»Der Sumpf«

Unser Beitrag über die Raubkopierer-Szene in der 64'er 6/84 hat weite Wellen geschlagen. Nachstehend bringen wir einen in mehrfacher Hinsicht bemerkenswerten Leserbrief zu diesem Thema — und unsere Antwort darauf.

Als Vorwegnahme meiner Meinung zu Ihrem Artikel »Der Sumpf« muß ich Ihnen sagen, daß ich selbst Raubkopierer bin. Ich bin 15 Jahre und mir voll und ganz bewußt, daß mein Tun illegal ist. Jedoch läßt sich von dieser »kleinen« Taschengeldaufbesserung (zirka 100 Mark pro Monat) ganz gut leben. Ich möchte aber Ihre These, »die Raubkopierer sind für die hohen Softwarepreise verantwortlich« widerlegen. Denn was war eher, die Software oder die Raubkopie? Es ist doch logisch, daß man ohne die teure Software, die urheberrechtlich geschützt ist, keine Raubkopie machen kann. Ich stimme Ihnen voll und ganz zu, daß es (auf gut deutsch gesagt) eine Schweinerei ist, daß Kinder genau dieselbe Strafe bekommen wie Profihacker. Ich will Ihnen für die wirklich gute Idee der billigen Kleinanzeigen keine Vorwürfe machen, sie ist nur ein weiterer guter Bestandteil Ihres sonst hervorragenden Heftes, aber meinen Sie nicht, daß Sie damit Raubkopierer etwas animieren? Meinen Brief unterzeichne ich mit

H. Acker

Es ist richtig, wenn Sie meinen, nicht die Raubkopierer allein wären für die hohen Software-Preise verantwortlich zu machen. Dazu kommt natürlich auch der oft enorme Entwicklungsaufwand, der in so einem Programm steckt. Nicht zuletzt wollen auch der Hersteller, die Distributoren und Händler einen kleinen Gewinn nach Hause tragen.

Es ist traurig aber wahr, daß die kleinen (sprich jungen) Raubkopierer mit demselben Strafmaß zu rechnen haben wie die großen. Die Firmen gehen in dieser Hinsicht aber nach dem Grundsatz: »Wehret den Anfängen« vor. Ein amerikanischer Datenschützer sprach einmal von der »fehlenden Moral« im Bereich der Software. Es ist in der Tat unter den Jugendlichen ein schwereres Vergehen, jemandem die Cola wegzutrinken, als einem relativ anonymen Entwickler das geistige Eigentum in Form eines Spiels oder Anwenderprogramms zu klauen. Sicher, die Materie, das Programm ist etwas reell nicht Greifbares; außer auf der Diskette (und das sollte man ja tunlichst vermeiden).

Die von Ihnen angesprochenen Kleinanzeigen in unserer Zeitschrift sind da nur ein Spiegel der Gesellschaft. Geplant waren sie, um dem Leser eine billige Möglichkeit zu bieten, Kontakte zu knüpfen, Erfahrungen auszutauschen, nicht mehr benötigte Hard- und Software zu verkaufen und vieles andere mehr. Nicht beabsichtigt war natürlich, den Raubkopierern ein preisgünstiges Forum für ihre illegalen Geschäftspraktiken zu sein. Wir sind um Abhilfe bemüht, aber wie läßt sich eine Anzeige über 100 selbstgeschriebene Programme von 100 raubkopierten unterscheiden, wenn dies nicht im Text erscheint. Außerdem gibt es in einigen Bereichen, wie zum Beispiel dem Lehrberuf, sogenannte »Public Domain Software«, die der Öffentlichkeit zugänglich gemacht wird.

Es ist nicht unsere Aufgabe und entspricht auch nicht unserem Selbstverständnis, der Vorreiter für die Industrie zu sein. Doch es ist unser Bestreben, ein in allen Belangen »sauberes« Magazin für den Computer-Fan zu machen. Dazu gehört es auch, rechtzeitig zu warnen, Mißstände aufzuzeigen und Hilfestellung zu leisten. Und diese Art der Taschengeldaufbesserung ist nicht in Ordnung. Die jungen Schüler und Studenten, denen es gelingt, den Software-Schutz fremder Programme zu knacken, eigene Kommentare und Veränderungen im Programm anzubringen, die verstehen doch etwas vom Programmieren. Diese Energien und dieses Know-how ließe sich doch wesentlich sinnvoller für die Entwicklung eigener, guter Programme einsetzen. Es gibt genug Programme auf dem Markt, die es nicht einmal wert sind, kopiert zu werden. Dagegen sollte man mit seinem ganzen Können angehen. Gerade in Deutschland sind wir in der Computertechnologie und der Softwareerstellung noch um Jahre hinter den USA, Großbritannien oder Japan. Es ist an der Zeit, denen zu zeigen, wo der Bartel den Most holt. Gute Software erstellen, zu einem fairen Preis, da liegt Eure Chance. (aa)

FEHLERSUCH

2. TEIL

Wie schon in der vorletzten Ausgabe des 64'er-Magazins angedeutet wurde, sind Eintipp-Fehler in DATA-Zeilen besonders tückisch. Und das hauptsächlich aus zwei Gründen. Erstens zeigt eine eventuelle Fehlermeldung nie auf den wirklichen Fehler, sondern in der Regel auf die Einlese-(READ-) Schleife dieser Daten. Zweitens, und das ist noch viel unangenehmer, kann es vorkommen, daß es gar keine Fehlermeldung gibt, sondern der Computer einfach aussteigt und »abstürzt«. Hoffentlich haben Sie das mühsam eingegebene Programm vor dem ersten RUN abgespeichert!

