Effektives Programmieren (4)

Daten sortieren mit dem Computer

Wie sortiere ich meine Daten am besten? Dieses Problem ist so alt wie der Computer, und entsprechend vielfältig sind die Lösungsvorschläge. Wir stellen Ihnen in mehreren Folgen die wichtigsten und bekanntesten Sortiermethoden und dazu jeweils ein entsprechendes Listing vor.

Wie oft haben Sie wohl schon ein Telefonbuch oder ein Adressenverzeichnis aufgeschlagen, um nach einem bestimmten Namen zu suchen? Eine Frage, die wohl kaum zu beantworten ist.

Es passiert alle Augenblicke, daß man etwa ein Lexikon zur Hand nimmt, um ein Fremdwort nachzuschlagen oder daß man den Fahrplan einer Buslinie nach der nächsten Abfahrtszeit durchsucht.

Stellen Sie sich jetzt einmal vor, Sie hätten ein Lexikon in der Hand, das nicht, wie üblich, in alphabetisch sortierter Form vorliegt, sondern alle Stichwörter völlig durcheinander enthält.

Sie werden wohl zugeben, daß sich die Suche nach einem bestimmten Wort nun als ziemlich hoffnungslos herausstellen wird.

Mit dieser Feststellung sind wir aber schon beim Thema.

Heutzutage wird die Verwaltung und Weiterverarbeitung großer Datenmengen fast ausschließlich von Computern vorgenommen. Alle Stichwörter eines Lexikons sind zum Beispiel in Großrechenanlagen gespeichert und werden vollelektronisch in den Satz gegeben.

Nun wird Datenverarbeitung aber nicht nur auf Großrechenanlagen durchgeführt, sondern auch durchaus auf Mikrocomputern; sei es als Kundendatei oder als elektronisches Notizbuch.

Die Notwendigkeit einer Ordnung in diesen Datenbeständen wurde schon zur Sprache gebracht. Uns soll nun in dieser Folge interessieren, was es für Methoden des Ordnens von Daten gibt.

Wir werden uns also im Laufe dieser Reihe mit den verschiedensten Sortieralgorithmen beschäftigen; angefangen beim Sortieren durch direktes Einfügen (straight insertion) bis hin zum schnellsten Algorithmus, der zur Zeit existiert, dem Sortieren durch Zerlegen (Quicksort).

Jede Sortiermethode soll dabei bis ins Detail erklärt werden, und Sie werden sehr schnell erkennen, daß Sortieren nicht gleich Sortieren ist.

Haben wir ein großes Variablenfeld angelegt, so gibt es generell zwei verschiedene Methoden beim Suchen eines bestimmten Elements:

1) Durchsuchen sämtlicher Elemente
2) binäre Suche

Die erste Methode ist uns klar. Hierbei werden einfach alle Elemente des Feldes vom Anfang bis zum Ende durchgekämmt, um das gewünschte herauszufinden. Bei der Geschwindigkeit, mit der ein Computer seine Variablenfelder durchgehen kann, müssen schon gewichtige Gründe für das Sortieren sprechen.

Diese Gründe gibt es auch. Einer davon ist das Verwalten großer Datenmengen.

Sortieren von Feldern

Auch ein »Superrechner« benötigt viel Zeit, um einige Millionen Daten durchzusehen. Dieser Zeitfaktor wird noch erhöht, wenn man nur nach bestimmten Teilen einer Datei suchen möchte, also nach bestimmten Buchstabenfolgen oder Zahlenkombinationen.

Gehen wir jetzt einmal davon aus, ein Feld wäre sortiert. Der Computer kann jetzt binär suchen, was selbst bei vielen tausend Elementen nur eine kurze Zeit benötigt. Unter binärer Suche versteht man folgendes: Wenn der Computer zum Beispiel nach einer bestimmten Zahl sucht, so geht er erst einmal zur Mitte des gesamten Feldes. Jetzt kann er anhand eines einfachen Größenvergleichs herausfinden ob die gesuchte Zahl in der einen oder in der anderen Hälfte des Feldes liegen muß. Hat er das herausgefunden, so wird diese Feldhälfte wiederum in der Mitte geteilt, und wieder wird festgestellt, in welcher Hälfte des Feldes die Zahl zu finden sein muß. Dies geht immer so weiter, bis nur noch zwei Zahlen übrigbleiben, von denen eine die gesuchte ist.

