87-6

MIKRO + KLEIN COMPUTER

Klimaauswertung in Turbo-Pascal

Prozeduren für die Benutzerschnittstelle

DAS SCHWEIZER FACHMAGAZIN FÜR KLEINE UND MITTLERE COMPUTERSYSTEME Fr. 8.–

DIE BESTE LÖSUNG
für Ihre
DATENSICHERUNG

Einfach, schnell, automatisch und voll netzwerkfähig

Die GALAXY TAPE Software zu den GENOA GALAXY Tape Backup-Systemen macht Ihre Datensicherung einfach und schnell. Wählen Sie Ihre Optionen aus dem übersichtlichen Menü, drücken Sie ein paar Tasten und vier Minuten später ist Ihre 20-MB-Festplatte vollkommen gesichert.

AUTOMATISCH

Wenn Sie wollen, macht der GENOA GALAXY Streamer Ihre Datensicherung von selbst, auch täglich. Falls Sie zum Zeitpunkt der automatischen Sicherung mit Ihrem Computer arbeiten, erinnert Sie die GALAXY Software daran, daß eine Datensicherung ansteht. Sie wartet aber auch, bis Sie mit Ihrer Arbeit fertig sind, um erst dann, wiederum von selbst, Ihre Festplatte zu sichern. Dabei zeigt GALAXY jederzeit am Bildschirm an, was sie macht.

NETZWERK UPGRADE

Wollen Sie später einmal vernetzen, oder arbeiten Sie schon im Netz, dann können Sie GenWare™ zusätzlich installieren, um Ihre Daten ebenso schnell wie einfach in Ihrem NOVELL-Netzwerk zu sichern. Außerdem können Sie auf diese Weise Ihre Daten beliebig zwischen Netzwerk und stand-alone-PCs transportieren.

Für alle NETBIOS-kompatiblen Netze gibt es außerdem NetSafe, wie GenWare eine spezielle Netzwerk-Tape-Software, die zusätzlich installiert, den Streamer zu einer vollen Netzwerkanwendung on-line ins Netz integriert, ohne daß er irgendwelche seiner Eigenschaften dabei verliert.

GENOA hat die Antwort auf Ihre jetzigen und zukünftigen Backup-Probleme: Eine ganze Palette von Tape-Backup-Geräten, von 20 bis 120 MB, die alle gleichermaßen einfach, schnell und automatisch funktionieren.

Sie erhalten die GENOA GALAXY TAPE BACKUP SYSTEME wie unsere anderen Produkte nur über den autorisierten PC-Fachhandel.

5 MB pro Minute! Genoa's menügesteuerte Software macht es Ihnen leicht!

Electronic Marketing AG

Your Swiss Distributor for high technology.

Electronic Marketing AG · Bahnhofstr. 60 · 4132 Muttenz-Basel · Tel. 061-61 53 53 · Tlx 965669 · Fax 061-61 48 60

Mathematik (1. Teil)

Wussten Sie, dass die Wahrscheinlichkeit, im deutschen Zahlenlotto «6 aus 49» sechs Richtige zu haben, genau 1:13'983'816 beträgt? Oder wie gross die Chance ist, dass bei 50-maligem Werfen einer Münze mindestens 30 Mal die Zahl zum Vorschein kommt? Wenn Sie diese und ähnliche Aufgabenstellungen aus dem Bereich der Wahrscheinlichkeitsrechnung und Statistik interessieren, sollten Sie jetzt weiterlesen. Auch wenn Sie keine weiteren mathematischen Kenntnisse mitbringen, können Sie allein mit dem gesunden Menschenverstand Aussagen über den Ablauf von Ereignissen machen. Und damit auch der Computer nicht zu kurz kommt, werden die Erkenntnisse natürlich in Programme umgesetzt. Diese sind in BASIC geschrieben und meist ohne Abänderung auf jedem Rechner mit BASIC-Interpreter lauffähig.

Michael Schlingmann

Ohne Mathematik kann man keine präzisen Aussagen über die Wahrscheinlichkeit machen. Doch keine Angst, ich möchte Sie hier nicht mit einem Formelwust belasten. Viele Gleichungen sind unmittelbar einleuchtend, die anderen kann man ohne allzu grosse Schwierigkeiten herleiten.

Beginnen wir mit einem Würfelspiel. Gegeben sind zwei handelsübliche Würfel. Gefragt ist nach der Wahrscheinlichkeit, mit einem Wurf mindestens elf Augen zu erzielen.

Damit man eine Wahrscheinlichkeit (im Folgenden mit WS abgekürzt) überhaupt angeben kann, muss man sie auf irgendein Eichmass beziehen. Die praktisch immer benutzte Definition lautet

$$WS = \frac{\text{Anzahl der günstigen Ereignisse}}{\text{Insgesamt mögliche Ereignisse}}$$

Ein günstiges Ereignis ist in unserem Fall ein Wurf mit mindestens elf Augen, da wir diesen ja haben wollen. Die insgesamt möglichen Ereignisse sind alle Zahlenkombinationen, die mit zwei Würfeln erreichbar sind. Diese hat man zu berechnen, bevor man weitermachen kann.

Dazu zerlegen wir das Problem in zwei kleinere Teilprobleme: Wieviel Möglichkeiten gibt es für einen einzigen Würfeln? Sechs natürlich, werden Sie sagen. Damit haben Sie den ersten Schritt im Gebiet der Wahrscheinlichkeitsrechnung getan und der Rest sind eigentlich nur logische Folgerungen.

Wenn man annimmt, dass der eine Würfel den anderen Würfel nicht beeinflusst, kann man sich dem zweiten Problem in ähnlicher Weise widmen: Für den ersten Würfel gibt es sechs Möglichkeiten. Dasselbe muss auch für den zweiten Würfel gelten. Angenommen, der zweite Würfel zeigt eine Eins. Diese Eins kann man mit den sechs Möglichkeiten des ersten Würfels kombinieren. Mit der Zwei und den folgenden Zahlen verhält es sich natürlich genauso. Insgesamt gibt es also 6*6 = 36 Alternativen, zwei Würfel zu kombinieren.

In diesen Kombinationsmöglichkeiten sind natürlich auch die Fälle eingeschlossen, dass die Augenzahl kleiner als elf ist. Im Weiteren interessieren wir uns nur für die oben genannten «günstigen» Fälle. Es gibt, wie schnell zu erkennen ist, nur drei Möglichkeiten:

5 + 6
6 + 5
6 + 6

Die WS für mindestens elf Augen beträgt demzufolge also 3/36.

Ein Rechnerprogramm zur Lösung des Problems zu schreiben, ist denkbar einfach: Man lässt die Augenzahl eines Würfels konstant und variiert die des anderen, wobei überprüft wird, ob die Summe mindestens elf ist. Dieses Beispiel ist aber zu trivial, man kann es auch im Kopf rechnen. Hierbei umgeht man die vielen «Misserfolge» die der Rechner hat, indem man mit dem grössten Augenzahlen zuerst anfängt.

Bevor wir weitergehen, noch das gegenteilige Beispiel: Wie gross ist die WS, höchstens zehn Augen zu würfeln?

Natürlich kann man auch hier wieder mit dem Abzählen beginnen. Es geht jedoch auch viel einfacher. Die obige Fragestellung ist genau das Umgekehrte zu unserem schon gelösten Problem: Wenn die WS für mindstens elf Augen gleich 3/36 ist, muss die WS für das Nichteintreten dieses Falls gleich 33/36 sein, da die Summe aller WS immer gleich Eins sein muss (gegeben durch die Festlegung unseres Eichmasses).