Gerade Anfänger haben jetzt Schwierigkeiten. Und deshalb möchte ich mich hauptsächlich an diese wenden.

Als erstes möchte ich Ihnen empfehlen, noch einmal das »Handbuch« des VC 20/C 64 in die Hände zu nehmen. Lesen Sie noch einmal durch, was dort über den READ/DATA-Befehl geschrieben steht. Auch ich bin diesem ominösen READ-Befehl anfangs möglichst aus dem Weg gegangen und stehe ihm jetzt noch manchmal mit Mißtrauen gegenüber. Und das liegt an den oben genannten zwei Punkten.

Oft sind nämlich diese DATA-Werte nichts anderes als codierte Maschinensprache. Und dann kann sogar eine einzige falsch abgetippte Ziffer zum absoluten Kollaps führen. Aber wenn es sich um Maschinensprache handelt, haben alle DATA-Werte einige gemeinsame Merkmale: Erstens sind sie nicht negativ, zweitens sind sie niemals größer als 256 und drittens sind es immer ganze Zahlen. Diese Merkmale können wir benutzen um eine kleine Prüfroutine zu entwickeln, die einen, aufgrund eines Eingabefehlers, falschen DATA-Wert sofort sichtbar macht, wenn er gegen diese Merkmale verstoßen sollte.

Dieses folgende kleine Programm sollten Sie vor oder hinter das zu überprüfende Programm

E IN BASICPROGRAMMEN

Immer wieder erreichen uns Anrufe von Lesern, die (besorgt) anfragen, ob in diesem oder jenem Listing Fehler bekannt sind. Meistens handelt es sich jedoch um Eintipp-Fehler, vom Leser selbst verursacht. Und meistens liegen die tückischen Fehler innerhalb von DATA-Zeilen. Was tun?

schreiben. Nach abgeschlossener Prüfung kann es ruhig wieder gelöscht werden. Hier das Programm:
50000 READ A$; PRINT "ZEILE "PEEK(64)*256+PEEK(63);A$
50010 POKE198,0: WAIT 198,1:GOTO 50000

Starten Sie das Programm in diesem Fall mit RUN 50000. Sie werden jetzt den ersten DATA-Wert am Bildschirm sehen, davor die Zeilennummer, in der dieser Wert steht. Wenn Sie dann eine Taste drücken , zum Beispiel die große Leertaste, wird der nächste Wert sichtbar, direkt darunter. Bei fortwährendem Drücken der Leertaste, rasen die Zahlen sehr schnell über den Bildschirm. Für den ersten Durchlauf sollten Sie das ruhig machen. Man sieht dann sofort, ob eine Zahl (unzulässiger Weise?) länger als drei Ziffern ist. Falls dies der Fall ist, können Sie sehr schnell im Originallisting nachschauen, ob der Wert in Ordnung ist. Falls der Fehler so jedoch nicht erkannt werden kann, bleibt nichts anderes übrig, als das Programm noch einmal zu starten und diesmal jeden einzelnen Wert mit dem Original zu vergleichen. Machen Sie sich aber keine Sorgen über die Fehlermeldung »OUT OF DATA ERROR» die unweigerlich am Ende auftaucht. Sie zeigt hier lediglich, daß alle DATAs angezeigt wurden.

Apropos Fehlermeldung. »OUT OF DATA ERROR« oder »ILLEGAL QUANTITY ERROR«. Wenn nach dem Starten des Hauptprogramms eine dieser Fehlermeldungen kommt, können Sie fast todsicher davon ausgehen, daß ein DATA-Wert falsch eingetippt wurde. Das »fast« nur deswegen, weil Sie eventuell die FOR-NEXT-Schleife, in der ein READ-Befehl steht, falsch eingegeben haben. Die Bedeutung dieser Fehlermeldungen sind im schon erwähnten Handbuch erklärt.