Diese Methode der binären Suche ist sehr effektiv und erlaubt selbst bei großen Datenmengen eine geringe Suchzeit. Ein kleines Rechenbeispiel:

Wir haben 100 Elemente und wollen eines davon binär suchen. Durch unser Suchsystem sinkt die Anzahl der zu durchsuchenden Elemente auf folgende Art und Weise:
100:50:25:13:8:4:2:1.

Dies war der ungünstigste Fall, bei dem sich der zu suchende Wert immer in der Hälfte des übriggebliebenen Feldes befand.

Bei 100 Elementen haben wir also maximal sieben Zugriffe, bis der Wert gefunden wird. Nun durchsuchen wir auf die gleiche Weise 1000 Elemente.

Es gilt jetzt die absteigende Reihe:
1000:500:250:125:63:32:16:8:4:2:1.

Wie Sie sehen sind nur drei Zugriffe hinzugekommen, obwohl sich die Anzahl der Elemente verzehnfacht (!) hat.

Mit der anderen Suchmethode hätten wir im Mittel 50 beziehungsweise 500 Zugriffe gehabt, wenn man davon ausgeht, daß die Auswahl gleichverteilt erfolgt. Es lohnt sich also bei größeren Feldern durchaus, diese vorher zu sortieren, wobei wir bei allen Sortiermethoden nur ein Ziel haben werden:

Die Zeit des Sortierens muß möglichst gering bleiben!

Maßgeblich für die Zeitdauer eines Algorithmus sind folgende zwei Kriterien:

1) die Anzahl der Vergleiche
2) die Anzahl der Bewegungen

Außer diesen generellen Kriterien werden wir auch feststellen, daß es eine Rolle spielt, in welcher Form das Feld vor dem Sortieren vorlag. Insgesamt kann man die Sortiermethoden in vier grobe Klassen unterteilen:

1) Sortieren durch Einfügen
2) Sortieren durch Auswählen
3) Sortiern durch Austauschen
4) Sortieren durch Zerlegen

Jede dieser Sortiermethoden hat bestimmte Vorzüge und wiederum auch spezielle Nachteile. Das hängt, wie schon erwähnt, von der anfänglichen Struktur eines Feldes ab. Wir müssen an dieser Stelle zwischen drei verschiedenen Anfangszuständen unterscheiden:

1) Das Feld ist bereits sortiert
2) das Feld ist völlig unsortiert
3) das Feld ist genau entgegengesetzt sortiert.

Manche Sortieralgorithmen sind um so schneller, je sortierter ein Feld vorliegt. Bei anderen Algorithmen kann das genau umgekehrt sein. Quicksort ist zum Beispiel am effektivsten, wenn es sich um zufällig durchmischte Felder handelt.

Direktes Einfügen

Nun aber zu unserem ersten Sortieralgorithmus, einer Sortiermethode, die Ihnen auch im täglichen Leben sicherlich am geläufigsten ist.

Es handelt sich um das Sortieren durch direktes Einfügen (straight insertion). Die hochtrabende Bezeichnung beschreibt einen eigentlich ganz einfachen Vorgang: Sie haben ein Feld aus zufällig durchmischten Elementen. Das Sortierprogramm beginnt jetzt beim zweiten Element und vergleicht dies mit dem ersten; ist es kleiner, so wird getauscht. Die ersten beiden Elemente dieses Feldes sind also schon sortiert.

Jetzt wird das dritte Element geholt und mit dem zweiten verglichen. Ist es größer, so bleibt es an seinem Platz; ansonsten wird es in die Reihe der vorherigen an den richtigen Platz geschoben und eingefügt.

Das geht weiter, bis zum letzten Element, und mit einem Durchlauf werden alle Variablen in aufsteigender Reihenfolge sortiert.

