| 30 Q1=5:Q2=23:Q3=10:GOSUB 100    |
390 REM"| 40 PRINT:PRINT''SIE HEISSEN''    |
400 REM"|    ;Y1$:END                      |
410 REM"|                                  |
420 REM"| BEISPIEL 2 FUER DEN AUFRUF :     |
430 REM"|                                  |
440 REM"| 10 Y3$=''.........←'':Y4$=''     |
460 REM"| 20 PRINT''□'':POKE214,5:POKE     |
470 REM"|    211,15:PRINT''□ALTER :''      |
480 REM"| 30 Q1=5:Q2=24:Q3=10:Q4=1:        |
490 REM"|    Q5=1:GOSUB 100                |
500 REM"| 40 PRINT:PRINT''SIE SIND'';      |
510 REM"|    VAL(Y1$);''JAHRE ALT'':END    |
520 REM"|                                  |
530 REM"| HINWEIS :                        |
540 REM"|                                  |
550 REM"| POKE 214,Y BEWIRKT, DASS DER     |
560 REM"| CURSOR IN DIE ZEILE Y SPRINGT    |
570 REM"|                                  |
580 REM"| POKE 211,X BEWIRKT, DASS DER     |
590 REM"| CURSOR AN D. SPALTE X SPRINGT    |
600 REM"└──────────────────────────────────┘

READY.
```

Formatierte Eingabe

Einzeiler-Wettbewerb: Die nächsten 14

Wieder haben wir die interessantesten Einzeiler für Sie herausgesucht. Darunter sind sowohl nützliche als auch witzige oder lehrreiche. Sogar ein Adventure ist darunter. Alle Veröffentlichungen werden mit 50 Mark und einer Diskette mit allen Programmen dieser Ausgabe belohnt.

Wenn man einigen Einsendern Glauben schenken will — und es besteht kein Grund, das nicht zu tun — , so sind ihre Einzeiler keineswegs schnell dahingeschriebene Programme, sondern oft das Ergebnis von tage-, ja wochenlangen Experimenten und Versuchen. Bei manchen anderen allerdings ist die programmiertechnische Umsetzung einer Idee nicht das Entscheidungskriterium gewesen, sondern vielmehr die Idee selbst. Sie werden auch sehen, daß bei einem Einzeiler weder das Programm noch die Idee besonders originell ist. Aber seine Programmbeschreibung ist so überzeugend, daß sie Ihnen nicht vorenthalten sein soll. Doch nun viel Spaß und hoffentlich einige »Aha«-Erlebnisse.

Umwandlung beliebiger Zahlensysteme (VC 20/C 64)

Die beiden Einzeiler dienen zum Umrechnen zwischen Dezimalzahlen und Zahlen beliebiger Basis. Kombiniert bilden beide eine Umwandlung zwischen verschiedensten Zahlensystemen. Beide Zeilen können sowohl als Unterroutine (mit RETURN) wie auch als Teil eines größeren Programms stehen.

a) **Umwandlung dezimal/beliebig**

Die Routine wandelt eine Dezimalzahl beliebiger Größe in der Variablen D in eine Zahl der Basis um, die in der Variablen B angegeben ist. Das Ergebnis steht in Z$. Zuerst wird Z$ gelöscht. Dann wird eine Dummy-Schleife eröffnet, die nur einen einzigen Durchlauf zu haben scheint (0 bis 0). Der Befehl wird dazu genutzt, später wieder mitten in die Zeile einspringen zu können. Bei jeder Stelle wird D durch die Basis B geteilt und dadurch die unterste Ziffer abgeschnitten. Die jeweils niederwertigste Stelle ist der ganzzahlige Rest dieser Division und steht in S. S wird nun in den ASCII-Code umgerechnet, indem zur Zahl S 45 addiert wird. Ist S eine Ziffer von 0 bis 9, so nimmt der Term (S < 10) den Wert —1 an und es wird 7 subtrahiert (siehe ASCII-Tabellen). Der Code wird durch den CHR$-Befehl in einen String gewandelt und vorne an Z$ angehängt. Die letzte (höchstwertige) Ziffer ist erreicht, wenn D < 1 ist (nächste Stelle = 0). Die Schleifenvariable P wird auf —D gesetzt. Beim darauffolgenden NEXT-Kommando wird P um den STEP (hier 1) erhöht und mit dem Endwert des FOR-Befehls (hier 0) verglichen. Wenn D noch > = 1 ist, ist P + 1 < = 0, die Endbedingung ist noch nicht erreicht und es wird zu dem Statement nach dem FOR-Befehl gesprungen.

b) **Umwandlung beliebig/dezimal**

Diese Routine wandelt eine beliebige große Zahl der von der Variablen B angegebenen Basis in eine Dezimalzahl. Die zu wandelnde Zahl muß in Z$ stehen; die Variable D enthält das Ergebnis. Am Anfang wird die Variable D auf Null gesetzt. Die Schleifenvariable S des darauffolgenden FOR-Befehls dient als Zeiger auf die einzelnen Stellen von Z$. Diese werden nun in den ASCII-Code gewandelt, der Code für Null (48) wird subtrahiert und das Ergebnis in H zwischengespeichert. Die schon umgewandelten Stellen (in D) werden zunächst durch Multiplikation mit B um eine Potenz dieser Basis erhöht, um dann die aktuelle Stelle (H) zu addieren. Ist

H>9 (Darstellung durch einen Buchstaben) müssen aufgrund des ASCII-Codes noch sieben abgezogen werden (Term (H>9) wird —1). Nach der niederwertigsten Stelle ist die Schleife beendet.

(Martin + Hartmut Sprave)