Dieser Selbstverständlichkeit liegt ein sehr wichtiges Prinzip der Wahrscheinlichkeitsrechnung zugrunde: Wenn einem die Berechnung der WS eines Ereignisses als zu aufwendig erscheint, so versucht man die WS für das Nichteintreten zu finden. Sei a die WS, dass das Ereignis eintritt und bezeichnet $\bar{a}$ die WS, dass es nicht stattfindet, so gilt die Verknüpfung $a = 1 - \bar{a}$.

Fakultäten

Bevor wir uns komplizierteren Fallstellungen zuwenden können, ist die Einführung einer Rechengrösse notwendig, die man «Fakultät» nennt, die aber nichts mit den Lehrern an Universitäten zu tun hat.

Zur Motivation das Problem, das sich stellt, wenn man sich vier neue Reifen für sein Auto kauft und diese selbst montieren will. Für das erste Rad hat man noch vier Möglichkeiten zur Montage. Nimmt man den zweiten Reifen, so sind nur noch drei Felgen frei. Gemäss den obigen Erkenntnissen gibt es bisher 4*3 = 12 mögliche Kombinationen. Für den dritten Reifen sind nurmehr zwei Stellen frei, die Position des letzten Reifens ist dann zwangsläufig festgelegt. Insgesamt gibt es also

4*3*2*1 = 24:= 4!

Möglichkeit zur Anordnung der Reifen. Das Ausrufezeichen nennt man «Fakultät», es ist eine Kurzbezeichnung des Produktes.

Schon die Fakultät von Acht ist eine ziemlich unübersichtliche Zahl:

8! = 40'320

Hier lohnt es sich also schon eher, den Rechner einzusetzen. Ein Programm dafür ist schnell ausgedacht, indem man eine Schleife definiert und deren Werte mit sich selbst multipliziert. Doch merkt man bald, dass man schon bei verhältnismässig kleinen Fakultäten an die Grenzen des Computers stösst, dessen Zahlenbereich in der Regel höchstens bis 10^{40} geht.

Will man nun aber zum Beispiel berechnen, wie viele Möglichkeiten es für die 200 Mitglieder des Schweizer Nationalrates gibt, sich zu setzen (geht man einmal davon aus, dass es im Nationalrat genau 200 Sitzplätze gibt), so steht man zunächst einmal vor einem schwerwiegenden Problem: Wie bekomme ich diese Zahl in den Rechner hinein und vor allem die Fakultät wieder heraus?

Die Lösung des Problems ist denk-

bar einfach, wenn man sich mit den Logarithmen etwas auskennt: Es gilt

$$\lg(a*b) = \lg(a) + \lg(b)$$

und damit

$$\lg(n!) = \lg(1) + \lg(2) + \ldots + \lg(n)$$

Als Programm sieht das dann so wie in Listing 1 aus. Es gibt also die lächerliche Anzahl von etwa 10^{375} Kombinationsmöglichkeiten, das ist eine Eins mit 375 Nullen hintendran.

Das Lottospiel

Nachdem dieses Problem gemeistert ist, können wir uns nun tiefschürfenderen Fragen zuwenden, so zum Beispiel dem Zahlenlotto. Hier gilt es, in einem Feld mit den Zahlen zwischen 1 und 49 sechs Zahlen so anzukreuzen, auf dass man nach der Ziehung der Lottozahlen Millionär ist.

Betrachten wir das Problem einmal vom Stand des Theoretikers: Es liegt hier der Fall von n Ziehungen ohne Zurücklegen vor. *Ohne Zurücklegen* heisst, dass die Zahl aus dem Rennen ausscheidet, wenn sie einmal gezogen ist. Sie darf also nicht doppelt vorkommen.

Die Lösungsweise ist analog zu der mit den Reifen: Am Anfang gibt es 49 verschiedene Zahlen, die gezogen werden können usw. Das ergibt also

$$49*48\ldots*44$$

Einige Kombinationen kommen aber mehrfach vor: Zum Beispiel ist es egal, ob die Zwei zuerst oder ganz am Schluss gezogen wurde, Hauptsache sie wurde gezogen. Die Fragestellung ist gleich wie beim Reifenproblem. Hier wurde auch nicht gefragt, welcher Reifen zuerst wohin kommt, es interessierte nur das Resultat, nämlich die Anzahl der Möglichkeiten, alle Reifen aufzuziehen.

Beim Lotto gibt es also demzufolge 6! Anordnungsmöglichkeiten der sechs Zahlen.

Da es nur eine Möglichkeit gibt, sechs Richtige zu haben (fünf Richtige sind zwar auch nicht schlecht, in diesem Zusammenhang aber falsch), ist die WS dafür

$$\frac{1}{\frac{49*48*47*46*45*44}{6*5*4*3*2*1}} = \frac{1}{13983816}$$

$$\frac{n(n-1)*\ldots*(n-k+1)}{1*2*3*\ldots*k} = \frac{n!}{k!(n-k)!}$$

Spielen Sie Lotto?

Da der obige Term etwas umfangreich ist, haben sich die Mathematiker wieder eine einfachere Schreibweise einfallen lassen: Die Binomialkoeffizienten, die wir noch einige Male brauchen werden. Es gilt

$$\frac{n!}{k!(n-k)!} := \binom{n}{k}$$

Ein Programm zur Ermittlung der Binomialkoeffizienten nach dem Muster des Fakultätsprogramms dürfte eigentlich nicht schwerfallen.

Fussballtoto

Der Sinn des Fussballtotos ist schnell erklärt: Es gibt insgesamt elf Spiele. Die Heimmannschaft bekommt eine Eins, wenn sie gewinnt, eine Null bei Unentschieden und eine Zwei falls sie verliert.

Ein solche Zahl ist möglichst richtig in jedes der elf Kästchen einzutragen. Hauptgewinner ist der, der elf Richtige erzielt hat. Wie gross ist die Wahrscheinlichkeit dafür?

Die Verhältnisse liegen etwas anders als beim Lotto. Denn dieses Spiel wird *Mit Zurücklegen* durchgeführt. Auch wenn z.B. im Spiel 1 die Eins richtig ist, so kann sie es im Spiel 4 auch wieder sein. Eine einmal gezogene Zahl scheidet also nicht aus. Betrachten wird das Spiel 1:

Die Wahrscheinlichkeit, das Spiel richtig zu tippen, beträgt ein Drittel. Dasselbe gilt für das zweite Spiel usw. Um weiterrechnen zu können, wenden wir einen wichtigen Satz der Wahrscheinlichkeitsrechnung an, den wir schon einige Male stillschweigend benutzt haben: Wenn zwei Ereignisse voneinander unabhängig sind, ist die WS dafür, dass beide Ereignisse eintreten, das Produkt aus den beiden Einzelwahrscheinlichkeiten. Dieses Ergebnis kann auf beliebig viele Ereignisse übertragen werden.

Nehmen wir an, dass der Ausgang eines Spiels nichts damit zu tun hat, wie eine andere Mannschaft gespielt hat, so erhalten wir für das Totospiel die WS

$$\frac{1}{3} *\ldots* \frac{1}{3} = \frac{1}{3^{11}} \approx 0.0000056$$

Das heisst, im Durchschnitt werden von einer Million Wetten etwa sechs gewinnen.