Dieses Sortieren wenden Sie zum Beispiel immer beim Kartenspielen an, wobei Sie Ihr Blatt systematisch durchgehen, alle verkehrt sitzenden Karten herausnehmen und an der richtigen Stelle einordnen.

In Listing 1 sehen Sie ein Programm abgedruckt, das für alle weiteren Sortierprogramme als Rahmen dienen soll. Es hat die Aufgabe, ein Feld zu erstellen und die Ausgabe auf Drucker oder Bildschirm festzulegen. Das Feld kann wahlweise zufällig oder von Hand bestimmt werden und besteht aus Stringvariablen der Länge 3.

In Listing 2 sehen Sie den Abschluß des Sortierprogramms. Alle Algorithmen sind nur mit diesen Rahmenprogrammen lauffähig.

Listing 3 schließlich zeigt ein Programm für das Sortieren

— Methoden, Techniken, Programme

```
10 REM ERSTELLEN EINES FELDES ZUM          <106>
20 REM SORTIEREN.                          <140>
30 REM DAS ERSTELLEN KANN ZUFAELLIG        <254>
40 REM ODER GEZIEHLT (DURCH EINGABE)       <239>
50 REM ERFOLGEN.                           <065>
60 REM                                     <203>
70 REM SORTIERALGORITHMEN ERHALTEN DIE     <092>
80 REM ZEILENNUMMERN VON 10000 BIS 50000   <127>
90 REM SIE BENOETIGEN JEWEILS DIESEN       <117>
99 REM VORSPANN ZUR AUSFUEHRUNG.           <229>
100 REM HERSTELLUNG EINES ARRAYS:          <192>
110 REM ARRAYVARIABLE       - A$           <084>
120 REM SCHLEIFENVARIABLEN - X, Y, Z       <188>
130 REM HILFSVARIABLEN      - B$, C$, D$   <213>
140 REM DREIECKTAUSCH MIT   - S$           <104>
150 PRINT"{CLR}":CLR                       <220>
160 PRINT"SOLL VON{SPACE,RVSON}H{RVOFF}AND
    ODER{SPACE,RVSON}Z{RVOFF}UFAELLIG ERS
    TELLT":PRINT                           <028>
170 INPUT"WERDEN ";X$                      <239>
180 IF X$<>"H"AND X$<>"Z"THEN 150          <021>
190 IF X$="H"THEN GOSUB 220:GOSUB 1000:GOT
    O 210                                  <169>
200 GOSUB 220:GOSUB 2000                   <114>
210 GOTO 4000: REM WEITERMACHEN            <100>
220 REM ANZAHL DER ELEMENTE BESTIMMEN      <247>
230 PRINT:INPUT"ANZAHL DER ELEMENTE ";A    <230>
240 IF A>10000 THEN PRINT:PRINT"ZU VIELE E
    LEMENTE":GOTO 230                      <128>
250 IF A<10 THEN PRINT:PRINT"ZU WENIGE ELE
    MENTE":GOTO 230                        <070>
255 DIM A$(A)                              <124>
260 INPUT"{2DOWN,RVSON}D{RVOFF}RUCKER ODER
    {SPACE,RVSON}B{RVOFF}ILDSCHIRM ";Y$    <011>
270 IF Y$<>"D"AND Y$<>"B"THEN 260          <088>
280 IF Y$="D"THEN D=4:GOTO 300             <130>
290 D=3                                    <076>
300 RETURN                                 <187>
1000 REM EINGABE VON HAND                  <115>
1010 PRINT"{CLR}"                          <101>
1020 PRINT:PRINT"SIE MUESSEN JETZT"A" ELEM
     ENTE EINGEBEN!":PRINT:PRINT           <025>
1030 PRINT"JEDES ELEMENT BESTEHT AUS 3 ZEI
     CHEN.":PRINT:PRINT                    <097>
1040 FOR X=1 TO A                          <181>
1050 PRINT X". ";:INPUT"ELEMENT";A$(X)     <018>
1060 IF LEN(A$(X))<>3 THEN 1050            <217>
1070 NEXT X                                <012>
1080 RETURN                                <202>
2000 REM ZUFAELLIGE EINGABE                <057>
2010 PRINT"{CLR}"                          <081>
2020 PRINT:PRINT"ES WERDEN JETZT"A" ELEMEN
     TE ZUFAELLIG":PRINT:PRINT"AUSGEWAEHLT
     "                                     <036>
2030 PRINT:PRINT"JEDES ELEMENT BESTEHT AUS
      3 ZEICHEN.":PRINT:PRINT              <031>
2040 FOR X=1 TO A                          <160>
2050 A$(X)=""                              <014>
2060 FOR Y=1 TO 3:A$(X)=A$(X)+CHR$(INT(RND
     (TI)*25)+65):NEXT Y                   <094>
2070 NEXT X                                <248>
2080 RETURN                                <150>
3000 REM ZWISCHENAUSGABE DER ELEMENTE      <223>
3010 FOR I=1 TO A-9 STEP 10                <077>
3020 FOR J=I TO I+9:PRINT#1,A$(J)" ";:NEXT
      J                                    <033>
3030 PRINT#1:NEXT I                        <175>
3040 RETURN                                <121>
4000 REM WEITERMACHEN                      <186>
4005 OPEN 1,D                              <244>
4010 PRINT"{CLR}AUSGABE DES ERSTELLTEN FEL
     DES"                                  <178>
4020 PRINT                                 <092>
4030 GOSUB 3000                            <029>
4040 REM SORTIERUNG STARTET                <159>
4050 REM                                   <112>
```

