Wenn Sie den Lichtgriffel öfter in selbstgeschriebenen Programmen einsetzen wollen, sollten Sie sich mit dem Fototransistor einen echten Licht-»Griffel« basteln. Nehmen Sie dazu einen Kugel- oder Filzschreiber und entfernen Sie alle »Innereien«. Löten Sie an die beiden Anschlüsse des Fototransistors zwei ausreichend lange Kabel an und isolieren Sie sie sorgfältig gegeneinander. Ein Kurzschluß kann zwar keinen Schaden anrichten, doch setzt er den Fototransistor außer Betrieb. Stecken Sie den Fototransistor jetzt von hinten in den Stift und befestigen Sie ihn vor der Minenöffnung. Je kleiner diese ist, desto konstanter werden Ihre Werte anschließend sein, weil ein kleinerer Bildschirmausschnitt auf den Fototransistor einwirkt. Sie müssen jedoch aufpassen, daß nicht zu wenig Licht auf den Fototransistor fällt, da er sonst nicht mehr anspricht. In diesem Fall bohren Sie die Minenöffnung einfach etwas auf. Sie können auch mit dem Helligkeitsregler am Fernseher oder Monitor etwas korrigieren. Nach dem Anschluß an den Commputer besitzen Sie dann einen richtigen Lichtgriffel.

Damit ist die Besprechung der Control-Ports vorerst abgeschlossen.

(Tobias Nicol/ev)

```
100 REM ******************************    <238>
110 REM *                            *    <159>
120 REM *   T O N G E B E R   M I T  *    <002>
130 REM *   ----------------------   *    <237>
140 REM *                            *    <189>
150 REM *   L I C H T G R I F F E L  *    <072>
160 REM *   ----------------------   *    <011>
170 REM *                            *    <219>
180 REM *  BY   TOBIAS        NICOL  *    <229>
190 REM *                            *    <239>
200 REM *       NEUWIESENSTRASSE 20  *    <202>
210 REM *                            *    <003>
220 REM *       6000  FRANKFURT  71  *    <042>
230 REM *                            *    <023>
240 REM ******************************    <122>
250 :                                     <226>
260 REM ****** BILDSCHIRMAUFBAU  ******    <126>
270 POKE 53281,1 : POKE 53280,3           <029>
280 PRINT "{CLR,DOWN,2SPACE}BITTE WAEHLEN
    SIE!"                                 <087>
290 PRINT "{2SPACE}------------------{DOWN
    }"                                    <095>
300 FOR I = 1 TO 3                        <145>
310 PRINT "{2SPACE}↑↓5"                   <142>
320 PRINT "{2SPACE}B B TONGENERATOR";I     <092>
330 PRINT "{2SPACE}7↓7{2DOWN}"            <001>
340 NEXT I                                <170>
350 :                                     <072>
360 REM ***** SID-CHIP EINSTELLEN *****   <176>
370 T1 = 4 : SI = 54272                   <238>
380 FOR I = 1 TO 10                       <016>
390 READ A,B                              <186>
400 POKE SI+A,B                           <027>
410 NEXT I                                <240>
420 DATA 24,15, 0,207, 1,50, 7,207        <214>
430 DATA 8,100, 14,207, 15,150, 6,240     <138>
440 DATA 13,240, 20,240                   <173>
450 :                                     <172>
460 REM ** LIGHTPENPOSITION ABFRAGEN **   <095>
470 Y = PEEK (53268)                      <176>
480 :                                     <202>
490 REM ***** POSITION AUSWERTEN  *****   <183>
500 IF Y >  79 AND Y <  91 THEN T2 =  4   <094>
510 IF Y > 119 AND Y < 131 THEN T2 = 11   <041>
520 IF Y > 159 AND Y < 171 THEN T2 = 18   <174>
530 :                                     <254>
540 REM **** TONGENERATOR TAUSCHEN ****   <025>
550 IF T2 = T1 THEN 460                   <136>
560 POKE SI+T1,0                          <078>
570 POKE SI+T2,17                         <183>
580 :                                     <048>
590 REM ** NEUE WERTE & RUECKSPRUNG  **   <024>
600 T1=T2                                 <119>
610 GOTO 460                              <150>
620 :                                     <088>
630 :                                     <098>
640 :                                     <108>
650 REM ******************************    <024>
660 REM *** AENDERUNGEN FUER VC-20  ***    <233>
670 REM ******************************    <044>
680 REM                                   <234>
690 REM LOESCHEN SIE 370-440!             <209>
700 REM                                   <254>
710 REM GEBEN SIE EIN:                    <101>
720 REM                                   <018>
730 REM 270 POKE 36879,27                 <160>
740 REM 370 T2 = 36874 : POKE 36878,15    <153>
750 REM 470 Y = PEEK (36871)              <085>
760 REM 500 IFY< 63THENT2=36874:GOTO540   <101>
770 REM 510 IFY< 85THENT2=36875:GOTO540   <025>
780 REM 520 IFY<103THENT2=36876:GOTO540   <109>
790 REM 560 POKE T1,0                     <158>
800 REM 570 POKE T2,(T2-36871)*50         <060>
```

© 64'er

Listing 2. Tongeber mit Lichtgriffel. Bitte mit dem Checksummer eingeben.

Sortieren mit dem Computer (5)

Ein ausführlicher Geschwindigkeitsvergleich aller bisher erarbeiteten Sortieralgorithmen soll die Spreu vom Weizen trennen. Und ein weiteres Sortieralgorithmus wird vorgestellt.

Entgegen der anderslautenden Vorhersage im letzten Teil unseres Kurses, soll heute noch einmal auf Sortieralgorithmen in Basic eingegangen werden. Dieser Entschluß wurde aus mehreren Gründen gefaßt.

Erstens trafen nach Erscheinen der letzten Ausgabe mehrere interessante Beiträge zum Thema Sortieren in der Redaktion ein. Zweitens soll noch einmal ein grafischer Vergleich sämtlicher Routinen stattfinden, und drittens werden wir uns heute noch ein wenig mit den Vor- und Nachteilen der einzelnen Sortiermethoden beschäftigen.

Als erstes soll eine Verbesserung an den Mann gebracht werden. Heinrich Studer schickte der Redaktion einen Sortieralgorithmus, der dem schon bekannten Straight Selection stark ähnelt, wegen seines einfacheren Aufbaus jedoch weniger Speicherplatz benötigt. Der Einfachheit halber soll dieser Sortieralgorithmus den Namen Minisort bekommen.