```
10 z$="":forp=0to0:d=d/b:s=(d-int(d))*b:
z$=chr$(55+s+7*(s<10))+z$:p=-d:next
11 :
12 :
13 :
20 d=0:fors=1tolen(z$):h=asc(mid$(z$,s))
-48:d=d*b+h+7*(h>9):next
21 :
22 :
23 :
30 rem zahlenumwandelung:
40 rem in 10 dez in beliebig
50 rem in 20 beliebig in dez
60 :
```

Scrollen in x-Richtung (C 64)

FORG=0TO3:NEXT ist eine Verzögerungsschleife; hier kann die Geschwindigkeit des Scrollers eingestellt werden. Zu beachten ist, daß alle Befehle in abgekürzter Schreibweise eingegeben werden müssen.

(Hans-Peter Harmann)

```
10 fort=1to7:poke53270,t:forg=0to3:next:
next:onagoto10:fory=1024to2023:pokey,194
:next:a=1:goto10
20 rem
```

Das abwechslungsreiche Programm (C 64)

Liebe Redaktionäre, endlich ein Wettbewerb, an dem auch ich als Novize mich beteiligen kann. Meine Computer- und Programmierkenntnisse sind mittlerweile schon weit fortgeschritten (die 1-Prozent-Hürde werde ich demnächst überspringen), so daß ich mich berufen glaube, auch etwas zum Besten zu geben.

Ich stellte mir selbst hohe Anforderungen:

1. Das Programm sollte abwechslungsreich sein
2. Es sollte Bewegung im Spiel sein
3. Die Farbe durfte nicht fehlen.

Bitte das Programm sorgfältig abschreiben, um eine langwierige Fehlersuche zu vermeiden.

Variablenliste:

A$ = Stunden von TI$
B$ = Minuten von TI$
C$ = Sekunden von TI$

Vor Eingabe des Programms muß selbstverständlich TI$ auf die aktuelle Uhrzeit eingestellt werden. Die Zeilennummer 11 fristet für gewöhnlich ein Schattendasein hinter der 10. Deshalb habe ich aus Mitleid die 11 gewählt. Nachdem Sie den Bildschirm durch CLR/Home gereinigt haben, starten Sie das Programm. Sie sehen, daß meine Anforderungen erfüllt wurden.

1. Das Programm ist abwechslungsreich (Wiederholungen treten höchstens nach 24 Stunden auf)
2. Es bewegt sich was
3. Es ist Farbe im Spiel

Zu 3. Das Programm wurde so gestaltet, daß der Anwender leicht die Farbe ändern kann.

PS: Das Ausschalten des Computers bringt das Programm zum Abstürzen!!

P.P.S: Freundlich zugedachte Geldspenden werden nur auf Antrag angenommen.

(Gerd Pickard)