Listing 1. Dieses Programm erstellt das Sortierfeld und ist der Rahmen für alle Sortierroutinen, die nur zusammen mit Listing 1 und Listing 2 laufen

```
50000 REM ENDEBEHANDLUNG                   <150>
50010 PRINT#1                              <230>
50020 GOSUB 3000                           <119>
50030 PRINT#1,A;" ELEMENTE"                <053>
50040 PRINT#1:PRINT#1:CLOSE 1              <018>
50050 END                                  <197>

Ⓒ 64'er
```

Listing 2. Der Abschluß des Sortierprogramms. Es muß zusammen mit Listing 1 und der Sortierroutine (Listing 3 oder 4) gestartet werden

```
10000 REM SORTIEREN DURCH DIREKTES          <082>
10010 REM EINFUEGEN                        <102>
10020 REM                                  <218>
10030 REM STRAIGHT INSERTION               <005>
10040 FOR X=2 TO A                         <001>
10050 IF A$(X)>=A$(X-1) THEN 10120         <234>
10060 REM EINFUEGEN DES ELEMENTS           <209>
10070 X$=A$(X): FOR Y=X-1 TO 1 STEP-1      <013>
10080 A$(Y+1)=A$(Y)                        <050>
10090 IF X$<=A$(Y-1) THEN 10110            <130>
10100 A$(Y)=X$: GOTO 10120                 <143>
10110 NEXT Y                               <128>
10120 GOSUB 3000: REM AUSGABE              <192>
10130 NEXT X                               <147>
10140 REM ENDE                             <110>

Ⓒ 64'er
```

Listing 3. Der einfachste Sortieralgorithmus: Straight Insertion oder Sortieren durch direktes Einfügen

```
HVK GNS ALJ PJW CHS GMO TCE LUP BXJ QXA
GNS HVK ALJ PJW CHS GMO TCE LUP BXJ QXA
ALJ GNS HVK PJW CHS GMO TCE LUP BXJ QXA
ALJ GNS HVK PJW CHS GMO TCE LUP BXJ QXA
ALJ CHS GNS HVK PJW GMO TCE LUP BXJ QXA
ALJ CHS GMO GNS HVK PJW TCE LUP BXJ QXA
ALJ CHS GMO GNS HVK PJW TCE LUP BXJ QXA
ALJ CHS GMO GNS HVK LUP PJW TCE BXJ QXA
ALJ BXJ CHS GMO GNS HVK LUP PJW TCE QXA
ALJ BXJ CHS GMO GNS HVK LUP PJW QXA TCE
-----------------------------------------
ALJ BXJ CHS GMO GNS HVK LUP PJW QXA TCE
 10  ELEMENTE
```