Wie Sie aus dem Flußdiagramm in Bild 1 sehen können, besteht Minisort aus zwei geschachtelten FOR-NEXT-Schleifen. In der äußeren Schleife wird jeweils der Reihe nach ein Element genommen und mit sämtlichen Elementen der inneren Schleife verglichen. Wird während eines Vergleichs festgestellt, daß das andere Element aus dem Restfeld kleiner als unser Vergleichswert ist, so werden beide Variablen vertauscht.

Ist die innere Schleife also einmal durchlaufen worden, so befindet sich das kleinste Element des Gesamtfeldes nun an erster Stelle.

Jetzt wird in der äußeren Schleife zum nächsten Element übergegangen, wobei dieses wiederum mit dem Restfeld verglichen wird... und so weiter, bis man beim letzten Element angekommen ist. Es muß sich hierbei dann zwangsläufig um den größten Wert handeln.

Für Minisort (Listing 1) wie auch für unsere übrigen Sortieralgorithmen gilt, daß sie das Hauptprogramm für ihren Betrieb benötigen. Das Hauptprogramm dient der Erstellung der verschiedenen Arrays und der Anzeige der erstellten und sortierten Felder. In Listing 2 wurde auch diesmal wieder das Hauptprogramm abgedruckt, da es um ein paar Sachen erweitert wurde. Es ist jetzt mit der Option »von Hand erstellen« zusätzlich möglich, jeweils vorwärts und rückwärts sortierte Felder zu erstellen, um die »Testkandidaten« (sprich: Sortierprogramme) auf Herz und Nieren zu testen.

Um Ihnen jedoch einen großen Teil der Arbeit abzunehmen, habe ich für diese Folge unseres Kurses noch einmal sämtliche Sortierprogramme, die bisher besprochen wurden, in den drei Bereichen »Feld un-

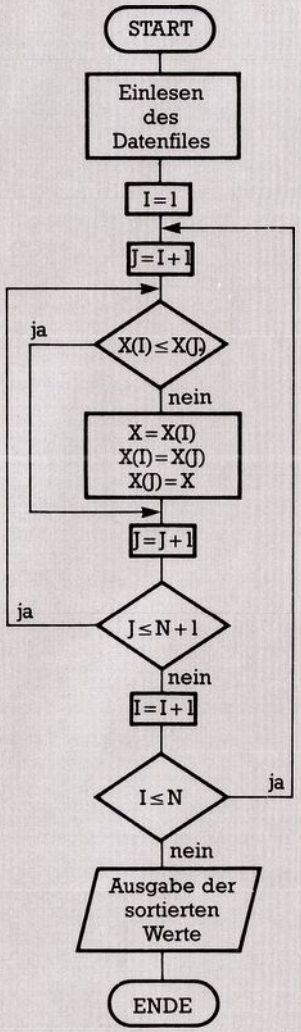

Bild 1. Programmablaufplan von »Minisort«, einer der einfachsten Sortierverfahren

sortiert«, »Feld vorwärts sortiert« und »Feld rückwärts sortiert« untersucht. Wir werden nachher noch ausführlich auf die Ergebnisse dieses »Monstertests« eingehen.

Jetzt jedoch noch zu einem anderen Sortierprogramm, das uns Horst Armin Kosog eingesandt hat. Es trägt den Namen »Mischsort« und wurde von besagtem Absender anhand einer Aufgabenstellung aus einem Informatikbuch (H. Balzert, Informatik 1, Hueber-Holzmann Verlag, München 1976, Seite 210) erstellt und selbständig weiterentwickelt.

Das Prinzip dieses Sortieralgorithmus ist folgendes:

Ähnlich wie bei Quicksort wird das zu sortierende Array bei Mischsort in Untereinheiten (Teilfelder) untergliedert. Diese Teilfelder fangen bei der Größe 2 an und nehmen dann im Quadrat an Umfang zu (4,8,16,...).

Zuerst werden die Untereinheiten der Größe 2 jeweils für sich sortiert, so daß wir A/2 (A ist die Anzahl der Elemente im Gesamtfeld), sortierte Teilfelder erhalten.

Jeweils zwei benachbarte Teilfelder werden nun »zusammengemischt« und wiederum sortiert.

Setzt man dies durch das gesamte Array fort, so erhalten wir nun A/4 sortierte Teilfelder der Länge 4.

Als nächstes werden jeweils zwei benachbarte Teilfelder der Länge 4 zusammengemischt und sortiert... und so weiter, bis schließlich nur noch ein (Teil) Feld übrigbleibt, nämlich das Gesamtfeld.

```
10040 TI$="000000"                                    <113>
10050 FOR X=1 TO A-1                                  <127>
10060 FOR Y=X+1 TO A                                  <008>
10070 IF A$(X)<=A$(Y)THEN 10090                       <242>
10080 S$=A$(X):A$(X)=A$(Y):A$(Y)=S$                   <198>
10090 NEXT Y,X                                        <089>
```