```
11 a$=left$(ti$,2):b$=mid$(ti$,3,2):c$=r
ight$(ti$,2):print"■";a$;":";b$;":";c$:g
oto11
```

Ein RENEW, das funktioniert (C 64/VC 20)

Routinen, die nach einem Reset oder NEW das Basic-Programm zurückholen, sind zwar schon oft veröffentlicht worden, doch waren die meisten entweder nicht lauffähig oder das Abtippen wurde durch die Länge zu einer umständlichen und unsicheren Prozedur. Dieser Einzeiler ist natürlich im Direktmodus, also ohne Zeilennummer, einzugeben. Vorher dürfen jedoch keine Variablen definiert oder Basic-Zeilen eingetippt werden, da sonst das gelöschte aber noch im Speicher befindliche Programm zerstört werden würde. Das Prinzip des UNNEW-Programms beruht auf dem Aufruf einer System-Routine, die die Basic-Zeilen neu bindet und das Ende des gelöschten Programms herausfindet. Der erste POKE dient nur dazu, der Routine vorzutäuschen, daß sich noch ein ungelöschtes Basic-Programm im Speicher befindet, da sie sonst nicht arbeitet. Die Endadresse des Basic-Programms wird um zwei erhöht und in den Zeiger auf den Start der Variablen (45/46) übertragen. Der CLR-Befehl gleicht alle weiteren Basic-Zeiger diesem Wert an.

SYS-Adresse für VC 20: 50483. Da beim VC 20 die Startadresse des Basic-Programms je nach Speicherausbau verschieden ist, muß man die Zahl »2050« durch das Ergebnis des folgenden Terms ersetzen« PRINT/PEEK(43)+256*PEEK(44)+1.

(Hartmut + Martin Sprave)

```
6 rem c-64
7 poke2050,8:sys42291:poke46,peek(35)-(p
eek(781)>253):poke45,peek(781)+2and255:c
lr
8 rem   vc-20:
9 rem   wie oben,aber sys50483  und
10 rem  statt 2050: poke(43)+256*poke(44
)+1

ready.
```

Eine Zeile — kompletter Datenschutz (C 64)

Dieses Programm verhindert das Auflisten des Inhaltsverzeichnisses jeder beliebigen Diskette. Nebenbei stellen sich noch einige sehr brauchbare Effekte ein. Das Löschen und Überschreiben der auf der Diskette befindlichen Programme ist nicht mehr möglich. Außerdem kann nichts mehr auf die Disk geschrieben werden. Ebenso werden Diskettenbefehle wie VALIDATE oder INITIALIZE ignoriert. Sogar ein Headern der Diskette ohne neue ID ist nicht möglich. Das einzige, was den Inhalt der Diskette noch manipulieren kann, ist das Headern (Diskettenbefehl »NEW«) mit einer neuen ID-Nummer.

Das Laden der bereits auf der Disk befindlichen Programme funktioniert hingegen ganz normal. Allerdings sollte man sich die Namen der gespeicherten Files merken, denn das

Suchen mit Jokerzeichen ist bestenfalls ein netter Zeitvertreib, aber nicht unbedingt immer erfolgreich. Hat man beispielsweise zwei Programme, die mit A anfangen, auf der Diskette, so muß man von dem zweiten Programm ja mindestens die ersten beiden Buchstaben angeben, um es laden zu können.

Daß das Directory nicht mehr gelistet werden kann liegt daran, daß es als Basic-Programm geladen wird und in dem veränderten Directory drei Nullen am Anfang erscheinen. Dies ist für den Interpreter jedoch das Zeichen für das Programmende. Das »Directory-Programm« endet also bereits nach zehn Bytes. Die ersten fünf Bytes stellen den Zeilenanfang und die Zeilennummer dar. Darauf folgen ein Leerzeichen und ein Anführungszeichen, die ja immer am Anfang eines Inhaltsverzeichnisses stehen. Um diese drei Zeichen beim Auslisten verschwinden zu lassen, folgen nun drei chr$(20), die jeweils ein Delete darstellen. Dies bewirkt, daß das Directory beim Auslisten nun völlig verschwindet. Das Listen wird hier abgebrochen, da nun unmittelbar die drei Nullen folgen, die das Programmende markieren. So erfolgt auf den List-Befehl nur die Meldung »READY«.

Die reversen »t« werden am einfachsten eingegeben, wenn man zuerst zwei Anführungszeichen hintereinander schreibt, dann den Cursor zurück auf das zweite Anführungszeichen bewegt, dreimal die Taste »INST« (= Insert) und danach dreimal die Tasten »SHIFT« und »INST« (= Delete) betätigt. Die Zeile kann natürlich genauso im Direktmodus abgeschickt werden.

(Volker Ritzhaupt)