Die beiden obigen Beispiele kann man beliebig ausbauen. Statt der 49 Lottokugeln kann man auch vier rote und fünf grüne Kugeln in einen Topf werfen und nach der Wahrscheinlichkeit fragen, dass genau drei gezogene Kugeln rot sind. Je nachdem, ob man eine Kugel nach der Ziehung wieder in den Topf zurücklegt oder nicht, erhält man verschiedene Ergebnisse.

Wenn Sie in einen Karton mit Glühbirnen greifen und wissen, dass jede Zwanzigste kaputt ist, können Sie mit den gerade gelernten Formeln die WS dafür berechnen, dass unter drei herausgegriffenen Glühbirnen alle drei nicht funktionsfähig sind.

Wollen Sie wissen, wie hoch die WS ist, mindestens zwei ganze Glühbirnen zu erwischen, brauchen Sie nur die jeweils günstigen Einzel-WS zu addieren.

Zuletzt noch eine Warnung: Wenn Sie versuchen sollten, die WS dafür zu berechnen, im Lottospiel mit den ersten vier gezogenen Zahlen drei Richtige zu haben, so können Sie das auch schaffen. Allerdings ist die Rechnung etwas aufwendiger als in den oben beschriebenen einfachen Fällen.

Vollends kompliziert wird die Sache, wenn Sie eine Münze 1'000 mal werfen und nach der WS fragen, wie oft mindestens 388 mal Zahl erscheint. Doch dazu gibt es andere Rechenmethoden.

Die Binomial-Verteilung

Wenn Erika Hess im alpinen Ski-Worldcup durchschnittlich 60 % aller Rennen gewonnen hat, so heisst das nicht, dass sie bei zehn Rennen aus genau sechs als Siegerin hervorging. Vielmehr handelt es sich hier um ei-

```
1 REM fakultät
10 A = 0
20 INPUT N
30 PRINT N"!= ";
40 FOR I = 1 TO N
50 A = A + LOG(I)
60 NEXT I
65 A = A / LOG(10)
70 IF A < 37 THEN PRINT INT(10^A + .5)
80 IF A >= 37 THEN PRINT 10^(A - INT(A));"E";INT(A)
90 END
```

Listing 1

nen Durchschnittswert. Im Wettkampf werden mehr oder weniger starke Abweichungen von diesem Durchschnittswert auftreten.

Betrachten wir das Ganze etwas mathematischer: Die WS für Gewinn beträgt 0.6, das WS-Experiment lässt nur zwei Ausgänge zu: Entweder sie gewinnt oder sie gewinnt nicht.

Für den Fall, dass die WS bekannt ist und für den Ausgang des Ereignisses nur zwei Möglichkeiten zur Verfügung stehen, hat der Schweizer Mathematiker Jakob Bernoulli eine einfache Formel hergeleitet (die Herleitung ist nicht schwierig):

Sei w die WS, dass das Ereignis eintritt und n die Anzahl der durchgeführten Experimente. Ist nach der WS gefragt, dass genau k Ereignisse stattfinden, so gilt

$$P(k) = \binom{n}{k} * w^k (1-q)^{n-k}$$

Dabei ist P die Wahrscheinlichkeit.

Fragt man sich nun also, wie gross die WS ist, dass Erika Hess von zehn Rennen genau sechs gewinnt, so erhält man also

$$P(6) = \binom{10}{6} * 0.6^6 * 0.4^4 = 0.251 \approx 25\,\%$$

Diese WS liegt deutlich unter Eins (dies würde bedeuten, dass sie grundsätzlich sechs von zehn Rennen für sich entscheiden kann).

Dazu noch ein Beispiel aus der Praxis: Eine Firma vertreibt einen Massenartikel in Paketen zu je 15 Stück. Dabei wird vereinbart, dass Pakete mit mehr als zwei schadhaften Stücken nicht berechnet werden. Wieviel Prozent der ausgelieferten Paketen muss die Firma als unberechnet kalkulieren, wenn ihr bekannt ist, dass durchschnittlich zwei Prozent der Artikel schadhaft sind?

Es gibt nur zwei Ausgänge des Experimentes: Entweder ist der Artikel gut oder er ist schadhaft. Man darf also Bernoulli anwenden und erhält

$$P(k>2) = 1 - P(k=0) + P(k=1) + P(k=2) = 0.0323$$

Im Durchschnitt braucht die Firma also nur etwa 3 % der Pakete als unberechnet zu kalkulieren.

Die Formel von Bernoulli wird in der Wahrscheinlichkeitsrechnung als «Binomial-Verteilung» bezeichnet.

Die Poisson-Verteilung

Bei praktischen Anwendungen der WS-Rechnung sind oft Ereignisse zu betrachten, die nur selten eintreten (z.B. Hühnereier mit zwei Dottern). Wenn die WS aber gering ist, so muss man sehr viele Ereignisse betrachten um ein oder zwei günstige Ereignisse zu bekommen. Die Anwendung der Binomial-Verteilung ist dann recht aufwendig. Beim Einsatz von Computern bekommt man Schwierigkeiten mit der Genauigkeit, weil die meisten Potenzen aufgrund von Rundungen gleich Null werden. Man sucht deshalb nach einer Näherungsformel, die für grosse Werte von n und kleine Werte von q an die Stelle der Bernoulli-Formel treten kann. Diese Arbeit nahm uns vor genau 150 Jahren der Franzose S. D. Poisson ab, der folgende Näherungslösung fand:

$$P_{n;q}(k) \approx \frac{(nq)^k}{k!} e^{-(nq)}$$

Dabei sollte das Produkt n*q nicht grösser als 5 werden, da die Formel sonst zu ungenau wird.

Die Poisson-Verteilung wird z.B. bei der Untersuchung von radioaktiven Zerfällen angewandt.

Die Gauss-Verteilung

Natürlich steht man auch des öfteren vor dem Problem, dass man zwar sehr viele Stichproben hat, die WS für ein günstiges Ereignis aber keineswegs klein ist. Wenn Sie schon einmal einen Intelligenztest gemacht haben, dann werden Sie sich sicher an die glockenförmige Verteilung der verschiedenen IQ erinnern: Es gibt in der Bevölkerung kaum Menschen mit einem IQ kleiner als 50, aber genauso wenige, die mehr als IQ 150 aufweisen können. Die grösste WS für den IQ liegt bei dem Wert 100 (im vorliegenden Fall wurde der grössten WS willkürlich der Wert 100 gegeben, um einen Bezugspunkt zu haben).

Vielleicht haben Sie auch schon von den sogenannten Vertrauensintervallen gehört, bei denen bei Kenntnis des Mittelwerts und der Varianz (das ist die mittlere quadratische Abweichung, wir kommen noch darauf zu sprechen) eine Aussage über bestimmte WS gemacht werden kann.

Die obigen Probleme werden mit einer sehr globalen Wahrscheinlichkeitsverteilung gelöst, die C. F. Gauss als Erster beschrieben hat. Ihm zu Ehren wurde sie «Gauss-Verteilung» benannt.

Je universeller eine Formel einsetzbar ist, umso schwieriger ist sie in der Regel herzuleiten und zu beweisen. Deshalb soll auf die Herleitung der Gauss-Formel verzichtet werden, man kann sie aber in jedem guten Statistik-Buch nachlesen, sollte aber dabei mindestens schon einmal etwas von Integralrechnung gehört haben.

