RUN Preisträger

C64

Spaghetticode entwirrt

Viele Befehle in eine Zeile zu quetschen spart Speicherplatz. Der Nachteil dabei: Programme werden unübersichtlich. Recom64 zieht den Zeilenverhau wieder auseinander.

Ein Rekompaktor — was ist das? Es handelt sich dabei um das Gegenteil eines Kompaktors: Ein Quellprogramm wird nicht komprimiert, das heißt, es werden nicht möglichst viele Befehle in eine Zeile gestopft, sondern im Gegenteil die einzelnen Zeilen auseinandergezogen.
Im einzelnen bedeutet das, aus:

```
10 TI$="000000":FOR I = 1 TO 25:
PRINT I, I*I:NEXT:PRINT TI/60
"SEKUNDEN"
```

wird:

```
100 TI$="000000"
110 FOR I=1 TO 25
120 : PRINT I, I*I
130 NEXT
140 PRINT TI/60 "SEKUNDEN"
```

Außerdem wird das Programm ab einer beliebigen Startzeile in frei wählbaren Schritten neu numeriert. Dabei werden auch die Zeilennummern nach GOTO, GOSUB, ON. . .GOTO, THEN umnumeriert.
Die Anwendung von Recom64 ist immer dann sinnvoll, wenn man sich in einem Programm nicht mehr auskennt, weil man im Eifer des Programmierens zu viele Befehle in eine Zeile gepackt und gar nicht an schulbuchmäßige Strukturierung gedacht hat. Auch zur Analyse von fremden Programmen ist der Rekompaktor gut geeignet.
Recom64 liest ein Programm von Diskette ein und speichert es nach der Bearbeitung wieder unter einem frei wählbaren Namen ab. Nach dem Start gibt man zuerst den Namen des Quell- und des Zielfiles ein. Darauf erfolgt eine Abfrage, ob die Schleifen formatiert werden sollen. Dabei wird der Programmteil innerhalb einer FOR. .NEXT-Schleife um zwei Spaces eingerückt. Außerdem müssen die Startadresse (normal 2049), die Startzeilennummer und die Schrittweite, mit der renumeriert werden soll, eingegeben werden.
Das Rekompaktieren wird in vier Durchgängen (Passes) durchgeführt: Im ersten Pass wird das Quellprogramm von Diskette eingelesen, wobei die jeweils aktuelle Zeilennummer ausgegeben wird. Pass 2 zerlegt alle Quellzeilen in einzelne Befehle, die dann jeweils zu einer selbständigen Zeile werden. Hier und in Pass 3, wo die Sprünge umgerechnet werden, erfolgt auch die Renumerierung. Falls man am Anfang bei „Schleifen formatieren (j/n)" j oder y eingegeben hat, werden in Pass 4 die Schleifen formatiert.

Neues File

Danach wird das erzeugte Programm mit der eingegebenen Startadresse auf Diskette abgespeichert. Sollte dabei ein Fehler auftreten, wird ungefähr 30 Sekunden lang auf einen Tastendruck gewartet, dann erfolgt ein neuer Save-Versuch. Falls der Fehler 'File exists' oder 'Disk full' auftritt, muß man während dieser Zeit die Diskette wechseln.
Das oben Gesagte gilt für alle auftretenden Fehler: Der Fehler wird ausgegeben, 30 Sekunden oder bis zu einem Tastendruck gewartet, und dann geht es — dem Fehler entsprechend — im Programm weiter.
Variablen:
Arrays:
s$(x): Zeilen des Quellprogramms
s(x): dazugehörige Zeilennummern
u(x): Zeilennummern des Zielprogramms, die jeweils den Zeilennummern s(x) des Quellprogramms entsprechen.
d$(x): Zeilen des Zielprogramms
d(x): dazugehörige Zeilennummern
Wichtige Strings:
sn$: Name des Quellprogramms
dn$: Name des Zielprogramms
d$: enthält den Floppystatus (wie DS$ beim Basic 4.0)
err$: enthält den zuletzt aufgetretenen Fehler
sp$: wird benutzt beim Einrücken innerhalb von FOR. .NEXT-Schleifen
ze$: enthält den zuletzt aus s$(x) gelesenen Befehl
Wichtige numerische Variablen:
ss: Anzahl der Zeilen des Quellprogramms
dd: Anzahl der Zeilen des Zielprogramms
sa: Startadresse des Zielprogramms
sz: Startzeilennummer
sw: Schrittweite beim Renumerieren
ga: Geräteadresse der Floppy, definiert in Zeile 1320
ja/jn: alte/neue Sprungadresse beim Renumerieren
d: Fehlernummer bei Floppy
f1: Flag für REM oder THEN
f2: Flag für Anführungszeichen
f3: Flag für Stringende
i,m,mm,nn: Laufvariablen
yy,x,as: verschieden genutzte Variablen
Programmbeschreibung:
1000 Sprung zur Initialisierung
1040 Input-Unterprogramm
1090 Unterprogramm, das den Fehlerkanal liest und in den Variablen d und d$ übergibt