```
1 open1,8,3,"#":open2,8,15,"b-p3,144":pr
int#1,"■■"chr$(0)chr$(0)chr$(0):print#2
,"u2:3,0,18":print#2,"i
2 rem
3 rem verhindert auflisten des directory
      und macht schreibschutz auf disk
4 rem
```

Geänderter Zeichensatz (C 64)

Das Programm enthält eine Variable, sechs POKE-Befehle, einen PEEK-Befehl und eine FOR-NEXT-Schleife. In der Variablen R wird die Adresse des Interrupt-Registers definiert (#56334/$DC0E). Nun wird die Position des Zeichengenerators geändert, indem Bit 3 der Adresse 53272 ($D018) gesetzt und Bit 1 gelöscht wird. So wird der Bereich ab #8192 beziehungsweise $2000 ausgewählt. Um den alten Zeichensatz lesen zu können, muß der Interrupt ausgeschaltet werden (Bit 0 in Adresse 56334 gelöscht), ebenso der Video-Chip (Bit 2 in Adresse 1 gelöscht). Die folgende Schleife würde in einem übersichtlichen Programm so aussehen

FOR I = 0 TO 4095
POKE 8192+I,
PEEK(I+53248) AND 60
NEXT

Es wird also der Zeichensatz von Adresse #53248 beziehungsweise $D000 an nach #8192 übertragen, wobei jedes Byte durch die AND-Funktion geändert wird. Diese bewirkt, daß die Bits 0, 1, 6 und 7 in jedem Byte gelöscht werden, also ganz einfach jedem Zeichen der rechte und der linke Rand abgeschnitten wird. In meinem Programm überdeckt die Schleife einen etwas größeren Bereich als eigentlich nötig wäre, nämlich von 6hoch5 = 7776 bis R/5 = 11267, was dem Resultat jedoch keinen Abbruch tut.

Schließlich werden Video-Chip und Interrupt wieder eingeschaltet (Bit 2 in Adresse 1 und Bit 1 in Adresse 56334 gesetzt).

Bemerkenswert ist vielleicht, daß ich nach der Realisierung meiner Idee ein einziges Zeichen zuviel im Programm hatte. Nach drei Stunden angestrengten Tüftelns kam mir die erlösende Idee. Ursprünglich hatte ich den Video-Chip durch POKE1,55 wieder eingeschaltet. Es genügt aber auch POKE1,7, da die übrigen Bits vom Prozessor automatisch gesetzt werden.

(Klaus Vorwalter)

```
1 r=56334:poke53272,24:poker,0:poke1,51:
fori=6↑5tor/5:pokei,peek(i+45056)and60:n
ext:poke1,7:poker,1
2 rem
```

Trick 17 mit ON...GOTO (C 64/VC 20)

Eine sehr interessante Version einer ON...GOTO-Anweisung. Zur Erklärung braucht eigentlich nur gesagt werden, daß der Ausdruck (A$="A") den Wert —1 hat, wenn ein »A« eingegeben wurde, sonst den Wert 0. Rechnen Sie nach oder probieren Sie es aus: Es funktioniert einwandfrei.

(Peter Zankl)

```
100 rem tastaturabfrage mit sprung
200 rem :
300 rem vorher:
400 :
410 geta$:ifa$=""then410
420 ifa$="a"then2000:rem programmteil a
430 ifa$="b"then3000:rem programmteil b
440 ifa$="x"then end:rem ende
450 goto 410
499 :
500 rem nachher:
600 :
610 geta$:on1-(a$="a")-2*(a$="b")-3*(a$=
"x")goto610,2000,3000:end
620 :
```

Das kürzeste Abenteuerspiel der Welt (C 64/VC 20)

Das Spiel heißt »Nosferatu«. Im Spiel selbst stehen Sie einem Vampir gegenüber und müssen herausfinden, wie Sie ihn besiegen. Erlaubt sind deutsche Zweiwortkommandos. Eine Alternative für die Lösung besteht, indem Sie anstatt »WIRF KNOBLAUCH« »ZEIGE KREUZ« eintippen. Sie können sich auch andere Situationen überlegen, was das Programm noch zum kürzesten Adventure-Generator macht Cursorsteuerzeichen: Reverses Q = Cursor hoch, Reverses q = Cursor runter. Bitte achten Sie auf mein Copyright (für einen Vermerk war leider kein Platz) DARUM HIER: (C) Copyright 1984, by me. Und noch etwas: Mogeln Sie beim Abtippen bitte nicht! Es wäre schade, wenn Sie schon beim Eintippen die Lösung kennen würden.

Vampirische Grüße
(Thomas Werner)