Die Anwendung der Gauss-Formel gelingt aber auch dem Nichtmathematiker.

Vor Einführung der Gauss-Formel (sie wird auch oft «Normalverteilung» genannt) muss aber noch ein anderer Begriff geklärt werden: Die Varianz.

Varianz

Wenn man ein Experiment durchführt und Messwerte aufnimmt, so werden sie selten auf einer Geraden liegen, wie es die Theorie vorhersagt. Vielmehr streuen sie mehr oder weniger stark. Man ist daher bestrebt, eine Gerade so durch die Messwerte zu legen, dass sich die (unter Umständen statistischen) Messfehler weitgehend herausmitteln. Mathematisch geschieht das mit Hilfe der «Linearen Regression», von der in einem anderen Beitrag die Rede sein wird. Zu deren Berechnung muss man natürlich wissen, wie weit die Messwerte vom

```
10 REM BINOMIALVERTEILUNG
20 INPUT "Anzahl der Würfe: "; Q
30 INPUT "Untergrenze Anzahl der günstigen Ereignisse: ";U
40 INPUT "Obergrenze Anzahl der günstigen Ereignisse: ",O
50 INPUT "Wahrscheinlichkeit für günstiges Ereignis: ";P
60 A = 0
70 FOR Y = U TO O
80 UU = Y
90 A2 = 1
100 I = Q - UU + 1
105 IF UU = 0 THEN GOTO 160
110 II = UU
120 A2 = A2 * I / II
130 I = I + 1
140 II = II - 1
150 IF I <= Q GOTO 120
160 A = A + A2 * P ^ UU * ( 1 - P ) ^ ( Q - UU )
170 NEXT Y
180 PRINT "Wahrscheinlichkeit = "; A
190 END
```

Listing 2

```
1 REM GAUSSVERTEILUNG
10 INPUT "Mittelwert :";M
20 INPUT "Standardabweichung :";S
30 INPUT "Untergrenze :";U
40 INPUT "Obergrenze :";O
50 P = .2316419
60 X1 =  ( O - M ) /S
70 X2 =  ( U - M ) /S
80 T1 = 1 / ( 1 + P * X1)
90 T2 = 1 / ( 1 + P * X2 )
100 B1 = .31938153#
105 B2 = -.356563782#
110 B3 =  1.781477937#
120 B4 = -1.821255978#
130 B5 = 1.330274429#
140 Q1 = 1 / SQR ( 2 * 3.14159 ) * EXP ( -X1 * X1 / 2 ) * (B1 * T1 + B2 * T1 * T
1 + B3 * T1 * T1 * T1 + B4 * T1 * T1 * T1 * T1 + B5 * T1 * T1 * T1 * T1 * T1 )
150 Q2 = 1 / SQR ( 2 * 3.14159 ) * EXP ( -X2 * X2 / 2 ) * (B1 * T2 + B2 * T2 * T
2 + B3 * T2 * T2 * T2 + B4 * T2 * T2 * T2 * T2 + B5 * T2 * T2 * T2 * T2 * T2 )
160 R = Q2 - Q1
170 IF R > 0 THEN PRINT "Wahrscheinlichkeit = "; R
180 IF R < 0 THEN PRINT "Wahrscheinlichkeit = "; 1 + R
```

Listing 3

eigentlichen Mittelwert entfernt liegen. Ein Mass dafür die die Varianz, die manchmal auch die «mittlere quadratische Abweichung» genannt wird.

Sie wird in der Regel definiert durch

$$s2 = \frac{1}{n-1} \sum_{i=1}^{n} (\bar{x}-x(i))^2$$

Dabei sind $\bar{x}$ der eigentliche Mittelwert und n die Anzahl der Messwerte. Es werden hier also die Abweichungen vom Mittelwert aufsummiert und durch die Anzahl der Messungen geteilt. Das Quadrat ist notwendig, da sich sonst Abweichungen in verschiedene Richtungen gegenseitig aufheben würden. Auf den Faktor -1 ergibt sich aus der Theorie und es kann an dieser Stelle nicht näher darauf eingegangen werden.

Die Quadratwurzel aus der Varianz heisst übrigens «Standardabweichung». Ein gutes Programm zur Berechnung der Varianz wurde in M+K 87-2 (Schnellere Statistik) abgedruckt.

Wie Laplace zeigte, kann man die Binomialverteilung bei grossen Werten von n durch folgende Näherungsformel approximieren:

$$P(k) \approx \frac{1}{\sigma} \varphi \left(\frac{k-\bar{x}}{\sigma}\right)$$

φ heisst «Gauss-Funktion«. Ihre Werte kann man in mathematischen Tabellenwerken nachlesen. Den Mittelwert $\bar{x}$ berechnet man durch

$$x = n*q$$

σ = SQR(n*q(1-q)) ist die Standardabweichung.

Die Formel gilt in guter Näherung für n>9/(q*(1-q))

Dieses Ergebnis kann man auch mit verhältnismässig wenig Aufwand durch explizite Berechnung der Binomialverteilung erzielen. Ihren wahren Nutzen zeigt die Gauss-Formel erst, wenn über mehrere k summiert werden soll. Die Gauss-Formel geht dann über in das Integral

$$\sum_{k=k_1}^{k=k_2} P(k) \approx \int_{x_1}^{x^2} \varphi(t)dt$$

mit $x1 = \frac{k1-0,5-\bar{x}}{\sigma}$

und $x2 = \frac{k2-0,5-\bar{x}}{\sigma}$

Das Integral von Hand zu berechnen, ist ein fast hoffnungsloses Unterfangen. Auch die numerische Integration mit dem Computer führt in der Regel zu keinem guten Ergebnis (das liegt an der Exponentialfunktion, die sehr schlecht konvergiert). Deshalb behelfen wir uns mit einer Näherungsformel, die zwar nicht ganz einsichtig erscheinen wird, aber trotzdem ganz gute Ergebnisse liefert. Das entsprechende Programm zeigt das Listing 3.

Ein Beispiel zur Illustration: Jemand behauptet, dass es in der Bevölkerung 20 % Brillenträger gibt. Wenn diese Behauptung zutrifft, wie gross ist dann die WS, dass die Zahl der Brillenträger zwischen 170 und 230 liegt (1'000 Personen werden dazu willkürlich ausgewählt)?

Es ist also k1 = 170 und damit

$$x1 = \frac{170-0,5-200}{\sqrt{160}} \approx -2,41$$

Mit analoger Rechnung erhält man x2 = +2.41

So ergibt sich P(170<=k<=230) = G(2.41)-G(-2.41) = 0.9840

Ein ähnliches Ergebnis bringt auch unsere Näherungsformel im Listing. Man erhält hier 0.999.

Die Formel im Listing sollte mit Vorsicht behandelt werden. Wenn eine WS grösser als 1 herauskommt, ist das Ergebnis natürlich unbrauchbar. WS im Bereich zwischen 0.05 und 0.95 kommen der Realität aber meist sehr nahe.