Bild 1. Sie sehen die Entwicklung eines Feldes beim Sortieren durch direktes Einfügen. Die oberste Reihe ist der Ausgangszustand, die unterste Reihe das Ergebnis

durch direktes Einfügen. Wie Sie aus dem Listing erkennen können, ist es wichtig, daß das erste Element des Feldes nicht, oder als das absolut kleinste Element definiert wird, da es die letzte und höchste Vergleichstufe darstellt und somit nicht mehr vertauscht werden kann, da das Programm sonst über die Grenzen des Feldes hinaus arbeiten müßte. In unserem Fall ist dieses Element (A$(0)) ein Leerstring ("").

Bild 1 zeigt, wie Straight Insertion arbeitet. Die Elemente, die jeweils behandelt werden, sind unterstrichen.

Aus Bild 1 können Sie aber noch mehrere Informationen über den Sortieralgorithmus erhalten. Es wird zum Beispiel deutlich, daß das Sortieren durch direktes Einfügen bei a

Elementen genau a-1 Elemente durchgehen muß, um vollständig zu arbeiten.

Diese Zahl ergibt für die Berechnung der Anzahl der notwendigen Vergleiche folgende Formel:

$$(a^2+a)/4$$

Wir haben in unserem Beispiel (Bild 1) mit 10 Elementen gearbeitet. Die Anzahl der Vergleiche beträgt also nach dieser Formel 28.

Für die Anzahl der Bewegungen im Variablenfeld sieht die Sache folgendermaßen aus:

$$(a^2+9a)/4$$

Hier kommen wir gar auf 48 Bewegungen innerhalb unserer 10 Feldelemente.

Diese Formeln lassen an sich gar nichts Schlimmes vermuten. Wenn wir sie jedoch einmal genauer unter die Lupe nehmen, so werden wir eine bestürzende Feststellung machen: beide Formeln haben im Nenner jeweils einen Faktor a^2 stehen.

Anders ausgedrückt heißt das; wenn wir die Anzahl der Elemente verdoppeln, vervierfacht sich die Anzahl der Bewegungen, der Vergleiche und ebenso natürlich die Sortierdauer.

Bei einer dreifachen Anzahl müssen wir schon neunmal (!) solange warten, wie zu Beginn.

Wie schon erwähnt, besteht das Ziel des effektiven Sortierens darin, die Zeitdauer möglichst gering zu halten. In der Praxis werden wir versuchen, die Anzahl der Bewegungen und Vergleiche auf ein Mindestmaß zu drücken, und wir werden erkennen, daß sich die Proportionalität von a zu a^2 auf a zu $\log_2$ (a) (Logarithmus der Basis 2) vermindern läßt, wenn man entsprechende Algorithmen einsetzt.

Bubblesort

Nachdem wir einen sehr einfachen Algorithmus bereits kennengelernt haben, soll uns nun eine weitere, recht einfache Sortiermethode interessieren. Es handelt sich hierbei um ein Sortieren durch Austauschen.

Bubblesort zählt mit zu den bekanntesten Sortieralgorithmen und arbeitet nach folgendem Prinzip:

Wir fangen mit dem gesamten Variablenfeld an. Hier nehmen wir nun das erste Element und vergleichen es mit dem zweiten. Ist es größer, so wird getauscht; ansonsten bleiben die beiden

```
SPU IOF CEH FSO AIF XKY BHW QTR OPC KBL
IOF CEH FSO AIF SPU BHW QTR OPC KBL XKY
CEH FSO AIF IOF BHW QTR OPC KBL SPU XKY
CEH AIF FSO BHW IOF OPC KBL QTR SPU XKY
AIF CEH BHW FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
AIF BHW CEH FSO IOF KBL OPC QTR SPU XKY
  10  ELEMENTE
```