64'er

Listing 1. Ein kleines und effektives Sortierprogramm: Minisort

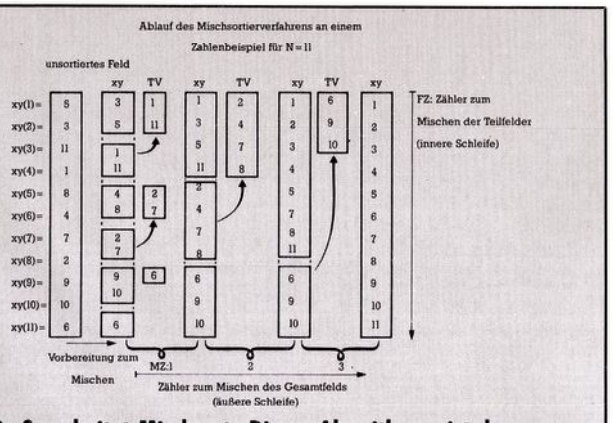

Bild 2. So arbeitet Mischsort. Dieser Algorithmus ist der schnellste unseres Tests.

```
10 REM ERSTELLEN EINES FELDES ZUM                     <007>
20 REM SORTIEREN.                                     <053>
30 REM DAS ERSTELLEN KANN ZUFAELLIG                   <198>
40 REM ODER GEZIEHLT (DURCH EINGABE)                  <005>
50 REM ERFOLGEN.                                      <251>
60 REM                                                <122>
70 REM SORTIERALGORITHMEN ERHALTEN DIE                <053>
80 REM ZEILENNUMMERN VON 1000 BIS 10000               <167>
90 REM SIE BENOETIGEN JEWEILS DIESEN                  <211>
99 REM VORSPANN ZUR AUSFUEHRUNG.                      <089>
100 REM HERSTELLUNG EINES ARRAYS:                     <090>
110 REM ARRAYVARIABLE       - A$                      <220>
120 REM SCHLEIFENVARIABLEN - X, Y, Z                  <082>
130 REM HILFSVARIABLEN      - B$, C$, D$              <124>
140 REM DREIECKTAUSCH MIT   - S$                      <230>
150 PRINT"{CLR}":CLR                                  <102>
160 PRINT"SOLL VON{SPACE,RVSON}H{RVOFF}AND
    ODER{SPACE,RVSON}Z{RVOFF}UFAELLIG ERS
    TELLT":PRINT                                      <027>
170 INPUT"WERDEN ";X$                                 <239>
180 IF X$<>"H"AND X$<>"Z"THEN 150                     <056>
190 IF X$="H"THEN GOSUB 220:GOSUB 1000:GOT
    O 210                                             <161>
200 GOSUB 220:GOSUB 2000                              <223>
210 GOTO 4000: REM WEITERMACHEN                       <132>
220 REM ANZAHL DER ELEMENTE BESTIMMEN                 <098>
230 PRINT:INPUT"ANZAHL DER ELEMENTE ";A               <115>
240 IF A>10000 THEN PRINT:PRINT"ZU VIELE E
    LEMENTE":GOTO 230                                 <119>
250 IF A<10 THEN PRINT:PRINT"ZU WENIGE ELE
    MENTE":GOTO 230                                   <138>
255 DIM A$(A)                                         <078>
260 INPUT"{2DOWN,RVSON}D{RVOFF}RUCKER ODER
    {SPACE,RVSON}B{RVOFF}ILDSCHIRM ";Y$               <207>
270 IF Y$<>"D"AND Y$<>"B"THEN 260                     <160>
280 IF Y$="D"THEN D=4:GOTO 300                        <128>
290 D=3                                               <253>
300 RETURN                                            <104>
1000 REM EINGABE VON HAND                             <107>
1010 PRINT"{CLR}V ODER R";                            <093>
1020 INPUT X$:IF X$<>"R"AND X$<>"V"THEN 10
     20                                               <051>
1030 R=1:X1=1:X2=A                                    <123>
1040 IF X$="R"THEN R=-1:X1=A:X2=1                     <181>
1050 Z1=65:Z2=65:Z3=65:FOR X=X1 TO X2 STEP
     R                                                <127>
1060 A$(X)="":A$(X)=CHR$(Z1):Z1=Z1+1:IF Z1
     >90 THEN Z1=65                                   <126>
1062 A$(X)=CHR$(Z2)+A$(X):IF Z1=90 THEN Z2
     =Z2+1:IF Z2>90 THEN Z2=65                        <254>
1064 A$(X)=CHR$(Z3)+A$(X):IF Z2=90 THEN Z3
     =Z3+1:IF Z3>90 THEN Z3=65                        <000>
1070 NEXT X                                           <002>
1080 RETURN                                           <122>
2000 REM ZUFAELLIGE EINGABE                           <177>
2010 PRINT"{CLR}"                                     <220>
2020 PRINT:PRINT"ES WERDEN JETZT"A" ELEMEN
     TE ZUFAELLIG":PRINT:PRINT"AUSGEWAEHLT
     "                                                <212>
2030 PRINT:PRINT"JEDES ELEMENT BESTEHT AUS
     3 ZEICHEN.":PRINT:PRINT                          <198>
2040 FOR X=1 TO A                                     <232>
2050 A$(X)=""                                         <206>
2060 FOR Y=1 TO 3:A$(X)=A$(X)+CHR$(INT(RND
     (TI)*25)+65):NEXT Y                              <134>
2070 NEXT X                                           <242>
2080 RETURN                                           <037>
3000 REM ZWISCHENAUSGABE DER ELEMENTE                 <197>
3010 FOR I=1 TO A-9 STEP 10                           <145>
3020 FOR J=I TO I+9:PRINT#1,A$(J)" ";:NEXT
     J                                                <048>
3030 PRINT#1:NEXT I: PRINT#1                          <051>
3040 RETURN                                           <048>
4000 REM WEITERMACHEN                                 <180>
4005 OPEN 1,D                                         <020>
4010 PRINT"{CLR}AUSGABE DES ERSTELLTEN FEL
     DES"                                             <149>
4020 PRINT                                            <056>
4030 GOSUB 3000                                       <192>
4040 REM SORTIERUNG STARTET                           <169>
4050 REM                                              <046>
```

Listing 2. Modifiziertes Hauptprogramm für alle Sortieralgorithmen
(Fortsetzung auf der nächsten Seite)

Diese Arbeitsweise sei noch einmal in Bild 2 grafisch dargestellt.

Damit Sie beim Abtippen von Listings nicht aus der Übung kommen, haben wir auch zu Mischsort ein Programm (Listing 3). Es muß, wie alle übrigen Algorithmen auch, in unser Hauptprogramm eingefügt werden.

Ein dokumentiertes Flußdiagramm zu Mischsort ist in Bild 3 abgedruckt.

Schneller als Quicksort!

Mit dieser Überschrift stellte kürzlich eine Zeitschrift für Personal Computer (Computer persönlich, Ausgabe 14/85) ein neues Sortierverfahren vor. Das Programm lief ursprünglich auf einem CBM 8032 und machte also keine Mühe bei der Umstellung auf den Commodore 64. Ein derartiger Beitrag darf natürlich in unserem Kurs nicht unerwähnt und unbesprochen bleiben, weshalb ich das Programm in unsere Reihe der Sortiermethoden aufgenommen habe.

Das interessante an Supersort (so heißt das Wunderprogramm) ist sicherlich die Art und Weise, wie bei diesem Programm Zeit gespart wird. Wir haben es hier nämlich eigentlich mit zwei Sortierprogrammen zu tun.