Wie man das Integral in der Gauss-Formel explizit behandelt (und was ein Integral überhaupt ist) erfahren Sie in einer der nächsten Ausgaben. □

Computer-Splitter

Intelligentes Computerzubehör

Die Firma Wiesmann + Theis, in der Schweiz vertreten durch die Firma Weber + Co., zählt zu den führenden Herstellern von intelligentem Computerzubehör. Computerzubehör wie Buffer, Umschalter, Leitungstreiber usw., soll in der Anwendung einfach und problemlos sein. Die meisten Geräte werden komplett mit allen notwendigen Verbindungen, Netzteilen und einer 3-sprachigen (D,F,E) Bedienungsanleitung geliefert. Weitere Informationen finden Sie im neu erschienenen kostenlosen Gesamtkatalog. Neben den Produktebeschreibungen mit Anwendungsbeispielen, umfasst der Katalog einen technischen Teil mit Anschlussbildern, Schnittstellenbeschreibungen, Codetabellen und Fehlersuchplänen für die wichtigsten Schnittstellen. Info: Weber + Co. Mühlestrasse 52, 8623 Wetzikon 3, Tel. 01/930'20'03. □

Von Prozessen, Tasks und Coroutinen (2. Teil)

Der erste Teil dieser Artikelserie (M+K 87-5) war der Einführung und der Begriffsbestimmung gewidmet. In diesem abschliessenden Teil wenden wir uns nun den Koordinierungskonzepten zu. (Red. Wie Sie sicher bemerkt haben, sind irrtümlich bereits im 1. Teil die Abbildungen 5a bis 10 abgedruckt worden. Zum besseren Verständnis werden wir aber diese Abbildungen in diesem 2. Teil nochmals bringen).

Koordinationskonzepte

Schliesst man Time-Sharing-Systeme und ähnliches aus, so erfolgt die Prozesskoordination über Objekte (Prozeduren, Anweisungen), die das jeweilige Konzept zu Verfügung stellt. Ein Beispiel:

SEND meldung TO nachbar_prozess

Diese Objekte spezifizieren nicht die Art und Weise, wie Prozesse umzuschalten sind (vgl. mit Coroutinen), sondern drücken eine Funktion aus, in unserem Beispiel das Senden einer Meldung zu einem Nachbarprozess. Der Anwender muss dabei gar nicht wissen, dass diese Funktion koordinierende Massnahmen des Schedulers zur Folge hat. Für ihn ist es lediglich eine Anweisungen, die er genau gleich wie konventionelle Anweisungen brauchen kann. Er wird damit aus der Verantwortung für das Prozess-Management entlassen und kann sich voll seiner eigentlichen Arbeit widmen. Zudem garantiert die Abstraktheit dieser Objekte, dass in der Prozessdefinition keine spezifische Implementierungsform präjudiziert wird. Das einzige Zugeständnis bezüglich Parallelverarbeitung ist die Verpflichtung, diese Objekte auch zu gebrauchen. Mit anderen Worten: Will er etwa einen Systemdrucker benutzen, so hat er die entsprechenden Anweisungen oder Prozeduren zu verwenden und darf keinesfalls direkt auf den Drucker zugreifen, da sonst das gesamte System korrumpiert würde!

Andreas Pichler

Wir verfügen nun über genügend Rüstzeug, um uns endlich mit den einzelnen Koordinierungskonzepten befassen zu können. Beginnen wir mit dem ältesten hier vorgestellen Konzept: dem Semaphorenkonzept.

Semaphoren

Das Semaphorenkonzept wurde 1968 von E.W. Dijkstra in seiner inzwischen schon klassisch gewordenen Arbeit [1] eingeführt. Der ursprüngliche Ausdruck Semaphor stammt aus der Eisenbahnsignalisierung. Ein Semaphor im Sinne von Dijkstra ist dagegen nichts weiter als ein Flag, auf das nur mittels zweier privilegierter Prozeduren P(s) und V(s) zugegriffen werden kann. P(s) wird bei Eintritt in den kritischen Abschnitt aufgerufen, V(s) bei Verlassen. Da die beiden Prozeduren als eine Art Codeklammerung angesehen werden können, gehört das Semaphorenkonzept zur Kategorie des sogenannten «code locking».

Betrachten wir uns die Implementierung dieser beiden privilegierten Prozeduren:

```
P(s) : BEGIN
        WHILE (s = 0) DO (* Siesta *) END;
        s := s - 1
       END;

V(s) : BEGIN
        s := s + 1
       END;
```

Bemerkung:

1) Die Abkürzungen für die Prozeduren stammen aus dem niederländischen und bedeuten:

   - P: Proberen = Probieren
   - V: Verhogen = Erhöhen

2) Zu Beginn der Ausführung muss das Semaphor s initialisiert werden. Der Initalwert ergibt sich dabei aus der maximalen Anzahl von Prozessen, die den kritischen Abschnitt gleichzeitig benützen dürfen.

Kehren wir zurück zu unserer Andenbahn. P(s) entspricht unseren (unteilbaren) Operationen:

- Schale prüfen, und wenn leer (d.h. s <> 0 !)
- Stein suchen und in Schale legen

und V(s) der Operation:

- Stein aus Schale entfernen.

Damit sieht ein typisches Zugführerleben etwa wie folgt aus:

```
LOOP (* Zugführerleben *)
 P(s)
 "Fahre durch kritischen Abschnitt"
 V(s)
 "Fahre restliche Strecke"
END (* Zugführerleben *)
```

Da nur jeweils ein Zug den kritischen Abschnitt gleichzeitig passieren darf, muss das Semaphor s zu Beginn der Ausführung auf den Wert 1 initialisiert worden sein. Nun mag man es ja einem Zugführer durchaus gönnen, dass er von Zeit zu Zeit Siesta halten kann, aber in einem Prozesssystem ist die Konstruktion

```
WHILE (s = 0) DO (* Siesta *) END
```

sehr ineffizient, da sie den Prozessor zu 100% mit Nichtstun (Idle loop) belegt. Es ist daher sinnvoll, wenn wir uns wieder an die nun ja bereits wohlbekannten Warteschlangen erinnern und wartende Prozesse suspendieren, anstatt sie in nutzlosen Warteschleifen kreiseln zu lassen. Wir erhalten dann die folgende Formulierung für die beiden Prozeduren:

```
P(s) : BEGIN
        IF (s = 0) THEN "in Warteschlange einreihen" END
        s := s - 1
       END

V(s) : BEGIN
        s := s + 1
        "nächsten Prozess in Warteschlange aktivieren"
       END
```

Erlaubt der kritische Abschitt nur die gleichzeitige Benutzung durch einen einzigen Prozess, so kann eine Sonderform, nämlich das sogenannte binäre Semaphor, verwendet werden:

- allgemeines Semaphor

s ∈ CARDINAL [0..max. mögl. gleichzeitige Benutzung]

- binäres Semaphor

s ∈ {0,1} bzw.
s ∈ {FALSE,TRUE} bzw.
s ∈ {unbenutzt,benutzt} bzw.
s ∈ {free,used} usw.