Bild 2. In diesen Schritten wird beim Bubblesort sortiert. Die oberste Reihe ist das unsortierte Feld und die unterste Reihe zeigt das Ergebnis nach dem Sortieren

```
10000 REM SORTIEREN DURCH AUSTAUSCHEN      <059>
10010 REM                                  <208>
10020 REM                                  <218>
10030 REM BUBBLESORT                       <216>
10040 FOR X=A-1 TO 1 STEP-1                <097>
10050 FOR Y=1 TO X                         <034>
10060 IF A$(Y)<=A$(Y+1)THEN 10080          <252>
10065 REM AUSTAUSCHEN BEIDER ELEMENTE      <069>
10070 S$=A$(Y):A$(Y)=A$(Y+1):A$(Y+1)=S$    <231>
10080 NEXT Y                               <098>
10090 GOSUB 3000: REM AUSGABE              <162>
10100 NEXT X                               <117>
```

© 64'er

Listing 4. Bubblesort ist ebenfalls eine einfache, aber auch nicht gerade die schnellste Sortiermethode

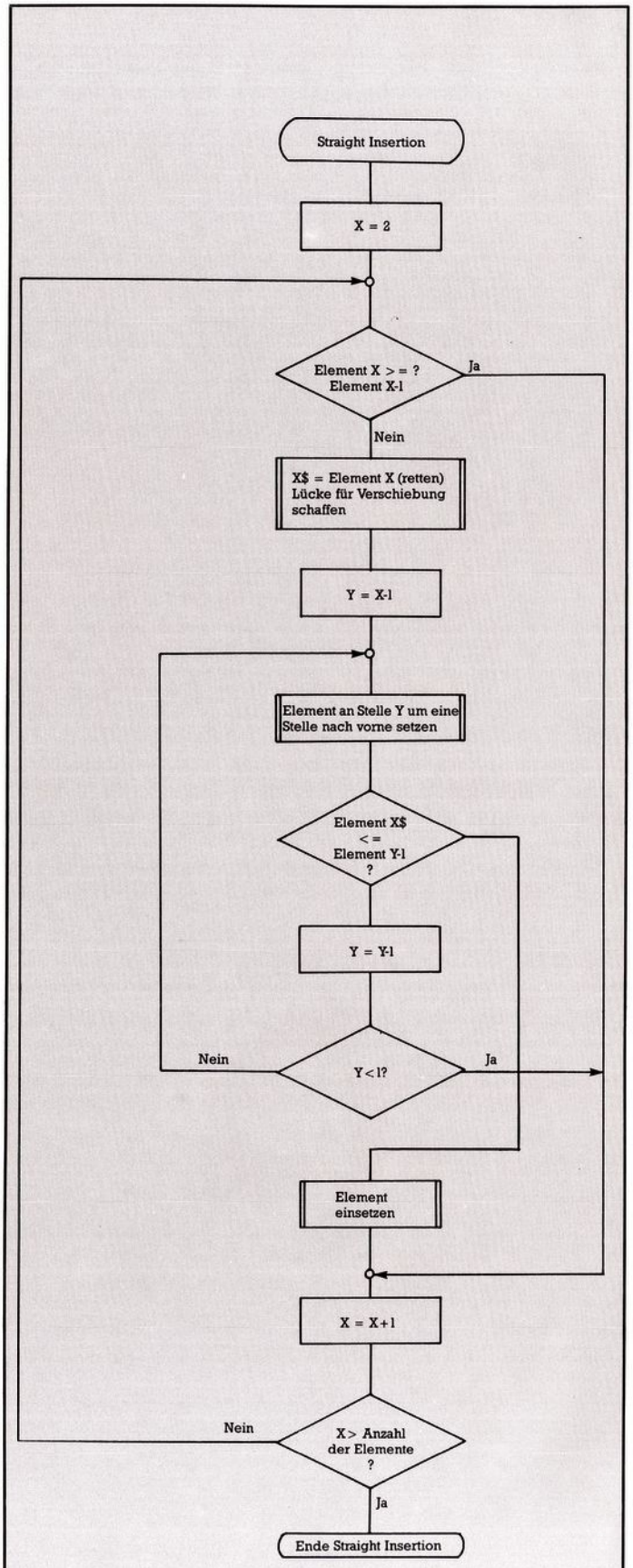

Bild 3. Das Flußdiagramm für Straight Insertion, dem Sortieren durch direktes Einfügen

Elemente, so wie sie sind, stehen. Jetzt gehen wir eine Position weiter und vergleichen das jetzige zweite Element (das auch das vorherige erste sein kann) mit dem dritten der Reihe und tauschen gegebenfalls aus. Das geht immer so weiter, bis zum Ende des Feldes.