Das eine Sortierprogramm davon kennen wir bereits. Es handelt sich um Bubblesort 2, also die verbesserte Version des ursprünglichen Bubblesort. (Wir erinnern uns: die Verbesserung von Bubblesort bestand im Einführen eines Zeigers, der dafür sorgte, daß Bubblesort mit der Arbeit aufhört, sobald keine Vertauschungen mehr stattfinden. Dieser »Kunstgriff« sorgte dafür, daß Bubblesort 2 bei Feldern, die insgesamt schon sortiert sind, einige wenige Elemente aber neu eingeordnet werden müssen, äußerst schnell wird und die kürzesten Sortierzeiten überhaupt erreichen kann.).

Bei Supersort nutzt man nun die Tatsache, daß Bubblesort bei Feldern bis maximal zehn Elementen sehr schnell arbeitet. Es wird also das Gesamtfeld einfach in Teilfelder mit maximal zehn Elementen aufgeteilt, die dann alle einzeln von Bubblesort nachsortiert werden.

Damit die Teilfelder dann auch in der richtigen Reihenfolge stehen, wird das Gesamtfeld vorsortiert, wobei die Teilfelder entsprechend der Anfangsbuchstaben der Feldelemente zusammengestellt werden.

Das Flußdiagramm von Supersort sehen Sie in Bild 4, wobei auf die detaillierte Darstellung von Bubblesort verzichtet wurde.

Auf unser Hauptprogramm zugeschnitten können Sie Supersort in Listing 4 finden. Es kann ohne weitere Änderungen mit den anderen Routinen dieses Kurses zusammenarbeiten.

Nachdem wir nun eine ziemliche Menge von verschiedenen Sortieralgorithmen kennen, sollen diese jetzt auch einmal im Vergleich betrachtet werden. Für den Anwender stellt sich in der Regel nur die Frage, wel-

```
50000 T$=TI$:REM ENDEBEHANDLUNG                <041>
50010 PRINT#1                                  <204>
50020 GOSUB 3000                               <207>
50030 PRINT#1,A;" ELEMENTE"                    <251>
50040 PRINT#1:PRINT#1:PRINT#1,T$:CLOSE 1       <158>
50050 END                                      <011>
```

© 64'er

Listing 2. (Schluß)

