Hier die ersten Preisträger.

EINZEILER-WETTBEWERB:

Wir wußten gar nicht, was auf uns zukommen sollte, als die Idee Täglich gingen zirka 20 bis 40 Einzeiler bei uns ein.

Man glaubt gar nicht, wie interessant manche Einzeiler sind. Hier werden Probleme in eine Zeile gesteckt, bei denen manche Programmierer etliche KByte benötigen. Ein Beispiel ist das Mini-Orgel-Programm. Vielleicht finden Sie, daß hier mit einem Trick gearbeitet wurde, aber Einzeiler ist Einzeiler. Sehr viele Programme lassen sich nur eingeben, wenn die Abkürzungen der Basic-Befehle benutzt werden. Sie stehen im Anhang D Ihres C 64 Handbuches. Doch schauen Sie sich die folgenden Mini-Programme an, von denen man sagen kann: Klein — aber oho! Jedes von ihnen wird deshalb mit 100 Mark honoriert.

Merge (C 64)

```
10 a=peek(45)+256*peek(46)-2:poke44,a/25
6:poke43,a-peek(44)*256:print"prg laden
& p043,1:p044,8
```

Möchte man zwei Programme zu einem einzigen zusammenfassen, gab es oft nur eine Möglichkeit: das kürzere an das Ende des längeren zu tippen. Manche längere Hilfsprogramme vollbringen durchaus ein richtiges MERGE. Es geht aber auch ganz kurz. (Andreas Gast)

Directory laden, ohne Basic-Programm zu zerstören (C 64/VC 20)

```
0 get#1,a$:a=asc(a$+"■"):printchr$((a=13
0and13or((31<aanda<95)anda)));:goto0
```

Zunächst eine Bemerkung zu einem dem Programmierer sehr bekannten Problem. Hat man nun, mit Ach und Krach, ein wichtiges Programm in Basic geschrieben, und man will seine Arbeit mit dem Abspeichern dieses Programmes beenden, so kann es vorkommen, daß man seine freien Disketten nicht wiedererkennt. Also muß man sich die Disketten listen. Mit LOAD"$",8:LIST würde aber das eigene Basic-Programm gelöscht und die ganze Arbeit wäre umsonst. Wenn man dagegen diese kleine Zeile eingibt, und zwar so, daß sie vor dem eigenen Programm liegt, läßt sich das Directory listen, ohne das im Speicher befindliche Programm zu zerstören.
Zum Starten:
OPEN 1,8,2,"$":GOTO 0

Beschreibung

OPEN 1,8,2,"$"	eröffnet eine sequentielle Datei (hier das Directory) zum Lesen.
GET#1,A$	holt ein Byte vom Disketten-Puffer (Buffer).
A=ASC(A$+"SHIFT/HOME")	wandelt ASCII-Zeichen nach Zahl (von 0-255). Wenn A$ eine Länge von 0 hat, gibt es keine Fehlermeldung, da der Ausdruck trotzdem eine Länge von 1 behält.
PRINT CHR$((A=130 AND 13 OR ((31<A AND A<95) AND A)));	druckt für A das zugehörige Zeichen, wenn sich A zwischen 32 und 95 bewegt. Wenn A=130 ist, wird Return ausgegeben. Durch die Formel innerhalb der Characterstring-Klammer werden alle Steuerzeichen und Grafiksymbole »herausgefiltert«. Es bleiben dann nur die Zahlen, Buchstaben und Satzzeichen, die ausgedruckt werden.

Weitere Anwendungen:
Da man alle auf der Diskette befindlichen Programme als sequentielle Datei lesen kann, ist es ohne weiteres möglich, mit dem obigen Programm alle Kommentare, Inhalte von Printanweisungen und Texte des auf der Diskette befindlichen Programms auf den Bildschirm zu bringen. Zum Starten eröffnet man dann die Datei nach dem folgenden Schema:
OPEN 1,8,2,"filename"
und startet das Programm mit GOTO 0.
Wenn die Ausgabe beendet ist, wird mit der RUN/STOP-Taste die Endlosschleife verlassen.
(Reinhard Abdel-Hamid)

Speicherblockverschiebung — Blitzschnell (C 64)

```
1 poke95,al:poke96,ah:poke90,el:poke91,e
h:poke88,nl:poke89,nh:sys41919
11 rem beispiel:
12 poke95,0:poke96,160:poke90,0:poke91,1
92:poke88,0:poke89,192:sys41919
13 rem ↑↑ basic ins ram ↑↑
```