Sie werden sicherlich erkannt haben, daß sich auf diese Weise das allergrößte Element immer weiter nach unten bewegt hat und nach Abschluß dieses Durchgangs an lezter Stelle zu finden (also bereits richtig einsortiert) ist.

Jetzt begrenzen wir also das gesamte Feld auf alle Variablen, bis auf die letzte (a = a-1) und wiederholen den Vorgang. Als Ergebnis steht nun das zweitgrößte Element an der vorletzten Stelle und wir vermindern die Gesamtzahl wiederum um 1.

Das geht so weiter, bis die Länge des Feldes auf 1 geschrumpft ist; wir also nur noch das kleinste Element übrig haben. Damit ist der Sortiervorgang beendet.

Der Name von Bubblesort kommt übrigens von der Eigenschaft dieses Verfahrens, die größten Elemente quasi bis ans Ende des Feldes »durchzuperlen«. Dreht man das Feld um und hat man die größten Elemente am Anfang, so kann man diese Bewegungen innerhalb der Variablen durchaus mit dem Aufsteigen von Blasen (»bubbles«) vergleichen.

Listing 4 zeigt das Programm für den einfachen Bubblesort-Algorithmus, und in Bild 2 können Sie wiederum einen Beispielausdruck mit 10 Elementen sehen. Der erste Unterschied zwischen Bubblesort und unserem vorherigen Straight Insertion wird sofort klar, wenn Sie sich die beiden Ausdrucke im Vergleich betrachten.

Während sich das Feld bei Straight Insertion vom Anfang her aufbaut und beim kleinsten Element zu sortieren beginnt, fängt Bubblesort beim größten Element an und bringt dieses zuerst an dessen Platz.

Die Zeitbedingungen für Bubblesort sind denen von Straight Insertion ziemlich ähnlich. Auch hier haben wir den Faktor a^2 als zeitbestimmenden Faktor in den Formeln.

Die Formel für die Anzahl der Vergleiche lautet jetzt:

$$(a^2-a)/2$$

Um die Anzahl der Bewegungen zu berechnen dient folgende Formel:

$$3/4*(a^2-a)$$

Bild 3 und 4 zeigen einen Programmablaufplan der beiden Sortiermethoden, so daß eine Umstellung auf andere Programmiersprachen kein Problem darstellen sollte.

An dieser Stelle wollen wir den ersten Abschnitt unserer Folge bereits beenden. Überlegen Sie sich bis zum nächstenmal, wie man Bubblesort vielleicht noch verbessern könnte; wir bringen dann nämlich eine Version, die einige Nachteile der jetzigen nicht mehr besitzt.

Oder vielleicht fallen Ihnen inzwischen auch einige Methoden zum günstigen Sortieren von Feldern ein?

Sicherlich werden Sie die eine oder andere Möglichkeit im Laufe unserer Reihe noch finden, wenn auch unter einem vielleicht noch unbekannten Namen.

(Karsten Schramm/gk)

Wichtige Begriffe in diesem Artikel

Feld, Variablenfeld

(auch Matrix, Tabelle, indizierte Variable)

In einem Feld wird eine Gruppe von in der Regel gleichartigen Werten (Daten) zusammengefaßt. Felder müssen dimensioniert werden. Dabei wird entsprechend der Datenmenge Speicherplatz reserviert. Im Commodore-Basic können sowohl numerische als auch alphanumerische Felder definiert werden.