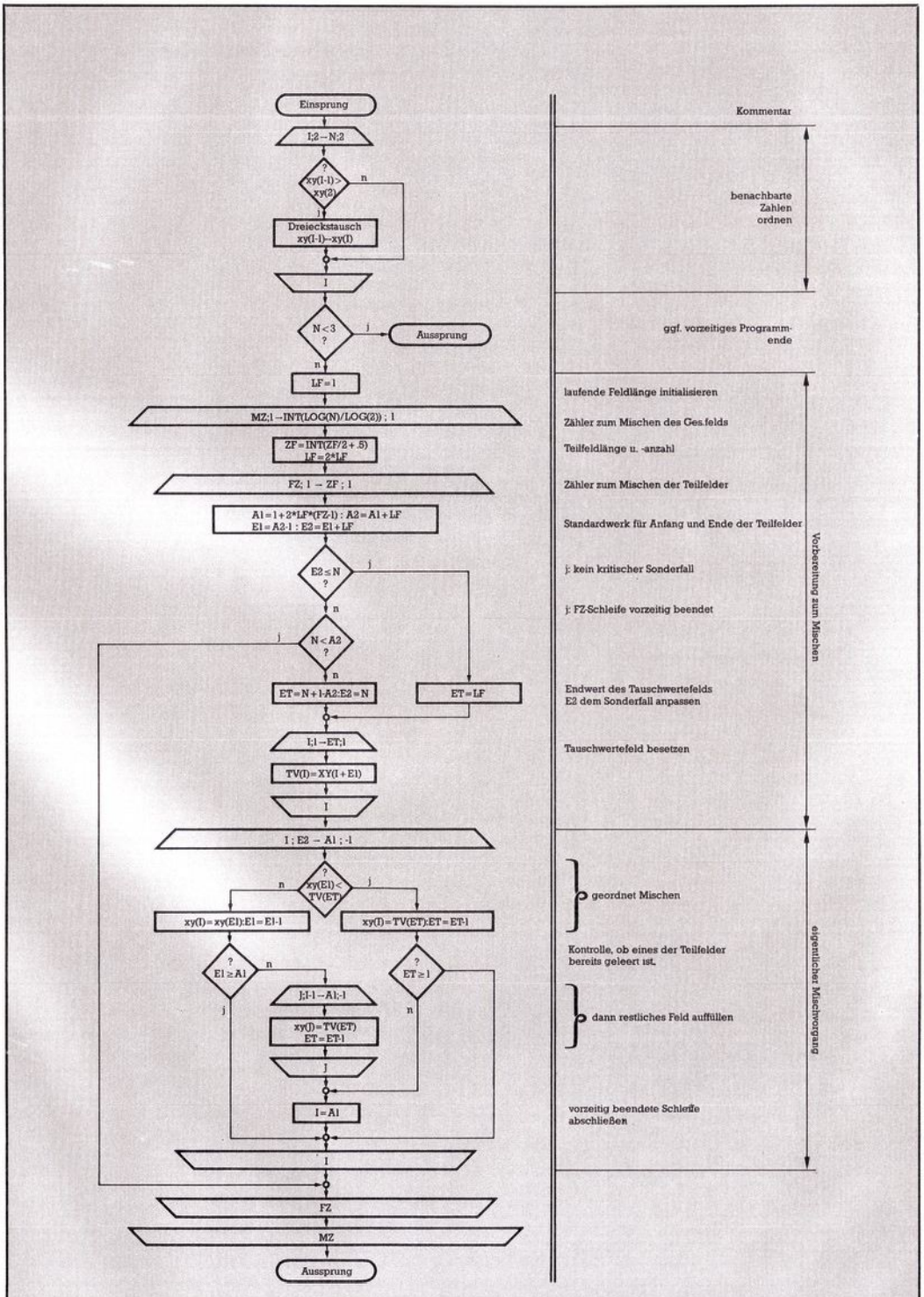

Bild 3. Flußdiagramm von Mischsort. Deutlich ist die Aufteilung in zwei Arbeitsgänge zu erkennen.

```
10040 TI$="000000"                                    <113>
10050 ZF=INT(A/2+.5):DIM TV$(ZF)                      <179>
10060 FOR X=2 TO A STEP 2                             <169>
10070 IF A$(X-1)>A$(X)THEN TV$(0)=A$(X-1):
      A$(X-1)=A$(X):A$(X)=TV$(0)                      <203>
10080 NEXT X:IF A<3 THEN 50000                        <163>
10090 LF=1:FOR MZ=1 TO INT(LOG(A-1)/LOG(2)
      )                                               <009>
10100 ZF=INT(ZF/2+.5):LF=2*LF                         <057>
10110 FOR FZ=1 TO ZF:A1=1+2*LF*(FZ-1):A2=A
      1+LF:E1=A2-1:E2=E1+LF                           <030>
10120 IF E2<=A THEN ET=LF:GOTO 10150                  <009>
10130 IF A<A2 THEN 10230                              <207>
10140 E2=A:ET=A+1-A2                                  <023>
10150 FOR X=1 TO ET:TV$(X)=A$(X+E1):NEXT              <003>
10160 FOR X=E2 TO A1 STEP-1                           <048>
10170 IF A$(E1)<TV$(ET)THEN 10200                     <183>
10180 A$(X)=A$(E1):E1=E1-1:IF E1>=A1 THEN
      10220                                           <036>
10190 FOR Y=X-1 TO A1 STEP-1:A$(Y)=TV$(ET)
      :ET=ET-1:NEXT Y:GOTO 10210                      <075>
10200 A$(X)=TV$(ET):ET=ET-1:IF ET>=1 THEN
      10220                                           <056>
10210 X=A1                                            <096>
10220 NEXT X                                          <006>
10230 NEXT FZ,MZ                                      <042>
```

64'er

Listing 3. Das schnellste Sortierprogramm: Mischsort

```
10000 TI$="000000":DIM ZP$(A),AN$(26),X(26
      ),B$(A)                                         <182>
10040 FOR II=1 TO 26                                  <252>
10050 AN$(II)="":X(II)=0                              <173>
10060 NEXT II                                         <130>
10070 IF ASC(A$(1))>64 THEN MIN=64                    <158>
10080 IF ASC(A$(1))>192 THEN MIN=192                  <153>
10085 FOR II=1 TO A                                   <204>
10090 JJ=ASC(A$(II))-MIN                              <239>
10100 IF JJ<0 THEN JJ=26                              <215>
10110 AN$(JJ)=AN$(JJ)+STR$(II+100)                    <073>
10120 NEXT II                                         <190>
10130 L=1                                             <185>
10140 FOR JJ=1 TO 26                                  <120>
10150 X(JJ)=LEN(AN$(JJ))/4                            <128>
10160 IF AN$(JJ)=""THEN 10210                         <018>
10170 FOR II=1 TO LEN(AN$(JJ))STEP 4                  <042>
10180 X=(VAL(MID$(AN$(JJ),II,4))-100)                 <011>
10190 B$(L)=A$(X):A$(X)="":L=L+1                      <194>
10200 NEXT II                                         <014>
10210 NEXT JJ                                         <048>
10220 FOR II=1 TO A                                   <083>
10230 A$(II)=B$(II):B$(II)=""                         <050>
10240 NEXT II                                         <056>
10250 Y=0                                             <087>
10260 FOR L=1 TO 26                                   <149>
10270 X=Y+1:Y=X+X(L)-1                                <105>
10280 IF X(L)=0 OR X(L)=1 THEN 10340                  <033>
10290 FOR JJ=Y-1 TO X STEP-1:FL=-1                    <160>
10300 FOR N=X TO JJ                                   <210>
10310 IF A$(N)>A$(N+1)THEN FL=0:TE$=A$(N):
      A$(N)=A$(N+1):A$(N+1)=TE$                       <133>
10320 NEXT N                                          <028>
10330 IF NOT FL THEN NEXT JJ                          <018>
10340 NEXT L                                          <032>
```

64'er

Listing 4. Das Programm Supersort für den C 64

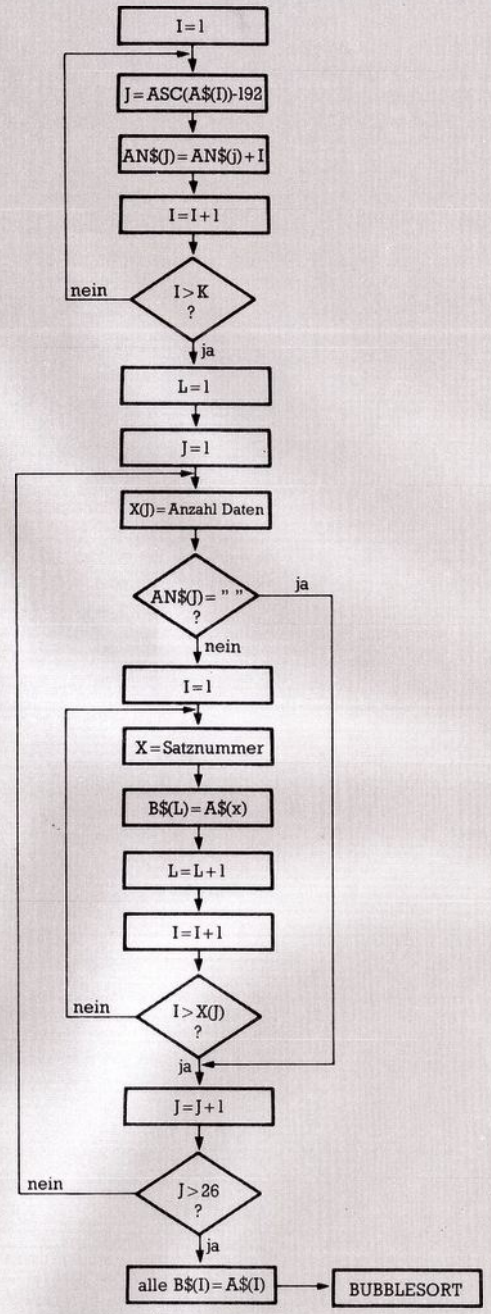

Bild 4. Flußdiagramm von Supersort

ches Sortierprogramm er für seine Problemlösung am besten verwenden kann, und diese Frage ist mitunter nicht so leicht zu beantworten.

Für den Vergleichstest habe ich folgende Sortierprogramme verwendet:

Straight Insertion, Straight Selection, Bubblesort 2, Shellsort, Heapsort, Quicksort, Supersort und Mischsort.

Fangen wir mit den Bedingungen des Tests an. Die einzelnen Programme wurden mit vorsortierten und unsortierten Feldern getestet, wobei nur Stringarrays verwendet wurden. Während des Tests wurde die Anzahl der Elemente jedesmal vergrößert, damit sich anschließend ein Kurvendiagramm darstellen ließ.