Prüfsummliste. Das sind die Prüfsummen des in dieser Ausgabe abgedruckten Mailbox-Programms.

```
PRUEFSUMMENLISTE
BLOCKGROESSE 30

ZEILE   ANZAHL    SUMME      KEIN POKE?
-------------------------------------
 10030
 10040     30       3564
 10040
 10060     60       6829
 10090     90      10532
 10100
 10100
 10110    120      14114
 10140    150      16953
 10150
 10160    180      20668
 10170
 10190    210      24477
 10210    240      27726
 10220
 10240    270      31445
-------------------------------------
GESAMT    282      32970
```

Listing. Diese Prüfroutine soll in Zukunft die Suche nach DATA-Fehlern erleichtern. Näheres im Text.

```
62000 REM------------------------------
62010 REM PRUEFSUMMENLISTE
62020 REM------------------------------
62030 BL$="                "
62040 INPUT"BLOCKGROESSE";BG
62050 OPEN1,4:CMD1
62060 PRINT"PRUEFSUMMENLISTE"
62070 PRINT"BLOCKGROESSE";BG
62080 GOSUB62200
62090 PRINT"ZEILE  ANZAHL   SUMME    KEI
N POKE?"
62100 GOSUB62200
62110 RESTORE
62120 READA$:A=VAL(A$)
62130 IFA$="*"THENGOSUB62200:RC$="":GOSU
B62210:PRINT"GESAMT" RC$:GOTO62300
62140 AN=AN+1:S=S+A:B=B+1
62160 IFA$<>MID$(STR$(A),2)THENPRINTPEEK
(64)*256+PEEK(63);SPC(20);A$
62170 IFA>256THENPRINT PEEK(64)*256+PEEK
(63) SPC(20);A$
62175 IFA<>INT(A)THENPRINT PEEK(64)*256+
PEEK(63) SPC(20);A$
62180 IFAN=BGTHENGOSUB62205  :PRINTRC$:A
N=0
62190 GOTO62120
62200 FORI=1TO10:PRINT"——";:NEXT:PRINT
:RETURN
62205 REM------------------------------
62206 RC$=""
62207 Z$ =STR$(PEEK(64)*256+PEEK(63))
62208 RC$=RC$+LEFT$(BL$,6-LEN(Z$))+Z$
62210 B$=STR$(B):S$=STR$(S)
62220 RC$=RC$+LEFT$(BL$,6-LEN(B$))+B$
62240 RC$=RC$+LEFT$(BL$,10-LEN(S$))+S$
62250 RETURN
62260 DATA*
62300 PRINT#1:CLOSE1
READY.
```

Ich hoffe, daß Sie mit dieser kleinen Routine in Zukunft schneller diese lästigen Fehler finden. Um Ihnen aber die Fehlersuche noch etwas weiter zu erleichtern, habe ich ein weiteres Programm geschrieben. Wir werden in den kommenden Heften bei Programmen mit vielen DATA-Werten eine spezielle Prüfsummenliste mit abdrucken, die es gestattet, alle DATA-Werte blockweise zu überprüfen. Dazu müßen Sie dann jedesmal das hier abgedruckte Programm (siehe Listing) an das zu testende Programm anhängen (entweder mit MERGE oder durch Abtippen) und laufen lassen. Damit kann ein Tippfehler auf wenige DATAs eingegrenzt werden. Wenn Sie das nebenstehende Programm laufen lassen, sollten Sie am Bildschirm die gleichen Werte erhalten wie die dann abgedruckte Prüfsummenliste. Eine Abweichung kann auf einen Eingabefehler hinweisen (muß aber nicht). Die vier Spalten der Prüfsummenliste haben folgende Bedeutung:

Ganz links steht unter ZEILE die Zeilennummer des letzten DATA-Wertes des jeweiligen Blocks, daneben die Anzahl der bisher gelesenen DATAs, rechts davon die Summe aller bisher gelesenen Werte und ganz rechts eine mögliche Fehlerquelle, das heißt, hier wird ein Wert angezeigt, wenn er größer als 256 ist oder einer gebrochenen Zahl, aber auch, wenn er keine Zahl ist und Buchstaben enthält.

Die als Beispiel abgedruckte Prüfsummenliste sieht so aus, wenn Sie den DATA-Tester an das in diesem Heft veröffentlichte Mailbox-Programm angehängt haben. Es wurden hier jeweils 30 DATAs gelesen und deren Prüfsumme errechnet. Man sieht auch, daß sich zum Beispiel in Zeile 10030 (die erste Zahl in der Prüfsummenliste) ein Leerstring befindet. An dieser Stelle werden Sie im Mailbox-Programm zwei Kommata finden. Das gleiche gilt für die Zeilen 10040,10100,10150,10170 und 10220. Insgesamt werden 282 DATAs gelesen. Die Gesamtsumme beträgt 32970. Die Spalte unter »TEXT« bleibt leer, weil in den DATAs weder Text noch negative Zahlen vorkommen.

Ich hoffe, daß Sie mit diesen Mitteln mögliche Eingabefehler schneller finden und somit nicht vorschnell Frust aufkommt. Aber einen Fehler werden Sie oft vergebens suchen: Das ist der, den unser Fehlerteufel hinzaubert. Aber dem werden wir in der Redaktion schon das Leben schwer machen (umgekehrt allerdings genauso). (gk)