1130 Unterprogramm, das Lo- und Hi-Byte liest, den Wert errechnet und in ad zurückgibt
1160 Unterprogramm, das ein File bis zum nächsten chr$(0) liest und die gelesenen Daten in da$ zurückgibt
1200 Unterprogramm, das das alte Sprungziel ja im Array s(ss) sucht, und, falls gefunden, das neue Sprungziel jn zurückgibt oder, falls nicht gefunden, 'Undef'd Statement Error' ausgibt.
1310 Initialisierungsroutine, in der die Konstanten definiert und die Arrays dimensioniert werden.
1380 Parameter-Eingabe
1660 Pass 1: Einlesen des Quellfiles in die Arrays s$(ss) und s(ss) mit Hilfe der Unterprogramme ab 1090, 1130 und 1160
1840 Pass 2: Zerlegen des Quellprogramms in Array s$(ss) in seine Befehle und Speicherung derselben im Array d$(dd)
2130 Pass 3: Umrechnung der alten Sprungadressen der GOTO/GOSUB/THEN-Befehle; dabei:
2130 Suchen der Tokens von Sprungbefehlen
2310 Umrechnung von GOTO/GO TO/GOSUB und THEN mit Zeilennummer
2490 Prüfen, ob auf THEN eine Zeilennummer folgt; wenn ja, dann Sprung zur Umrechnung ab 2310
2560 Test, ob auf ON ein GOTO/GOSUB oder GO TO erfolgt. Wenn ja, dann Sprung nach 2750 oder 2670; wenn nein, dann Ausgabe von 'On without Gosub Error'
2660 Prüfen, ob nach ON. . .GO ein TO folgt. Wenn ja, dann Sprung zu 2750; wenn nein, dann Ausgabe von 'Go without To Error'
2750 Heraussuchen der einzelnen Sprungadressen und Umrechnung derselben, danach Rücksprung nach 2190 zum weiteren Suchen nach Tokens.

FOR . . NEXT-Schleife eingerückt

2970 Test, ob nach 'GO' ein 'TO' folgt. Wenn ja, Sprung zur Umrechnungsroutine ab 2310; wenn nein, dann Ausgabe von 'Go without To Error'
3090 Pass 4: Einrücken der Texte zwischen FOR. .NEXT-Schleifen, dabei:
3090 Test, ob Formatierung gewünscht. Wenn nein, dann Sprung zur Save-Routine
3110 Setzen der Schachtelungstiefe auf 0
3120 Schleife, die den jeweils aktuellen d$(i) um fo Bytes aus sp$ einrückt, und dann Byte für Byte nach FOR/NEXT-Tokens durchsucht
3210 Falls Anführungszeichen oder REM, dann Flag setzen
3230 Wenn Flag gesetzt, nicht nach Token suchen
3240 Wenn Token für FOR gefunden, dann Schachtelungstiefe in fo um 2 erhöhen
3250 Wenn Token für NEXT gefunden und die Schachtelungstiefe noch größer als 0 ist, dann Schachtelungstiefe um 2 verringern und den aktuellen d$(i) um 2 Spaces weniger einrücken
3260 Rücksprung, nächstes Byte prüfen
3300 Saven des erzeugten Programms unter dem am Anfang eingegebenen Namen. Falls Floppyfehler, dann ausgeben und noch ein Save-Versuch.
3520 Ausgabe, wie viele Fehler beim Rekompaktieren aufgetreten sind; danach Programmende
3580 Fehler-Ausgaberoutine, die err$/err ausgibt
3600 Fehler-Ausgaberoutine, die d$/d ausgibt
Es empfiehlt sich, das Programm zu kompilieren, da die nicht kompilierte Version bei längeren Programmen wegen der häufig zuschlagenden Garbage-Collection recht langsam wird.