Die ersten fünf Algorithmen wurden mit 10, 20, 50, 100 und 200 Elementen gemessen (Bubblesort nur bis 100). Die drei schnelleren Methoden Quicksort, Supersort und Mischsort wurden auch noch mit 500 Elementen geprüft.

Höhere Anzahlen von Feldelementen waren aus mehreren Gründen nicht nötig.

Erstens steigt die Sortierzeit einiger Routinen nach kurzer Zeit enorm an, was zu Werten außerhalb der Grafikskala führt.

Zweitens kommt es zu Konflikten mit der Geduld des Testers, wenn zu viele Elemente verwendet werden (Bubblesort 2 benötigte beispielsweise bei 1000 rückwärts geordneten Elementen sage und schreibe 6702 Sekunden; das sind 1 Stunde 51 Minuten und 42 Sekunden!). Und drittens kommt ab einer gewissen Zahl an Feldelementen noch ein ganz anderes Problem zum Tragen, das sich Garbage Col- ►

lection nennt und auch den schnellsten Sortieralgorithmus zur Verzweiflung bringt.

Nun also zur Besprechung der Testergebnisse:

Sehen Sie sich einmal Bild 5 an, es enthält die Testergebnisse von Bubblesort einmal grafisch dargestellt.

Auf der X-Achse ist dabei die Anzahl der Elemente und auf der Y-Achse die benötigte Sortierzeit in Sekunden angeführt.

Wie man erkennen kann, steigt die Sortierzeit dabei exponentiell an, was auf lange Wartezeiten hoffen läßt. Eine Ausnahme bilden jedoch die schon sortierten Felder. Hier zeigt sich Bubblesort von seiner besten Seite. Es stellt innerhalb von wenigen Sekunden fest, ob ein Feld schon sortiert ist und eignet sich deshalb hervorragend zum Einordnen weniger Elemente in ein großes sortiertes Feld.

Bild 6 zeigt Straight Insertion in Aktion. Dieses Sortierprogramm zeigt ein ähnliches Verhalten wie Bubblesort und ist sogar noch ein wenig schneller. Die Anwendungsgebiete liegen also auf der gleichen Ebene wie bei Bubblesort: Einordnen weniger Elemente in ein großes vorsortiertes Feld (vielleicht ließe sich durch Straight Insertion Supersort noch ein wenig mehr »aufmöbeln« ?!?).

Bild 7 macht uns mit der Charakteristik von Straight Selection vertraut, und man kann nicht verhehlen, daß dieser Sortieralgorithmus auf allen Gebieten langsam und uneffektiv arbeitet. Er lohnt sich der Einfachheit halber bei sehr kleinen Feldern mit unterschiedlichen Sortierungen.

Bild 8 zeigt uns den ersten effektiven Algorithmus, der durchaus schon praktisch eingesetzt werden kann. Es handelt sich um Shellsort, und wir können unschwer erkennen, daß diese Kurve im Gegensatz zu der von Straight Selection schon ziemlich flach verläuft, was einen langsameren Anstieg der Sortierzeiten bei steigender Anzahl von Elementen zur Folge hat.

Die Kurve in Bild 9 hat einen ganz ähnlichen Verlauf zur Kurve in Bild 8, nur ist die Steigung wieder um ein Stück flacher geworden. Heapsort läßt also durchaus auch praktische Anwendungen zu, es wird jedoch wegen seiner komplizierten Struktur kaum eingesetzt.

Bild 10 zeigt den Zeitverlauf bei Quicksort. Das Erstaunliche an den höheren Sortieralgorithmen ist, daß sich alle Sortierzeiten, sowohl die von völlig unsortierten Feldern als auch die von sortierten Feldern ständig aneinander annähern, was auf einen sehr breit gefächerten Anwendungskreis dieser Sortiermethoden schließen läßt.

① – – – – – – –
= Daten abwärts steigend vorsortiert

② ————————
= Daten zufällig sortiert

③ ·····
= Daten aufwärts steigend vorsortiert

= keine weitere Angabe möglich, da Programm die Arbeit einstellt

Eine Sache sollte im Zusammenhang von Quicksort nicht unerwähnt bleiben: Supersort bedient sich Bubblesort zur Endsortierung seiner Teilfelder. Diese Methode ist auch bei Quicksort üblich, wenngleich wir das in unserem Kurs nicht berücksichtigt haben. Es zeigt sich nämlich in der Praxis, daß die »großen« Sortiermethoden bei kleiner werdenden Teilfeldern mitunter sehr schwerfällig arbeiten, weshalb es besser ist, diese Teilfelder (zirka 10 Elemente) an kleine (und bei diesen Größen auch schnelle) Sortieralgorithmen zu übergeben.

Bild 11 macht uns mit der Geschwindigkeit von Supersort vertraut und bringt schon die erste Überraschung:

Bei unsortierten Feldern bis zu 400 Elementen und bei aufwärts sortierten ist der Sortieralgorithmus schneller als Quicksort, bei abwärts sortierten langsamer. Außerdem zeigte sich Supersort im Test wenig entgegenkommend, was seine Zuverlässigkeit betrifft.

Erstens baut Supersort seine Teilfelder mit Hilfe einer Stringvariable auf, die bekanntermaßen nur 255 Zeichen lang werden kann. Das hat den Haken, daß Supersort mit einer »?STRING TOO LONG«-Meldung ausstieg, sobald eine ganze Reihe von gleichartigen Elementen verarbeitet werden mußten. Das ist jedoch bei einem vorsortierten Feld leider oft der Fall. Aus diesem Grund mußte ich bei einer Zahl ab 100 sortierten Elementen mit dem Test aufhören.

Zweitens fiel bei Supersort unangenehm auf, daß diese Sortierroutine nicht in der Lage ist, Strings zu sortieren, die Zahlen enthalten, was auf die Beschränkung auf 26 Zeichen des Alphabets zurückzuführen ist.