```
1 print"ein vampir!":inputa$:print"sieg!
":ifa$<>"wirf knoblauch"thenprint"■klapp
t nicht!■":goto1
2 rem
```

Dividieren mit beliebig vielen Stellen hinter dem Komma (V 64)

Das Programm muß mit den bekannten Abkürzungen eingegeben werden, damit es in das 80-Zeichenformat paßt.

Variablenliste
Z = Zahl (mathematisch korrekt : DIVISOR)
D = Dividend
N = Anzahl der gewünschten Stellen hinter dem Komma
E% = Ergebnis ohne Rest (wird ausgegeben)
R = Rest
Q% = Quotient beschränkt auf Vorkommazahl

Beschreibung:

Um die Vorkommazahl zu erhalten, benötigt man Q = INT(Z/D), was ich durch Q% = Z/D vereinfacht habe. Q% wird als Vorkommazahl ausgegeben. Es wird der Rest, nämlich die Differenz aus Z und Q% * D gebildet. Der Rest dient als Ausgangspunkt für die folgende Berechnung. Die Schleife wird Nmal durchlaufen, um exakt N Stellen hinter dem Komma zu ermitteln.

Analog zum Vorschleifenteil wird E% = R * 10/D gebildet. E% wird ausgegeben als I.te Stelle.

Der neue Rest wird durch $R_{(neu)} = 10 * R_{(alt)} - D * E\%$ (vergleiche Vorkommateil) gebildet.

Und wiederum kann die Schleife durchlaufen werden, bis alle Stellen ausgegeben sind.

Beispiel:

Wie lautet die 85. Stelle hinter dem Komma von $^{116}/_{13}$? (also: Z = 116, D = 13, N = 85) Programmlauf ....

Ergebnis:

Die 85. Stelle hinter dem Komma ist 9. (Selbstverständlich werden auch alle übrigen Stellen ausgegeben).

(Heinz Bauschke)

```
0 inputz,d,n:q%=z/d:printq%:r=z-d*q%:for
i=1ton:e%=r*10/d:print"■";e%;:r=10*r-d*e
%:next
10 rem
```

Simulation — GOTO X (C 64)

Das Programm simuliert den im Commodore-Basic nicht vorhandenen Befehl GOTO X.

Zu dem etwas seltsamen Aussehen der Zeile läßt sich folgendes sagen: In den Anführungszeichen steht ein Maschinenprogramm, dessen Opcodes im Listing des Basic-Interpreters diese merkwürdigen Zeichen erzeugen. Das disassemblierte Maschinenprogramm finden Sie in Listing 2, wobei die absoluten Adressen keine Bedeutung haben, das Programm adressiert nur relativ. Der mit der Materie etwas vertrautere Leser wird feststellen können, daß das Programm etwas umständlich geschrieben ist. Das liegt daran, daß nicht alle Character, die ein Opcode darstellen, eindeutig einer Zahl zugeordnet sind. So gäbe zum Beispiel die Zahl 255 im Maschinenprogramm das Zeichen (Commodore & K), wenn die Zeile aber dann editiert würde, interpretiert das Basic dieses Zeichen aber mit 161. So ließe sich die Zeile nicht editieren, wäre also keine Basic-Zeile. Aus diesem Grund wurden alle solchen »doppeldeutigen« Codes umgangen, was die Umständlichkeit des Maschinenprogramms zur Folge hatte. Die SYS-Anweisung stellt die Lage des Maschinenprogramms mit Hilfe zweier Zeiger des Interpreters fest und startet das Programm.

Nun zur Benutzung der GOTO X-Zeile und wie man sie eingibt: Es ist zwar möglich, die Zeile mit der in Ausgabe 7/84 beschriebenen »Finkel«-Methode einzugeben, was jedoch umständlich, zeitaufwendig und fehleranfällig wäre. Deshalb ist in Listing 3 ein Generierungsprogramm abgebildet. Dieses wird eingegeben und gestartet.

Wenn der Checksum-Test positiv ausgefallen ist und man sich vergewissert hat, daß das Programm genauso eingegeben wurde, wie es ausgelistet ist, gibt man LIST ein, gefolgt von NEW. Dann fährt man mit dem Cursor auf Zeile 10 und drückt RETURN. GOTO X steht nun im Speicher und kann gelistet, editiert und abgeSAVEt werden.

Selbstverständlich kann die Zeilennummer zwecks der Einbindung in eigene Programme geändert werden. Dazu weist man der Variablen LL% den Wert der anzuspringenden Zeile zu und springt mit GOTO zur Zeile, in der GOTO X steht.

(Reinhard Jurk)