Abb.5 zeigt einen vollständigen Bibliotheksmodul für binäre Semaphoren. Für die Prozesszustandsumschaltung sorgen Objekte aus dem Modul Processes (Anmerkung: In der diesem Modul zugrunde liegenden Originalversion aus [1] wurde die Frage nach der Art und Weise der Prozesszustandsumschaltung offengelassen, da sie für die Erklärung der Funktionsweise der Semaphoroperationen irrelevant ist!). Sinnvollerweise wurde die zum Semaphor

```
DEFINITION MODULE BinSema;

 EXPORT QUALIFIED SEMAPHORE,InitSemaphore,P,V;

 TYPE  SEMAPHORE;

 PROCEDURE  InitSemaphore(VAR s : SEMAPHORE);
   (* Initialisiere Semaphor s *)

 PROCEDURE  P(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Proberen" *)

 PROCEDURE  V(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Verhogen" *)

END BinSema.
```

Abb. 5a: Definitions-Modul BinSema

```
IMPLEMENTATION MODULE BinSema [7];

 FROM SYSTEM    IMPORT TSIZE;
 FROM Storage   IMPORT ALLOCATE;
 FROM Processes IMPORT SIGNAL,WAIT,SEND,Init;

 TYPE  SEMAPHORE = POINTER TO SemaphoreDescriptor;
       SemaphoreDescriptor = RECORD
                               CriticalSection : (free,used);
                               BecomeFree      : SIGNAL
                             END;

 PROCEDURE  InitSemaphore(VAR s : SEMAPHORE);
   (* Initialisiere Semaphor s *)
  BEGIN
   ALLOCATE(s,TSIZE(SemaphoreDescriptor));
   WITH s^ DO
    CriticalSection := free;
    Init(BecomeFree)
   END
  END InitSemaphore;

 PROCEDURE  P(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Proberen" *)
  BEGIN
   WITH s^ DO
    IF CriticalSection = used THEN WAIT(BecomeFree) END;
    CriticalSection := used
   END
  END P;

 PROCEDURE  V(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Verhogen" *)
  BEGIN
   WITH s^ DO
    CriticalSection := free;
    SEND(BecomeFree)
   END
  END V;

END BinSema.
```

Abb. 5b: Implementations-Modul Bin-Sema

gehörende Warteschlange in die Semaphor-Definition miteingeschlossen. Die Prozedur InitSemaphore() sorgt für die Bereitstellung von Speicher und die korrekte Initialisierung des Semaphores. Desweiteren wird darin auch die Warteschlange für die Suspendierung (Ereignis: BecomeFree) initialisiert. Die Ready-Queue wird bereits beim Import des Modul Processes aufgebaut. Das Semaphorenkonzept stellt das meiner Meinung nach subtilste Werkzeug für die Prozesskoordination dar. Sämtliche Koordinierungsgründe, also Synchronisierung, Kommunikation und gegenseitiger Ausschluss, lassen sich damit sehr elegant und sehr spezifisch programmieren. Leider erweist es sich im täglichen Gebrauch aber als nahezu unbrauchbar, da sich die im ganzen Code verstreuten P's und V's bei schon vergleichsweise geringer Komplexität des Prozesssystemes kaum mehr überblicken lassen. Auch ist es dem Compiler nicht möglich, die paarweise Verwendung von P(s) und V(s) zu überprüfen, da damit ein Grossteil von Anwendungsmöglichkeiten (asymetrische Zustandsumschaltung) ausgeschlossen würde. Die Problematik bei der Verwendung von Semaphoren weist grosse Aehnlichkeit mit der GOTO-Problematik auf: was auf den ersten Blick als subtil erscheint, erweist sich auf den zweiten als zu subtil, zu freizügig, zu unkontrollierbar, mit einem Wort: unbrauchbar. Aus diesem Grund vermochten sich Sprachen mit integriertem Semaphorenkonzept im allgemeinen auch nicht durchzusetzen.

Monitore

Die Monitortheorie basiert auf den Arbeiten von P. Brinch-Hansen und wurde 1974 von C.A.R. Hoare [2] zu ihrer heutigen Form ausgearbeitet. Ein Monitor ist eine Einheit, welche Datenstrukturen und die dazugehörigen Zugriffsprozeduren verkapselt:

```
MODULE Monitor [7]

 (* Deklaration der Datenstrukturen *)

 (* Deklaration der Zugriffsprozeduren *)

BEGIN

 (* Monitor-Initialisierung *)

END Monitor
```

Der Zugriff auf die Monitor-internen Datenstrukturen erfolgt ausschliesslich via Aufruf der im Monitor deklarierten Zugriffsprozeduren. Im Gegensatz zum Semaphorkonzept, wo die beiden Prozeduren P() und V() einen Codeteil schützen, schützt der Monitor eine Datenstruktur. Man nennt diese Schutzeinrichtung deshalb «data locking». Der gesamte Monitor befindet sich auf privilegiertem Niveau, d.h. nur jeweils ein Prozess kann auf den Monitor zugreifen. Offensichtlich haben wir damit ein Instrument für die Handhabung des gegenseitigen Ausschlusses.

Da die den Monitor benutzenden Prozesse nur die Zugriffsprozeduren - nicht aber die eigentliche Datenstruktur - sehen, obliegt dem Monitor bzw. den Zugriffsprozeduren die Verwaltung der Datenstruktur. Betrachten wir dazu einen Monitor mit einem Buffer und den beiden Zugriffsprozeduren append() für das Ablegen von Objekten im Buffer und remove() für das Abholen von Objekten aus dem Buffer. Selbstverständlich kann ein append() aufrufender Produktionsprozess nur dann sein Objekt im Buffer hinterlegen, wenn darin noch genügend Platz vorhanden ist. Ist das nicht der Fall, so muss append() dafür sorgen, dass der Prozess wartet, bis wieder genügend Platz vorhanden ist. Eine ähnliche Situation findet man im umgekehrten

Fall vor, wo ein remove() aufrufender Konsumationsprozess möglicherweise warten muss, bis im Buffer wieder ein Objekt zur Verfügung steht, das er abholen kann. Nun wissen wir natürlich mittlerweilen, dass man Prozesse nicht einfach in einer Warteschleife kreiseln lässt, sondern dass man sie solange suspendiert, bis der jeweilige Blockierungsgrund hinfällig geworden ist.

Der Monitor buffer im folgenden Beispiel verwaltet einen Buffer, der N Objekte aufnehmen kann. Eine Monitor-lokale Variable n beinhaltet den Füllgrad des Buffers. Es gilt:

n = 0 : Buffer ist leer
n = N: Buffer ist voll

Damit ergibt sich folgende mögliche Monitor-Implementierung, wobei PQ die Warteschlange bezeichnet, in der Produzenten eingereiht werden, wenn der Buffer voll ist und KQ die Warteschlange für Konsumenten, die auf ihr Konsumationsobjekt warten:

```
MONITOR buffer

 (* Deklaration des Buffers sowie lokaler Datenobjekte *)

 PROCEDURE append ()

  BEGIN
   IF n = N THEN "Produzent in PQ einreihen" END
   "Buffer-Operation"
   "Höchst priorisierten Konsument in KQ deblockieren"
  END

 PROCEDURE remove ()

  BEGIN
   IF n = 0 THEN "Konsument in KQ einreihen" END
   "Buffer-Operation"
   "Höchst priorisierten Produzent in PQ deblockieren"
  END

BEGIN (* Monitor-Init *)
 n := 0
 "Initialisiere PQ und KQ"
 "Initialisiere Monitor-lokale Variablen"
END
```

Der geneigte Leser wird sich spätestens jetzt wieder an die Semaphoren erinnert fühlen. Noch offensichtlicher dürfte die Verwandtschaft mit der (schematisierten) Wiedergabe des Monitors «single resource» aus [2] werden:

```
MODULE SingleResource [7]

 VAR busy       : BOOLEAN (* single resource *)
     BecomeFree : SIGNAL

 PROCEDURE append()

 BEGIN
  IF busy THEN WAIT(BecomeFree) END
  busy := TRUE
 END append

 PROCEDURE remove()

 BEGIN
  busy := FALSE
  SEND(BecomeFree)
 END remove

BEGIN
 busy := FALSE
 Init(BecomeFree)
END SingleResource
```

Schlüsselbegriffe

MODULA-2, Prozesse, Tasks, Coroutinen, Semaphoren, Monitore, Message Passing, Rendez-Vous, Interrupts, Prioritäten, Konflikte, Deadlock, Synchronisation, Kommunikation, gegenseitiger Ausschluss.