Hätte Supersort, abgesehen von den obigen Schwächen, einwandfrei gearbeitet, so hätte man sicher auf interessante Meßwerte kommen können. Es zeigte sich jedoch schon bei den zufallssortierten Elementen, daß Supersort auf ganz andere Schwierigkeiten stößt, wenn die Zahl der Elemente über 500 hinaus geht. Auch hier wieder das berüchtigte Wort Garbage Col-

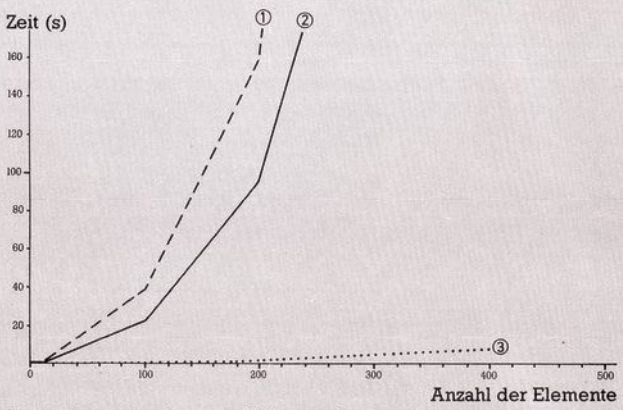

Bild 5. Zeitverlauf von Bubblesort 2. Die Sortierzeit steigt steil exponentiell an.

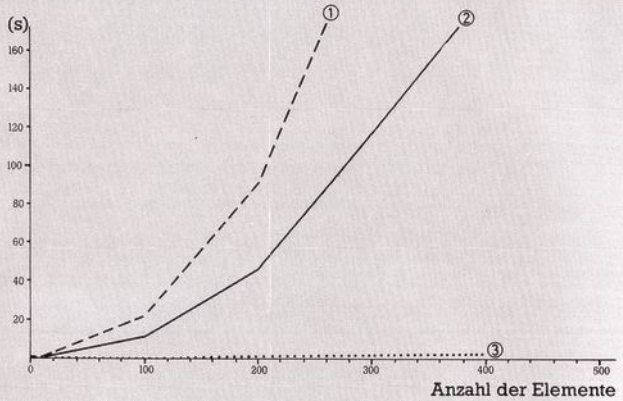

Bild 6. Straight Insertion zeigt vor allen bei vorsortierten Feldern seine Stärke

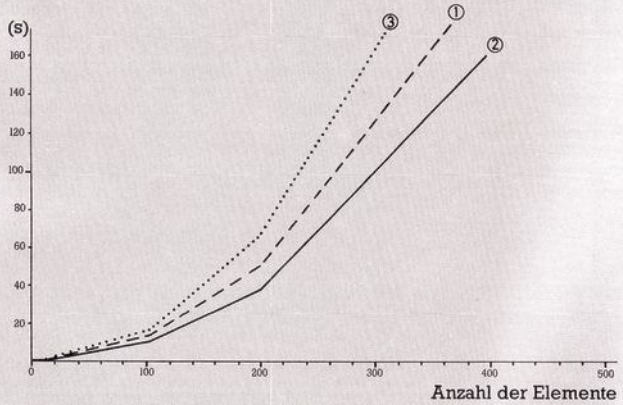

Bild 7. Straight Selection benötigt große Sortierzeiten und findet kaum praktischen Einsatz

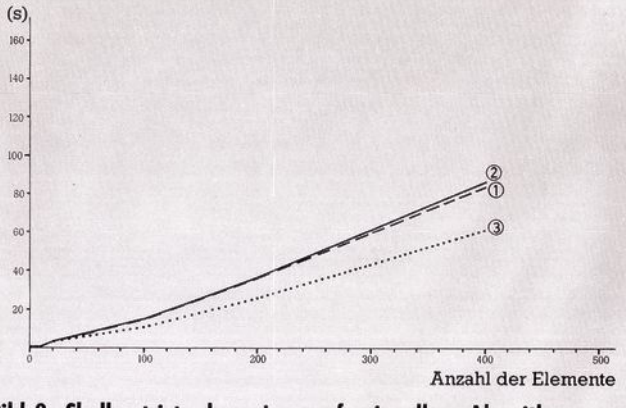

Bild 8. Shellsort ist schon ein »professioneller« Algorithmus

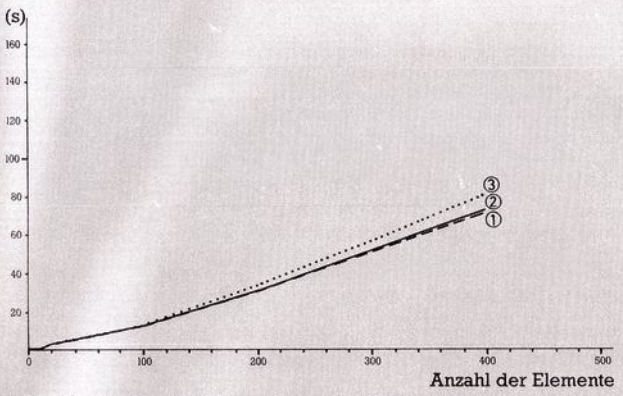

Bild 9. Heapsort zählt zu den schnellsten Sortiermethoden, wie die flache Kurve zeigt

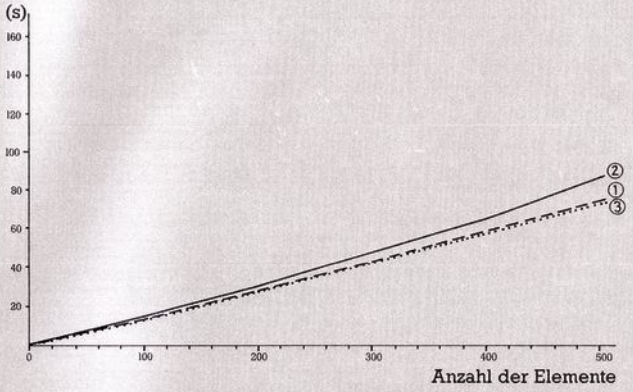

Bild 10. Quicksort war lange der schnellste Sortieralgorithmus

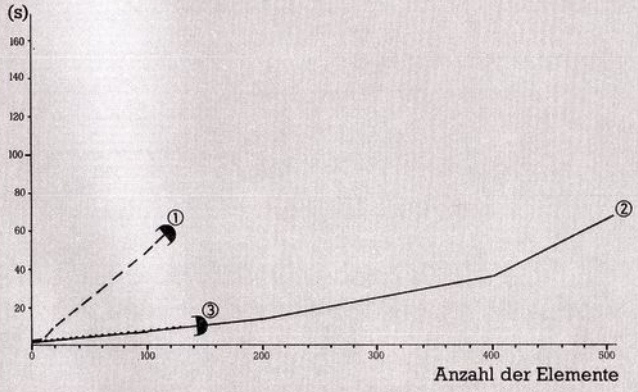

Bild 11. Supersort zeigt einige Mängel auf dem C 64, kann aber spezialisiert hervorragend eingesetzt werden