```
10 syspeek(61)+256*peek(62)+26:"■■■■■■■■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
20 rem
30 rem goto x: ll%=zeilennummer:goto10
40 rem
```

Listing 1

```
C*
     PC   IRQ   NV-BDIZC AC XR YR SP
.;   E147 EA31  00110001 08 08 02 F6
.
.,   081A   A9 CC       LDA #$CC
.,   081C   85 45       STA $45
.,   081E   85 46       STA $46
.,   0820   A5 9D       LDA $9D
.,   0822   85 55       STA $55
.,   0824   85 56       STA $56
.,   0826   A0 0C       LDY #$0C
.,   0828   C8          INY
.,   0829   91 55       STA ($55),Y
.,   082B   A2 81       LDX #$81
.,   082D   CA          DEX
.,   082E   C8          INY
.,   082F   8A          TXA
.,   0830   91 55       STA ($55),Y
.,   0832   AC 80 A3    LDY $A380
.,   0835   A6 2E       LDX $2E
.,   0837   A5 2D       LDA $2D
.,   0839   20 11 B1    JSR $B111
.,   083C   20 2B AF    JSR $AF2B
.,   083F   20 01 B8    JSR $B801
.,   0842   4C A3 A8    JMP $A8A3
.
.
READY.
```

Listing 2

```
G O T O  X

  BITTE UNBEDINGT GENAUSO EINGEBEN !!!
  43 KLAMMERAFFEN (@)


5 GOTO20
10 SYSPEEK(61)+256*PEEK(62)+26:"@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"
20 FORA=2082TO2124:READX:S=S+X:POKEA,X:N
EXT:IFS<>5185THENPRINT"DATA ERROR !":END
30 PRINT"OK."
40 DATA169,204,133,69,133,70,165,157,133
,85,133,86,160,12,200,145,85,162,129
50 DATA202,200,138,145,85,172,128,163,16
6,46,165,45,32,17,177,32,43,175,32,1
60 DATA184,76,163,168

READY.
```

Listing 3. Der Basic-Lader für GOTO X

Irrgarten (VC 20/C 64)

Das Programm belegt nur 40 Bytes Speicherplatz. Es erzeugt einen zufallsbedingten Irrgarten und läuft auf einem C 64 oder VC 20:

Zum Programm: X ist eine durch die RND-Funktion ermittelte Zufallszahl. Sie ist nicht gerundet und liegt zwischen 0 und 2. Der Computer druckt hierauf den CHR$-Code von 164 plus der Zufallszahl x aus, wobei er von sich aus X auf 0 oder 1 rundet. Es erscheint also auf dem Bildschirm je nach X das CHR$-Zeichen von 164 oder 165.

Der nach dem PRINT-Befehl folgende Strichpunkt bewirkt, daß das nächste Zeichen nicht erst in der nächsten Reihe, sondern gleich neben dem Zeichen davor erscheint.

Die Variable Q, welche durch den RUN-Befehl automatisch gleich Null gesetzt wurde, wird nun um 1 erhöht. Ist sie noch kleiner als 880, so wird die Schleife erneut durchlaufen. Das entstehende horizontale und vertikale Muster ist ein Labyrinth — aufgebaut aus nur diesen beiden Zeichen.

Durch Verwendung anderer CHR$-Zeichen (zum Beispiel 176 bis 179) läßt sich das Programm gut variieren.

Jedenfalls: Es ist nicht immer einfach den richtigen Weg zu finden!

(Marco Gleiter)