Dieses Programm dient zur Übertragung von Speicherblöcken. Die Variablen mit L sind also jeweils das Low-Byte, die mit H das High-Byte der Adresse.
Sie lassen sich für eine Adresse X so berechnen:
AL=X-256*INT(X/256): AH=INT(X/256)
Dieser Einzeiler benützt die Blockverschieberoutine des ROMs. Er ist zum Beispiel nützlich, um das Basic beziehungsweise das Betriebssystem vom ROM ins RAM zu verlegen oder um den Zeichengenerator zu kopieren (!).
(Jens Baas)

DI-AS, ein grafischer Disassembler für C 64

```
5 print"■■1■b■q■(■HP■■ :"peek(3)"■■ ":pok
e41,5:geta$:poke3,peek(3)-(a$="■")+(a$="
■"):sys1024:goto5
```

Der Speicherbereich des C 64 läßt sich in 256 Seiten (Pages) à 256 Byte aufteilen. DI-AS interpretiert die Speicherinhalte als Bildschirmcode und stellt die 64 KByte des C 64 Seite für Seite auf den Zeilen 7-13 des Bildschirms dar. Die Seitennummer wird am oberen Bildschirmrand angezeigt.
Die Bedienung:
Bevor das Programm geladen und gestartet wird, muß mit SYS64738 unbedingt ein Reset durchgeführt werden, da sonst möglicherweise die Zeiger der Zeropage nicht korrekt initialisiert werden. Es empfiehlt sich, den Bildschirm mit CLR zu löschen, um die Übersichtlichkeit zu erhöhen.
Das Programm wird mit RUN gestartet.
Dem Benutzer stehen nun folgende Optionen offen:
1. Mit CURSOR-RIGHT im Speicher vorwärts blättern
2. Mit CURSOR-DOWN im Speicher rückwärts blättern
3. Mit R/S POKE3, NR :CLR: RUN die Seite NR betrachten
Fehlerbehandlung:
Es stehen umfangreiche Fehlerbehandlungen zur Verfügung: Der Versuch, Seiten kleiner als 0 oder größer als 256 anzusehen wird mit »illegal quantity error in 5« quittiert. Mit CLR: RUN wird das Programm normal wiedergestartet.
Beachten Sie sorgfältig die Hinweise zum korrekten Eintippen des Programms!

DIE TOP 10

zum Einzeiler-Wettbewerb geboren war. Und diese Flut hat noch kein Ende.

Gehen Sie die Anwendungsbeispiele durch, um sich mit der Speicheraufteilung Ihres C 64 vertraut zu machen. Da das Programm die 80 Zeichen des Basic-Editors benötigt, muß der Cursor erst eine Zeile hochgefahren werden, bevor diese mit RETURN in den Programmspeicher übernommen wird.

Steuerzeichen:
Reverses s = Home
Reverses r = RVSON
Reverses R = RVSOFF
hinter dem P: Commodore O
die Restlichen: Cursor links, rechts, unten

Anwendungsbeispiele:
Wenn Sie das Programm zum ersten Mal starten, so wird Ihnen wahrscheinlich Seite 170 angezeigt. Drücken Sie solange auf CURSOR-DOWN, bis Sie bei Seite 0 angelangt sind. Falls Sie zu weit gefahren sein sollten, und das Programm den Fehler meldet, so starten Sie es einfach noch einmal. Sie sehen jetzt die Zeropage als Bildschirmcodes. Nun gehen Sie die Seiten Schritt für Schritt durch. Dabei können Sie beobachten:

Seite 0	: In Zeile 5 die Real-time Jiffy clock, die vom Betriebssystem versorgt wird.
Seite 1	: Unterste Zeilen = der Mikroprozessor System-Stack
Seite 2	: Oberste 2 Zeilen = Input Buffer. Hier stehen Ihre letzten Eingaben.
Seite 4-7	: Der Bildschirmspeicher selbst bildet sich ab. Halten Sie das Programm an, schreiben ein paar Sätze in die untere Bildschirmhälfte und schauen, was sich verändert hat. Außerdem zu sehen sind die Sprite-Pointer.
Seite 8	: Hier steht unser Basic-Programm. Erkennen Sie es wieder?
Seite 160	: Das CBM-BASIC-ROM.*
Seite 161	: Die Liste der Basic-Befehle*
Seite 162	: Die Fehlermeldungen*
192-207	: Leerer RAM für Ihre Maschinenprogramme
Seite 208	: Der VIC. Was da so flackert ist das VIC-Control-Reg. D011 und das Read-Raster-Reg. D012.
212-215	: Der SID
216-219	: Ein wüstes Bild. Das soll der Farb-RAM sein? Nur die Low-Nibbles behalten ihre Werte!!
220-223	: Die CIA. Beobachten Sie die Timer.
224-227	: Basic-Befehle: open, close, sin, cos, tan und andere
228-256	: Das Betriebssystem.*

*Anmerkung: Manches läßt sich erst entdecken, wenn durch gleichzeitiges Drücken von SHIFT und COMMODORE der Zeichensatz umgeändert wird. Nun nehmen Sie doch einmal Simons Basic oder Spiele und schauen, wo der Text, die Befehle etc. stehen!

(Andreas Carl)

Scrollen mit bleibendem Text (C 64)

```
1 fori=40960to49151:pokei,peek(i):next:f
ori=57344to65535:pokei,peek(i):next:poke
59639,10:poke1,53
```

Dieses kleine Programm bewirkt, daß nicht mehr der gesamte Bildschirm gescrollt wird. Wieviel Zeilen am oberen Rand fest stehen bleiben, bestimmt die Zahl, die Sie in 59639 POKEn. So kann man eine Information, die für ein Programm wichtig ist, auf dem Bildschirm festhalten. Oder die Kopfzeilen einer Tabelle, etc.

Mein Beispiel mit 0 gePOKEt hält eine Zeile fest. Eine 1 würde 2 Zeilen festhalten, etc. Abgestellt wird das Programm einfach durch »POKE 1,55«. Es muß noch darauf hingewiesen werden, daß dieses Programm, obwohl es so kurz ist, sehr lange bis zur ersten Ausführung braucht.

(Peter Eckart)

Maschinenprogramme abspeichern (C 64)

```
1 sys(57812)a$,x:poke193,ls:poke194,hs:p
oke174,le:poke175,he:sys62957
```

Mit folgendem Einzeiler können Maschinen-Programme sehr einfach gespeichert werden: (ohne Basic-Pointer zu verstellen)

Für die Berechnung von LE und HE muß die Endadresse + 1 genommen werden.

Als erstes Beispiel soll ein Programm von 20000 — 22000 unter dem Namen »BEISPIEL 1« auf Diskette gespeichert werden.

SYS(57812)"BEISPIEL 1", 8:POKE193,32:POKE194,78:POKE174,241:POKE175,85:SYS62957

Als zweites Beispiel soll ein Programm von $C000 — $C37E unter dem Namen "BEISPIEL 2" auf Kassette gespeichert werden.

SYS(57812)"BEISPIEL 2",1:POKE193,0:POKE194,192:POKE174,127:POKE175,195:SYS62957

Falls der Einzeiler infolge eines längeren Programm-Namens zu lang werden sollte, können die POKE-Befehle selbstverständlich durch P, Shift O abgekürzt werden.

(Markus Eicher)

Das kürzeste Heimcomputerorgelprogramm der Welt (C 64)

```
1 print"  )om x T) rm e T¬   P  p l d TN
  P%K r s)? m a TH1 d T- r PM r PPV r  ":s
ys4↑5
```

Um in Basic ein Maschinenspracheprogramm einzulesen, schreibt man normalerweise einen Basic-Lader, der in einer FOR/NEXT-Schleife mehrere DATAs einliest und in einen freien Speicherbereich POKEt, um dann das Programm mit SYS zu starten. Da diese Befehle und die DATAs sehr viel Platz benötigen, können so in einer Zeile kaum umfangreiche Programme geschrieben werden. Deshalb haben wir eine Möglichkeit entwickelt, den Basic-Lader zu umgehen: Wir PRINTen das Maschinenspracheprogramm in die erste Bildschirmzeile und starten das Programm mit SYS 1024. Mit diesem Trick ist es uns gelungen, ein Programm von 45 Maschinensprachebytes in einer Basic-Zeile zu laden und zu starten.