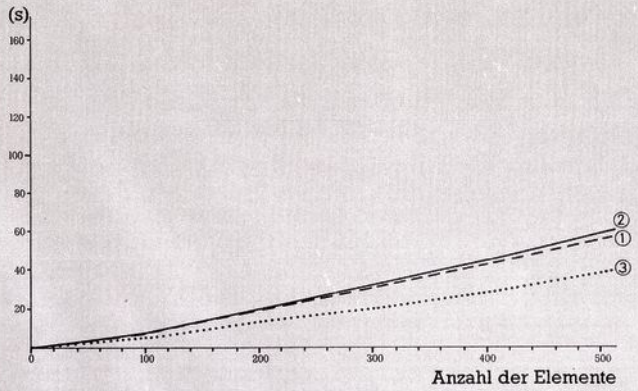

Bild 12. Mischsort besticht durch seine Schnelligkeit in allen Bereichen

lection. Supersort benötigt nämlich mehr Zwischenspeicher als Quicksort und wird somit von diesem überholt, sobald bei Supersort die Garbage Collection in Aktion tritt (und das tut sie natürlich früher als bei Quicksort).

Natürlich kann man jetzt den Entwickler dieses Programms nicht des Lügens bezichtigen, wenn er behauptet: »Supersort ist schneller als Quicksort!«. Supersort ist nämlich ursprünglich auf einem CBM 8032 geschrieben worden, und dieser verfügt über eine andere Stringverwaltung als der C 64, was eine erheblich schnellere Garbage Collection zur Folge hat

Für den C 64 gilt jedoch (leider): Supersort ist nur bedingt brauchbar, da es einige Schwächen aufweist, die Quicksort nicht hat.

Nun zu Bild 12. Es zeigt den letzten der Sortieralgorithmen, nämlich Mischsort. Und hier erlebte ich die große Überraschung: Auch Mischsort ist schneller als die Normalversion von Quicksort (ohne Tricks mit kleineren Sortierroutinen). Mischsort zeigte im Test wahrhaftig traumhafte Zeiten für Basic-Sortierprogramme, die teilweise nur 50 Prozent der Zeiten von Quicksort betrugen. Es wurde auch deutlich, daß bei Mischsort die Garbage Collection noch später einsetzte als bei Quicksort (das Einsetzen der Garbage Collection ist an den »Knickstellen« der Graphen zu erkennen), was die guten Zeiten aber nur teilweise rechtfertigte.

Mischsort ist genauso einsatzfähig wie Quicksort (ohne »Macken«) und kann deshalb als bestes Sortierprogramm dieses Tests betrachtet werden. Haben Sie viele große Felder zu sortieren, so spielt es bei diesem Algorithmus fast keine Rolle, ob die Felder vorsortiert sind oder nicht; schnell ist er auf jeden Fall.

Noch eine Randbemerkung zum Schluß: Vielleicht haben Sie auch schon Methoden entwickelt, wie sie die Garbage Collection »in den Griff bekommen«. Haben Sie vielleicht als »Maschinensprachefreak« ein neues Basic-ROM geschrieben (mit besserer Stringverwaltung) oder haben Sie anhand unseres Vorkurses über Strings die Lösung des Problems gefunden?

Die 64'er-Redaktion würde sich über Beiträge aus dem Leserkreis freuen. Vielleicht möchten Sie dazu beitragen, daß auch andere Anwender in den Genuß entsprechender Verbesserungen kommen?

Schreiben Sie uns doch! Bestimmt können wir dann den einen oder anderen Trick mit in unsere Reihe aufnehmen.

Bis zum nächsten Mal.

(K.Schramm/gk)

```
6067    LDA #$EA
6069    STA 0315
606C    LDA #$0E
606E    STA D020
6071    CLI
6072    RTS
```

Unser Programm ist komplett. Speichern Sie es bitte vor dem Starten ab. Nach dem SYS 24576 finden Sie einen hübschen bunten Rahmen vor, oberhalb und unterhalb des Textfensters ist er schwarz. Besonders gut — finde ich — sieht das Ganze aus, wenn man die Hintergrundfarbe des Textfensters auch auf Schwarz setzt. Das Programm erlaubt noch einige Experimente:

Durch POKE-Kommandos in die Speicherstelle 2 kann die aktuelle Streifenbreite variiert werden, durch POKEs in die Zelle 24645 der Startwert der Verzögerungsschleife. Probieren Sie's doch mal aus. Eine Erkenntnis werden Sie gewinnen: In der Unterbrechungs-Programmierung spielt die Zeit eine wichtige Rolle. Das zeigt sich auch, wenn man zum Beispiel Cursorbewegungen durchführt: Die Streifen fangen an zu wandern.

Weitere Möglichkeiten zum Experimentieren sind gegeben, wenn Sie die Rasterzeilen verändern, die den oberen und unteren Rand des Textfensters markieren:

Durch POKE 24661,Zahl verschieben Sie die obere, durch POKE 24635,X:POKE 24588,X die untere Rasterzeile, von der an alles schwarz ist. Wie schon vorhin erwähnt, habe ich im Programm diese Werte auf 50 beziehungsweise 248 fixiert, weil genau dort auf meinem Monitor das Textfenster liegt.

Mit diesem Beispiel und dem aus der Grafikserie sollte es Ihnen nun möglich sein, auch andere Unterbrechungsprogramme zu schreiben, die sich der Rasterzeilen-Unterbrechung per VIC-II-Chip bedienen. Eine Bemerkung sollte ich Ihnen noch auf den Weg Ihrer eigenen Versuche mitgeben: Der Elektronenstrahl, der über den Bildschirm saust und beim Erreichen des von uns bestimmten Rasterzeilenwertes zum Auslösen des IRQ führt, ist enorm schnell. Die Serviceprogramme dürfen deshalb nicht zu lang sein, sonst steht der nächste IRQ schon wieder an, bevor der vorangegangene bearbeitet ist.

In der kommenden Folge sollen Beispiele für andere Unterbrechungsformen vorgestellt werden, die durch die CIAs und die RESTORE-Taste angesprochen werden. Danach soll es um die Anwendung des bisher Gelernten gehen, wobei wir uns wieder mehr den Interpreter-Routinen und auch den Kernal-Möglichkeiten zuwenden wollen.

(Heimo Ponnath/gk)