(Gert Döring)

```
1000 goto 1310                                  579
1010 :
1020 rem ***** subroutinen *****
1030 :
1040 poke 19,64                                 807
1050 input a$                                   407
1060 poke 19,0                                  577
1070 print                                      153
1080 return                                     142
1090 input#1,x1$,x2$,x3$,x4$                   1660
1100 d=val (x1$)                                585
1110 d$=x1$+","+x2$+","+x3$+","+x4$            3225
1120 return                                     142
1130 get #2,l$,h$                               939
1140 ad=asc (l$+chr$ (0))+256*asc               3156
     (h$+chr$ (0))
1150 return                                     142
1160 da$=""                                     593
1170 get #2,x$                                  793
1180 if x$<>"" then da$=da$+x$:goto             2505
     1170
1190 return                                     142
1200 for yy=1 to ss                            1105
1210 :if ja=s(yy) then jn=u(yy):                2165
     return
1220 next                                       130
1230 err$="Undef'd Statement Error"             2595
1240 err=d(i)                                   758
1250 gosub 3580                                 651
1260 jn=63999                                   933
1270 return                                     142
1280 :
1290 rem ***** initialisieren ******
1300 :
1310 dim s$(500),s(500),u(500),                 3095
     d$(1000),d(1000)
1320 ga=8                                       596
1330 sp$=":[20spaces]"                          1407
1340 rem 20 spaces
1350 :
1360 rem ***** parameter-eingabe ****
     *
1370 :
1380 print "[clr,down,6spaces]                  3161
     Recompactor[4spaces]by GD-Soft
      [ctrl h,ctrl n]"
1390 print "[2down] Source-Filename             2401
     [6spaces]: ";
1400 gosub 1040                                 595
1410 sn$=a$                                     684
```

```
1420 print "[down] Destination-          3282
     Filename : ";
1430 gosub 1040                          595
1440 dn$=a$                              669
1450 print "[down] Schleifen            3758
      formatieren (j/n) : ";
1460 poke 204,0                          804
1470 get fo$                             395
1480 if fo$<>"j" and fo$<>"y" and        3238
     fo$<>"n" then 1470
1490 poke 204,1                          836
1500 print fo$                           387
1510 print "[down] Startadresse          2797
     [2spaces]: 2049[4left]";
1520 gosub 1040                          595
1530 sa=val (a$)                         788
1540 print "[down] Startzeilennr.:       2694
      1000[4left]";
1550 gosub 1040                          595
1560 sz=val (a$)                         838
1570 print "[down] Schrittweite          2145
     [2spaces]: 10[2left]";
1580 gosub 1040                          595
1590 sw=val (a$)                         832
1600 print "[clr,down,6spaces]           3161
     Recompactor[4spaces]by GD-Soft
      [ctrl h,ctrl n]"
1610 print "[2down,3spaces]Sourcefile    2086
     [6spaces]: "sn$
1620 print "[down,3spaces]              2175
     Destinationfile : "dn$
1630 :
1640 rem *****          pass 1         *
     ****
1650 :
1660 print "[home,10down]" tab(          2202
     3)"Pass 1 :"
1670 open 1,ga,15                        874
1680 open 2,ga,0,sn$                    1063
1690 gosub 1090                          635
1700 if d then close 2:close 1:gosub     1422
     3600:run
1710 gosub 1130                          591
1720 gosub 1130                          591
1730 if ad=0 then close 2:close 1:       1553
     goto 1840
1740 gosub 1130                          591
1750 print "[home,10down]" tab(11)ad    1582
1760 ss=ss+1                             819
1770 s(ss)=ad                            740
1780 gosub 1160                          615
1790 s$(ss)=da$                          902
1800 goto 1720                           603
1810 :
1820 rem ***** pass 2 *****
1830 :
1840 print "[home,10down]" tab(          2405
     23)"Pass 2 :"
1850 for i=1 to ss                      1006
1860 :print "[home,10down]" tab(         2015
     31)s(i)
1870 :u(i)=sw*dd+sz                      1615
1880 :f1=0                               546
1890 :f2=0                               550
1900 :f3=0                               554
1910 :for m=1 to len (s$(i))             1262
1920 :x$=mid$ (s$(i),m,1)                1950