Das Programm »MICROSOUND« ist das, unseres Wissens, kürzeste Heimcomputerorgelprogramm der Welt. Es ist 86 Basic-Bytes lang und erzeugt ein Maschinenspracheprogramm von 45 Byte. Davon werden 14 Byte für Grafik-Effekte und 3 Byte für eine Programmende-Abfrage genutzt. Da der Einzeiler bereits beim geringsten Tippfehler nicht mehr richtig funktioniert, haben wir zum Vergleich den normalen Basic-Lader und das Monitor-Listing mit Erklärungen beigefügt.

Das Musik-Programm funktioniert im Wesentlichen dadurch, daß die Zahl der gedrückten Taste (PEEK (203)) mit 63 AND-verknüpft in das Hi-Byte der 1. Stimme des SID 6581 gebracht wird. Der Grafik-Effekt kommt durch einen Vergleich mit dem Rasterregister in 53266 zustande.

MICROSOUND ist für den Commodore C 64 entwickelt. Das Programm wird mit RUN gestartet und mit INST/DEL beendet. Es können ohne weiteres Basic-Zeilen angefügt werden, die dann nach INST/DEL abgearbeitet werden. Wenn das Maschinenspracheprogramm in der erste Zeile nicht mehr sichtbar sein soll, muß vorher die Zeichenfarbe

EINZEILER-WETTBEWERB:

DIE TOP 10

auf die Bildschirmfarbe umgestellt werden. Der Basic-Lader kann auch ohne weiteres auf Bereiche außerhalb des Bildschirms gesetzt werden, indem man die Variable »anfang« entsprechend ändert. Alle Tasten, außer CTRL, COMM., SHIFT und RESTORE sind mit Noten belegt, die überwiegend so angeordnet sind, daß die linken Tasten tiefere Töne spielen als die rechten. Die Funktions- und Cursor-Tasten sind Bässe.

Tips zur Eingabe dieses Programms:
1. Schalten Sie vor der Eingabe durch SHIFT/COMM. auf Kleinschrift.
2. Geben Sie ? statt print und sY4(Pfeil n. oben)5 statt sys1024 ein und verzichten Sie auf unnötige Leerzeichen.
3. SPACEs bitte ohne COMM. oder SHIFT-Tasten zu drücken eingeben. Ausnahme: Das letzte SPACE **muß** ein SHIFT/SPACE sein.
4. Achten Sie genau auf Groß- und Kleinschreibung! Das Programm kann beim kleinsten Tippfehler abstürzen!

MICROSOUND als Basic-Lader:

```
10 rem***microsound***
20 anfang=1024
30 for x=0 to 44:read a:poke an+x,a:next
40 sys an
100 data 169, 143, 141, 24, 212, 169, 240, 141, 5, 212, 238
110 data 32, 208, 160, 16, 140, 4, 212, 206, 32, 208, 165
120 data 203, 240, 19, 41, 63, 141, 1, 212, 200, 140, 4
130 data 212, 173, 18, 208, 205, 18, 208, 208, 214, 240, 246, 96
```

Der Basic-Lader hat gegenüber dem Einzeiler den Vorteil, daß man ihn leichter abtippen und ohne weiteres in einen anderen freien RAM-Bereich verlegen kann. Wenn Sie richtig abgetippt haben, müssen beide Programme identisch arbeiten.

Monitor Listing:

```
0400 lda  #$8f;
0402 sta  $d418;      Lautstärke (si+24)
0405 lda  #$f0;
0407 sta  $d405;      Halten d. Tones (si+5)
040a inc  $d020;      Rahmenfarbe um 1 erhöhen
040d ldy  #$10;
040f sty  $d404;      Wellenform (si+4) = 16
0412 dec  $d020;      Rahmenfarbe um 1 vermindern
0415 lda  $cb;        gedrückte Taste (peek(203))
0417 beq  $042c;      wenn $cb=null (inst/del gedrückt)
                      dann Ende
0419 and  #$3f;       wenn $cb=64 dann akku=0
041b sta  $d401;      akku nach hi-byte von Stimme 1 (si+1)
041e iny;
041f sty  $d404;      Wellenform (si+4) = 17
0422 lda  $d012;      Rasterregister (v+18)
0425 cmp $d012;       (Grafik-Effekt durch registergesteuerte Verzögerung)
0428 bne  $0400;
042a beq  $0422;
042c rts;             Basic-Rücksprung
```

(Klaus und Stefan Holthausen)

Elektronisches Klavier für den VC 20

```
1 poke36878,15:geta$:a=val(a$):poke36876
,127+(10-(a>1)+(a>3)+(a>5)*.4+(a>6)*.3+(
a>7)*.6)*a:goto1
```

Das Programm läuft nur auf dem VC 20 und verwandelt den Computer in ein elektronisches Klavier. Die Tasten 1 bis 8 dienen dabei als Klaviatur, auf der man in etwa eine D-Dur Tonleiter spielen kann.