Durch die Reduzierung der Datenstruktur auf eine boolesche Variable vereinfachen sich die Bufferoperationen zu simplen Zuweisungen. Auch entfällt in remove() die Abfrage nach dem Zustand der Variablen, da diese bei folgerichtiger Anwendung (Produzent kommt vor Konsument) ja nur TRUE sein kann. Doch auch im umgekehrten Fall (Konsument vor Produzent) ist die Abfrage unnötig, da sich nach dem ersten Aufruf die richtige Reihenfolge von alleine einstellt und die Zuweisung nur einer neuerlichen Initialisierung gleichkommt.

Vergleichen Sie nun diesen Monitor mit Abb. 5. Offensichtlich stellt SingleResource nichts weiter als einen binären Semaphor dar, wobei append() für P() und remove() für V() steht. Natürlich verfolgt man mit einem Monitor normalerweise ein anderes Ziel, nämlich den Schutz einer (ausgedehnten) Datenstruktur. Das Beispiel soll lediglich zeigen, dass man auch mit reinen Monitorsprachen wie etwa PORTAL ein Semaphorenkonzept realisieren kann.

Im folgenden werden wir uns eine typische Monitoranwendung etwas eingehender betrachten. Die Beschreibung folgt dabei im wesentlichen [6], da sie verständlicher als die Originalversion in [2] ist und zudem auch keinen Fehler enthält (vgl. [2], S.553, Bsp. Bounded Buffer!). Der Monitor BoundedBuffer schützt einen zyklischen Buffer (Ringbuffer) mit Namen buffer. Er kann N Objekte vom Typ portion aufnehmen. Die lokale Variable n bezeichnet wiederum den jeweiligen Füllgrad des Buffers und die beiden Zeiger in und out die Stelle innerhalb des Buffers, wo der Produktionsprozess seine Objekte hinterlegen und von wo der Konsumationsprozess seine Objekte abholen kann (in [2] wird nur ein Zeiger lastpointer benutzt und n heisst dort count). Sie rufen dazu die schon bekannten Prozeduren append() und remove() auf. Ist der Buffer für die beabsichtigte Operation nicht bereit, werden die aufrufenden Prozesse in Warteschlangen eingereiht. Dabei gilt:

- Ereignis: nonfull für Produktionsprozesse, wenn der Buffer voll ist.
- Ereignis: nonempty für Konsumationsprozesse, wenn der Buffer leer ist.

In Modula-2 formuliert erhalten wir für die beiden Prozeduren folgende mögliche Implementierung:

```
PROCEDURE  append (x : portion);

 BEGIN
  IF n = N THEN WAIT(nonfull) END;
  INC(n);
  buffer[in] := x;
  in := (in + 1) MOD N;
  SEND(nonempty)
 END append;


PROCEDURE  remove (VAR x : portion)

 BEGIN
  IF n = 0 THEN WAIT(nonempty) END;
  DEC(n);
  x := buffer[out];
  out := (out + 1) MOD N;
  SEND(nonfull)
 END remove;
```

Diese Prozeduren funktionieren zwar einwandfrei, haben aber den Nachteil, dass bei jeder Bufferoperation ein Signal gesendet wird. Effizienter wäre hingegen, wenn nur jeweils dann ein Signal gesendet würde (und damit eine relativ zeitintensive Prozesszustandsumschaltung auslöst), wenn auch wirklich ein anderer Prozess auf dieses Signal wartet. Eine Verbesserung der Situation erhält man, indem man den Wertebereich der Variablen n ausdehnt:

bisher: $n = 0$ - Buffer leer
$n = N$ - Buffer voll

neu: $n < 0$ - Buffer leer, n Konsumenten warten
$n \in [0..N]$ - n Objekte in Buffer, kein Prozess wartet
$N < n$ - Buffer voll, n - N Produzenten warten

Unter Berücksichtigung dieses Wertebereiches können wir die beiden Prozeduren neu folgendermassen formulieren:

```
PROCEDURE  append (x : portion);

 BEGIN
  INC(n);
  IF n > N THEN WAIT(nonfull) END;
  buffer[in] := x;
  in := (in + 1) MOD N;
  IF n <= 0 THEN SEND(nonempty) END
 END append;


PROCEDURE  remove (VAR x : portion)

 BEGIN
  DEC(n);
  IF n < 0 THEN WAIT(nonempty) END;
  x := buffer[out];
  out := (out + 1) MOD N;
  IF n >= N THEN SEND(nonfull) END
 END remove;
```

In Abb. 6 finden Sie einen vollständig auscodierten Monitor in Form eines lokalen Modules, der für die beiden Prozeduren die oben gezeigte Implementierungsform verwendet.

Wie bereits erwähnt, ist das Monitorkonzept in erster Linie ein Instrument für die Handhabung des gegenseitigen Ausschlusses. Das Beispiel des Monitors SingleResource zeigt aber auch, dass das Konzept flexibel genug für andere Anwendungen (Implementierung von Semaphoren) ist.

Der grösste Vorteil des Konzeptes ist dessen Blockstruktur. Kernstück eines Monitors ist ja der Datensatz; die Zugriffsprozeduren sind nur Mittel zum Zweck. Obschon Aufrufe dieser Zugriffsprozeduren - gleich wie die Semaphoroperationen - im ganzen Code verstreut sind, konzentriert sich doch die eigentliche Interaktion auf einen einzigen, wohldefinierten Ort: dem Monitor; bei Semaphoren wird

```
MODULE BoundedBuffer [7];

 EXPORT append,remove;
 IMPORT SIGNAL,SEND,WAIT,Init,portion;

 CONST  N = AnyValue;

 VAR    buffer   : ARRAY [0..N-1] OF portion;
        n        : [0..N];
        in,out   : [0..N-1];
        nonfull  : SIGNAL;
        nonempty : SIGNAL;

 PROCEDURE append (x : portion);

  BEGIN
   INC(n); IF n > N THEN WAIT(nonfull) END;
   buffer[in] := x; in := (in + 1) MOD N;
   IF n <= 0 THEN SEND(nonempty) END
  END append;


 PROCEDURE remove (VAR x : portion);

  BEGIN
   DEC(n); IF n < N THEN WAIT(nonempty) END;
   x := buffer[out]; out := (out + 1) MOD N;
   IF n >= N THEN SEND(nonfull) END
  END remove;


BEGIN (* init module *)
 n := 0; in := 0; out := 0;
 Init(nonfull); Init(nonempty);
END BoundedBuffer;
```

Abb. 6: Lokaler Monitor-Modul BoundedBuffer

dieser Ort durch das Konzept nicht bestimmt. Zwar schränkt eine derartige Zentralisierung den Anwendungsbereich sowohl logisch (Software) als auch maschinell (Rechnerarchitekturen) ein, doch gewonnen wird der nicht zu unterschätzende Vorteil der automatischen Ueberprüfung des korrekten Gebrauches durch den Compiler. Desweitern sind Fehlerquellen viel leichter aufzufinden. So braucht man etwa beim Testen mit Debuggern oder In-Circuit Emulatoren nur den Monitor, nicht aber den gesamten Code (Semaphoren) zu verfolgen!