1930 :x=asc (x$+chr$ (0))                1415
1940 :if x$=":" and ze$<>"" then 2010    1755
1950 :if x=139 then f1=1                 1244
1960 :if x=143 then f1=1                 1402
1970 :if x=34 then f2=1-f2               1390
1980 :ze$=ze$+x$                         999
1990 :next m                             116
2000 :f3=1                               570
2010 :if f3=0 then if f1 or f2 then      1813
     1980
2020 :dd=dd+1                            886
2030 :d$(dd)=ze$                        1142
2040 :ze$=""                             539
2050 :d(dd)=sw*dd+sz-sw                 1957
2060 if f3=0 then next m                1205
2070 s$(i)=""                            774
2080 next i                              276
2090 if d(dd)>63999 then err$="Line      6649
      Increment too large !":err=0:
     gosub 3580:run
2100 :
2110 rem ***** pass 3 *****
2120 :
2130 print,"[home,13down]" tab(          2472
     3)"Pass 3 :"
2140 for i=1 to dd                       841
2150 :mm=.                               640
2160 :f1=.                               769
2170 :f2=.                               773
2180 :print "[home,13down]" tab(         1774
     11)d(i)
2190 :mm=mm+1                           1117
2200 if mm>len (d$(i)) then next i:      1958
     goto 3090
2210 yy=asc ( mid$ (d$(i),mm,1)+chr$     2201
     (0))
2220 if yy=34 then f2=1-f2:rem falls     1772
2230 if yy=143 then f1=1:rem anfuehru    1805
     ngszeichen oder rem,
2240 if f1 or f2 then 2190:rem dann n    1600
     icht pruefen
2250 if yy=137 or yy=141 then 2310:      2500
     rem goto/gosub
2260 if yy=167 then 2490:rem then-mit    1874
     /ohne zeilennr.?
2270 if yy=145 then 2570:rem on...       1676
2280 if yy=203 then 2980:rem go          1703
2290 goto 2190                           637
2300 rem goto/gosub
2310 ja=.                                519
2320 ja$=""                              599
2330 nn=.                                548
2340 nn=nn+1                             939
2350 nn$=mid$ (d$(i),mm+nn,1)           2211
2360 if (nn$<"0" or nn$>"9") and nn$<    3214
     >" " then 2390
2370 ja$=ja$+nn$                        1173
2380 goto 2340                           605
2390 ja=val (ja$)                        794
2400 gosub 1200                          571
2410 li$=left$ (d$(i),mm)               1597
2420 re$=mid$ (d$(i),mm+nn)             1922
2430 d$(i)=li$+str$ (jn)+re$            1882
2440 mm=mm+nn                           1022
2450 goto 2190                           637
2460 rem then
2470 rem erst pruefen ob zeilennummer
     , wenn ja, dann sprung zur
2480 rem adressenumwandlung (ab 2310,
      auch bei goto/gosub benutzt)
2490 nn=.                                549
2500 f4=.                                489
2510 nn=nn+1                             939
2520 nn$=mid$ (d$(i),mm+nn,1)           2211
2530 if (nn$<"0" or nn$>"9") and nn$<    3150
     >" " then 2190
2540 if nn$=" " then 2510:rem space u    1211
```

```
     eberlesen
2550 goto 2310                                  581
2560 rem on...goto/on...gosub
2570 mm=mm+1                                    912
2580 as=asc ( mid$ (d$(i),mm,1)+chr$           2165
     (0))
2590 if as=137 or as=141 then 2760             2197
2600 if as=203 then 2670                       1458
2610 if mm<len (d$(i)) then 2570               1498
2620 err$="ON without GOTO Error"              2837
2630 err=d(i)                                   758
2640 gosub 3580                                 651
2650 next i                                     276
2660 rem nach 'on...go' auf 'to' prue
     fen
2670 mm=mm+1                                    912
2680 as=asc ( mid$ (d$(i),mm,1)+chr$           2165
     (0))
2690 if as=164 then 2760                       1441
2700 if mm<len (d$(i)) then 2670               1562
2710 err$="GO without TO Error"                2715
2720 err=d(i)                                   758
2730 gosub 3580                                 651
2740 next i                                     276
2750 rem adressen einzeln holen und g
     leich umwandeln
2760 nn=.                                       549
2770 mm=mm+nn                                  1022
2780 nn=.                                       549
2790 ja$=""                                     599
2800 nn=nn+1                                    939
2810 nn$=mid$ (d$(i),mm+nn,1)                  2211
2820 if (nn$<"0" or nn$>"9") and nn$<          3413
     >" " then goto 2850