Algorithmus

Ein Begriff aus der Mathematik, der genau festlegt, wie ein Problem zu lösen ist. Im einfachsten Fall ist das eine Formel, zum Beispiel FLÄCHE = LÄNGE x BREITE. Aber selbst ein komplettes Programm kann als Algorithmus bezeichnet werden, denn es wird ein Problem nach vorgegebenen Regeln gelöst.

Binäre Suche

Die binäre Suche ist eine sehr schnelle Suchmethode, die ein sortiertes Feld voraussetzt. Dabei wird, vom gesamten Feld ausgehend, immer auf das Element in der Mitte des Feldes zugegriffen. Je nachdem, ob der gesuchte Wert (oder Wort) kleiner oder größer ist, wird entweder die übrigbleibende obere oder untere Hälfte wiederum halbiert und auf das mittlere Element in dieser verbleibenden Hälfte zugegriffen und mit dem gesuchten Begriff verglichen. Dieses Spiel wiederholt sich so lange, bis der Begriff gefunden wurde. Die maximale Anzahl der Suchschritte errechnet sich aus max = INT (LN(anz. Elemente)/LN(2)) +1.

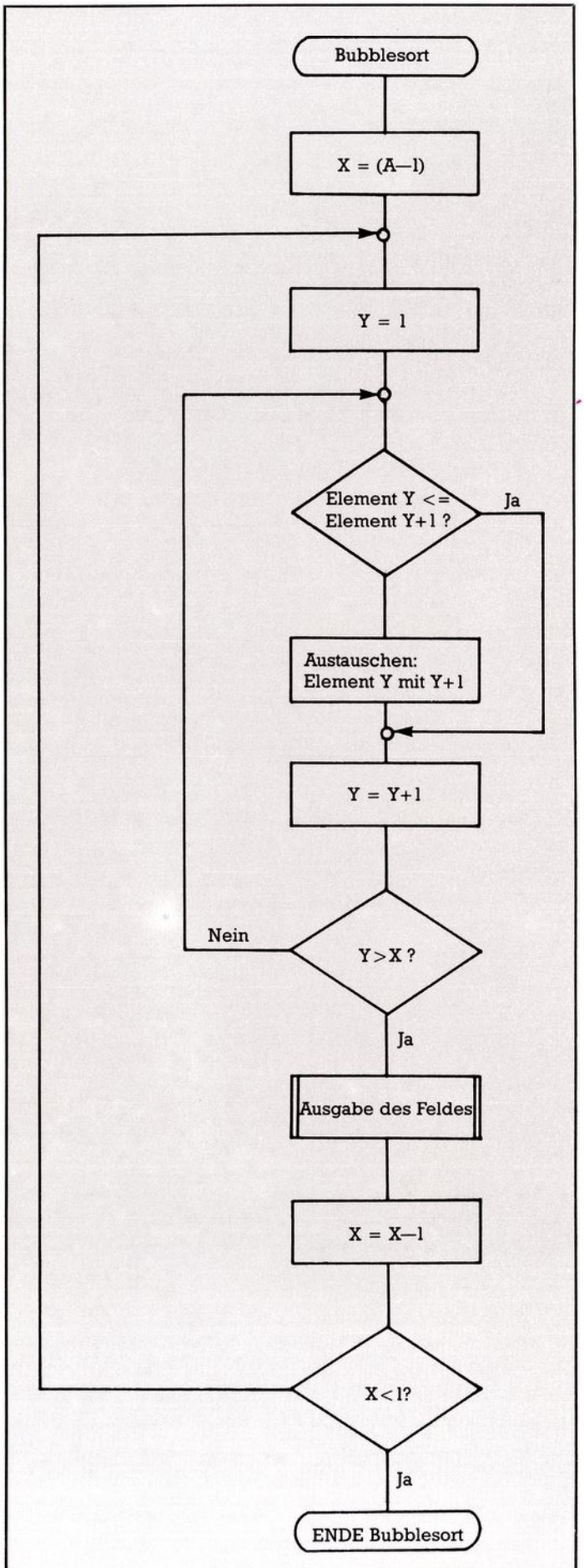

Bild 4. Das Flußdiagramm vom Bubblesort-Verfahren