```
1 x=rnd(1)*2:printchr$(164+x);:q=q+1:ifq
<880thengoto1
2 rem
```

Clevere Idee: Grafikbildschirm löschen mit dem DIM-Befehl (C 64)

Programmbeschreibung:

1. a=0:b=0

Im weiteren Verlauf wird das Variablenfeld beeinflußt. Die Variablen a und b müssen deshalb vor der DIM-Anweisung angelegt sein.

2. a=peek(49):b=peek(50)

Die Werte für das Variablenende werden gesichert.

3. dim f((16191-a-b*256) /5)

Durch das Dimensionieren einer Variablen wird ein entsprechend großer Platz hinter dem Variablenende freigemacht beziehungsweise mit dem Wert 0 gefüllt. Nun muß der Platz nur noch mit der Adresse des Bildschirms übereinstimmen.

Zur Formel (16191-a-b*256) / 5 :

Der Bildschirmspeicher liegt von 8 192 bis 16 191. Die Variable f wird hinter dem bisherigen Variablenende angelegt.
Adresse des Variablenendes: a + b*256
also: 16191-a-b*256
Pro indizierte Variable werden 5 Byte freigemacht,
also: (16191-a-b*256) / 5

Die mit 0 indizierte Variable ist nicht berücksichtigt, ebenso die ersten 7 Byte für Variablenname und Dimension.

4. poke 49,a : poke 50,b

Die alten Werte werden wieder hergestellt. Die Variable f ist jetzt nicht mehr dimensioniert.

Zum ordnungsgemäßen Verlauf müssen zwei Punkte beachtet werden.

1. Das Variablenende muß kleiner als 8186 sein, das heißt das Programm darf einschließlich Variablenfeld nicht größer als 6 KByte sein. Mit 6 KByte steht jedoch ein ausreichend großer Platz zur Verfügung.

Ist dies nicht der Fall, wird lediglich der Bildschirm nicht ganz gelöscht. Das Programm kann in keinem Fall Schaden nehmen.

2. Soll das Grafikprogramm zum Abspeichern von Bildern benutzt werden, darf man nicht vergessen, den Zeiger für Speichergrenze (peek(55)+peek(56)*256) auf eine Adresse kurz hinter den Bildschirmspeicher zu setzen.

In meinem Grafikprogramm rufe ich die Zeile 10 immer dann auf, wenn über die Tastatur »shift + clr home« eingegeben wird (entsprechend »freier Bildschirm im Textmodus«).

(Manfred Hedtke)

```
10 a=0:b=0:a=peek(49):b=peek(50):dimf((1
6191-a-b*256)/5):poke49,a:poke50,b
20 rem
```

Zweifarbiger Rahmen (C 64)

Wie jeder weiß, ist der Bildaußenrahmen des C 64 immer einfarbig. Ich habe nun einen kleinen Einzeiler geschrieben, der einen zweifarbigen Außenrahmen simuliert. Die beiden Farben sind: 0 = schwarz und 1 = weiß. Es ist darauf zu achten, daß die Basic-Zeile genauso eingetippt wird, wie sie abgebildet ist. Ich meine damit die Anzahl der Spaces und Doppelpunkte. Erklärung: Ich wollte die Grenze zwischen den beiden Farben möglichst ruhig auf dem Bildschirm haben und mußte dazu die genaue Farbwechselperiode finden. Der Interpreter arbeitet Doppelpunkte und Spaces unterschiedlich schnell ab; so konnte ich mit den Spaces eine Feineinstellung vornehmen. Aber vorsichtig: Sobald während des Programmlaufs eine Taste gedrückt wird, tritt die Raster-Interrupt Tastaturabfrage in Kraft und das Bild fängt an zu »laufen«.

Hilfe zum Abtippen:
Zwischen den beiden POKEs, und den Doppelpunkten befinden sich je fünf Spaces. Es folgen dann 17 beziehungsweise 15 Doppelpunkte.

So einfach es auch immer ist: Es verblüfft einen doch!

(Markus Hillebrand)

```
10 poke53280,1      :::::::::::::::::poke
53280,0      :::::::::::::::goto10
20 rem
```

Fakultät (C 64)

Das Programm berechnet Fakultäten, ist also sehr nützlich für einige mathematische Funktionen, da der C 64 keinen derartigen Befehl besitzt.

Variablen:
A = Eingabevariable,
B = Zählvariable,
C = Rechen- und Ausgabevariable.

Das Programm berechnet sogar Fakultäten, die größer als »69« sind. Es ist also besser als ein Taschenrechner.

(Detlef Marks)

```
10 rem fakultaeten
20 :
30 inputa:frb=1toa:c=c+log(b):next:c=c/l
og(10):print10↑(c-int(c));"e";int(c):run
40 :
```