2830 ja$=ja$+nn$                               1173
2840 goto 2800                                  593
2850 ja=val (ja$)                               794
2860 gosub 1200                                 571
2870 li$=left$ (d$(i),mm)                      1597
2880 re$=mid$ (d$(i),mm+nn)                    1922
2890 d$(i)=li$+str$ (jn)+re$                   1882
2900 nn=nn-1                                    971
2910 nn=nn+1                                    939
2920 nn$=mid$ (d$(i),mm+nn,1)                  2211
2930 if nn$=" " then 2910                      1156
2940 if nn$="," then 2770                      1199
2950 if nn$=":" or mm+nn>len (d$(i))           4329
     then mm=mm+nn:goto 2190
2960 goto 2910                                  605
2970 rem nach 'go' auf 'to' pruefen
2980 mm=mm+1                                    912
2990 as=asc ( mid$ (d$(i),mm,1)+chr$           2165
     (0))
3000 if as=164 then 2310:rem to gefun          1576
     den
3010 if mm<len (d$(i)) then 2980               1372
3020 err$="GO without TO Error"                2715
3030 err=d(i)                                   758
3040 gosub 3580                                 651
3050 next i                                     276
3060 :
3070 rem ***** pass 4 *****
3080 :
3090 if fo$<>"j" and fo$<>"y" then             1781
     3300
3100 print "[home,13down]" tab(                2829
     23)"Pass 4 :"
3110 fo=0                                       559
3120 for i=1 to dd                              841
3130 :mm=.                                      640
3140 :f1=.                                      769
3150 :f2=.                                      773
3160 :d$(i)=left$ (sp$,fo)+d$(i)               2617
3170 :print "[home,13down]" tab(               1790
     31)d(i)
3180 :mm=mm+1                                  1117
3190 if mm>len (d$(i)) then next i:            1898
     goto 3300
3200 yy=asc ( mid$ (d$(i),mm,1)+chr$           2201
     (0))
3210 if yy=34 then f2=1-f2:rem falls           1772
3220 if yy=143 then f1=1:rem anfuehru          1805
     ngszeichen oder rem,
3230 if f1 or f2 then 3180:rem dann n          1726
     icht pruefen
3240 if yy=129 then fo=fo+2:rem for -          1923
     > schachtelungstiefe +1
3250 if yy=130 and fo>0 then fo=fo-2:          5338
     d$(i)=left$ (sp$,fo)+mid$
     (d$(i),fo+3)
3260 goto 3180                                  631
3270 :


3280 rem ***** save *****
3290 :
3300 print "[home,17down]" tab(16)"            2659
     [rvs] saving "
3310 open 1,ga,15                               874
3320 open 2,ga,1,dn$                           1112
3330 gosub 1090                                 635
3340 if d then close 2:close 1:gosub           2023
     3600:goto 3300
3350 ad=sa                                      577
3360 lb=255 and sa                             1273
3370 hb=sa/256                                 1109
3380 print#2,chr$ (lb) chr$ (hb);              1628
3390 for i=1 to dd                              841
3400 :ad=ad+len (d$(i))+5                      1687
3410 :lp=255 and ad                            1118
3420 :hp=ad/256                                 865
3430 :print#2,chr$ (lp) chr$ (hp);             1250
3440 :lz=d(i) and 255                          1423
3450 :hz=d(i)/256                              1421
3460 :print#2,chr$ (lz) chr$ (hz);             1415
3470 :print#2,d$(i) chr$ (0);                  1500
3480 next                                       130
3490 print#2,chr$ (0) chr$ (0);                1444
3500 close 2                                    260
3510 close 1                                    258
3520 print "[home,20down,2spaces]              2899
     Errors :"er%
3530 print                                      153
3540 end                                        128
3550 :


3560 rem ***** fehler *****
3570 :
3580 d$=err$                                    504
3590 d=err                                      373
3600 print "[home,21down,right]error        3407
      for "sn$" :"
3610 print "[down,shift-space]"d$;             630
3620 if d<>val (d$) then print "               1838
     [right]in";d;
3630 for t=1 to 5000                            960
3640 :get a$                                    163
3650 if a$="" then next                         962
3660 print "[home,21down,39spaces]"           2984
3670 print "[down,39spaces]";                 1789
3680 return                                     142
```

Listing von Recom 64