Der Computer gibt beim Niederdrücken der Tasten 1 bis 9 jeweils einen kurzen Ton aus. Die Zahlen 1 bis 8 entsprechen dabei den Tönen einer D-Dur Tonleiter, also d, e, fis, g, a, h, cis, d. Das Betätigen der Taste 9 ergibt zwar auch einen Ton, der sich aber nicht mehr in die Tonleiter einreiht. Andere Tasten haben, außer der RUN/STOP-Taste, keine Funktion.

Basic-Programm:

```
1 POKE36878,15:GETA $:A=VAL(A$):POKE36876,127+(10—(A>1)+
(A>3)+(A>5)*. 4+(A>6)*. 3+(A>7)*.6)*A:GOTO1
```

Beschreibung des Programms:
POKE36878,15: Die Lautstärke des VIC-Chips wird gesetzt.
GETA$:A=VAL(A$): Der Variablen A wird ein Wert zwischen 0 bis 9 zugewiesen, je nachdem, welche Taste man drückt. Das Einlesen der Variablen A$ verhindert einen SYNTAX ERROR bei Betätigung einer Buchstabentaste.
POKE36876,...: Die Tonhöhe wird gesetzt, und ein Ton wird ausgegeben, wenn A>0. Die Ausdrücke (A>1), (A>3), und so weiter bewirken eine Anpassung des Tonregisters an die Tonleiter, da sie bei erfüllter Bedingung den Wert —1 ergeben und im anderen Fall 0 sind.
GOTO1: Das Programm wird so lange wiederholt, bis die RUN/STOP-Taste gedrückt wird.

(Joachim Günther)

Track-Zerstörer: Kopierschutz (C 64/VC 20)

```
1 open1,8,15:open2,8,2,"#":print#1,"u1 2
 0";t;0:print#1,"m-e"chr$(163)chr$(253)
```

Diese Zeile produziert auf dem gewünschten Track der Diskette (Variable T) einen READ ERROR 21. Das bedeutet einen relativ sicheren Kopierschutz. (Jörg Wegmeyer)

Formatierte Ausgabe (C 64/VC 20)

```
20 b=abs(a):printtab(int(log(b-(b=0))*.4
3429448188)*(b>=1)+int(-b)*(b<1)+22);a
```

Dieser Einzeiler gibt Zahlen, egal welcher Größe und unabhängig vom Vorzeichen, rechtsbündig aus.

Die Zahl 22 gibt die Spalte minus 2 für die Kommastelle an. Wollen Sie zum Beispiel die Kommata in der Spalte 30 haben, so müssen sie statt 22 dann 28 (30-2) eingeben.
A ist die Zahl die Sie ausgeben wollen.

Funktionsweise der Programmzeile:

Die Berechnung der Tabulatorfunktion unterteilt sich in die Bereiche für A größer-gleich eins, und A kleiner eins. Wobei A die Zahl ist, die ausgegeben werden soll.
Für A größer-gleich eins wird von A der Zehnerlogarithmus berechnet und von der Spalte abgezogen, bei der der Dezimalpunkt stehen soll. Da der Commodore 64 nur den Logarithmus zur Basis e berechnen kann, muß das Ergebnis mit der Konstante .43429448188 multipliziert werden. Dadurch erhält man den Zehnerlogarithmus.

Für diesen Teil der Berechnung ist nur der folgende Teil der Zeile maßgebend: INT(LOG(B-(B=0))*. 43429448188)*(B>=1)

Für Zahlen kleiner eins wird zu der Spalte eins dazuaddiert. Für Null entsteht ein Sonderfall, da die Null noch vor dem Komma stehen muß. Für diesen Teil ist der folgende Ausdruck maßgebend: INT(—B)*(B<1) (Volker Walter)