Grösster Nachteil des Monitors ist die vom Konzept her bedingte Sequentialisierung von Prozessen, die in dieser Strenge nicht immer notwendig ist. So wäre es beispielsweise unnötig, zwei lesenden Prozessen den gleichzeitigen Zugriff auf den Datensatz zu verwehren. Bei falscher oder ungenügender Modellierung kann es sogar vorkommen, dass ein Prozesssystem durch die verwendeten Monitore zu einem sequentiellen Programm degeneriert. Selbstverständlich ist man solchen Gefahren auch bei der Verwendung von Semaphoren ausgesetzt. Allerdings erlaubt das flexiblere Semaphorenkonzept schon beim Erkennen möglicher Flaschenhälse (Reduktion des Parallelitätsgrades durch erzwungene Sequentialisierung) deren Umgehung durch geeignete Programmierung (Unterscheidung lesend/schreibend, gleichzeitiger Zugriff usw.). Bei Monitoren sind die Programm-technischen Möglichkeiten selbst dann eingeschränkt, wenn die Flaschenhälse schon zum Voraus erkannt wurden. Dennoch existieren heute mehr Sprachen, die das Monitorkonzept in ihrem Sprachumfang integriert haben, als solche, die das Semaphorenkonzept verwenden.

Ein weiteres Indiz dafür, dass zu subtile Konzepte gar nicht gefragt sind, da die damit verbundenen Schwierigkeiten die Vorteile solcher Konzepte in den weitaus meisten Fällen mehr als nur aufheben.

Message Passing

Die Betrachtungen zum Message Passing erfordern eine einführende Erläuterung, wie die Begriffe lose und eng gekoppelt im vorliegenden Kontext zu verstehen sind:

- Lose gekoppelte Prozesse verkehren miteinander über statisch vorhandene Datensätze (Semaphoren, Monitore). Kennzeichnend für lose Kopplung ist, dass die Prozesse beim Informationsaustausch nicht unbedingt synchronisiert werden müssen.
- Eng gekoppelte Prozesse verkehren miteinander über dynamische Datenlinks, die eigens für die Interaktion aufgebaut werden. Das nur zeitlich begrenzte Vorhandensein eines solchen (logischen) Datenlinks impliziert eine Synchronisierung der beteiligten Prozesse. Im allgemeinen wird ein Datenlink nur zwischen zwei Prozessen errichtet; die Interaktion bleibt damit auf zwei Prozesse beschränkt und ein gleichzeitiger Zugriff mehrerer Prozesse auf kritische Abschnitte ausgeschlossen.

Bitte beachten Sie, dass diese Erläuterung einen gewissen Blickwinkel voraussetzt. Eine Definition in dieser Form deckt nicht alle Aspekte der Kopplung ab, genügt aber für die weiteren Betrachtungen.

Den Austausch von Informationen über einen dynamischen Datenlink nennt man Message Passing. In der Regel wird dabei pro Interaktion eine Meldung von einem Senderprozess zu einem Empfängerprozess übertragen. Beide Prozesse müssen für den Austausch bereit sein, was eine vorhergehende Synchronisierung bedingt. Abb. 7 zeigt

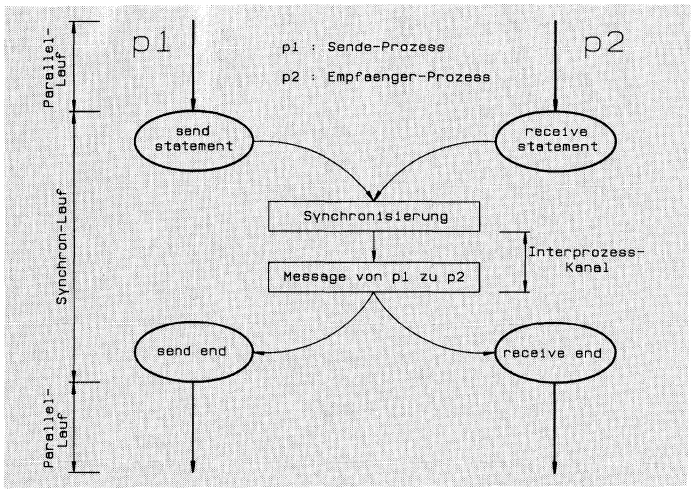


Abb. 7: Message Passing

grafisch, wie man sich eine solche Interaktion vorzustellen hat. In einer fiktiven Sprache formuliert, könnten die beiden Prozesse etwa wie folgt aussehen:

```
Senderprozess: producer()

  LOOP
   "Generiere Meldung message"
   SEND message TO consumer
   "Restliche Arbeit ausführen"
  END

Empfängerprozess : consumer()

  LOOP
   RECEIVE message FROM producer
   "Verarbeite Meldung message"
   "Restliche Arbeit ausführen"
  END
```

Verwendet man statt des dynamischen Datenlinks eine funktionsgleiche statische Struktur als Link zwischen den Prozessen, gelangt man zum sogenannten Mailbox-Konzept. Im Gegensatz zum dynamischen Datenlink, der eine Synchronisierung erfordert, erreicht man mittels Mailboxen eine völlige Entkopplung der Prozesse. Eine Zustandsumschaltung ist nur dann erforderlich, wenn der Sender in eine volle Mailbox schreiben oder der Empfänger aus einer leeren Mailbox lesen will. Da die Mailbox erstens statischer Natur ist und zweitens in der Regel von mehreren Prozessen benutzt wird, koppelt sie Prozesse auf unsere Erklärung bezogen lose. Abb. 8 zeigt grafisch einen Informationsaustausch via Mailbox und der folgende Programmausschnitt fragmentarisch und schematisch eine mögliche Formulierung:

```
VAR mbx : MAILBOX

Senderprozess: producer()

  LOOP
   "Generiere Meldung message"
   SEND message TO mbx
   "Restliche Arbeit ausführen"
  END

Empfängerprozess : consumer()

  LOOP
   RECEIVE message FROM mbx
   "Verarbeite Meldung message"
   "Restliche Arbeit ausführen"
  END
```

Natürlich ist eine Mailbox nichts weiter als ein versteckt betriebener Monitor. Unterschiedlich ist einzig, dass Meldungen nur einmal gelesen werden können und dass die Verwaltung der Mailbox, meist als FIFO organisiert, nicht unter der Kontrolle des Anwenders steht, sondern fest von der Implementierung vorgegeben wird. Diese zwei Gründe lassen es als sinnvoll erscheinen, die beiden Konzepte, trotz Verwandtschaft, als ebenbürtige Koordinierungsmittel zu führen. So kennt beispielsweise CHILL [5,10,11] die Konzepte REGION, im wesentlichen ein Monitor, der allerdings nur die Deklarationen enthält, aber nicht initialisiert werden kann, und den Datentyp BUFFER sowie die Anweisungen (das eigentliche Message Passing wird in CHILL SIGNAL-Konzept genannt).

Das Message Passing ist in erster Linie ein Konzept für die Kommunikation, im Falle des eigentlichen Message Passing mit impliziter, im Falle von Mailboxen mit möglicher Synchronisierung. Nachteilig an dem Konzept ist, dass Meldungen nur einmal gelesen werden können und damit mögliche Anwendungen ausschliessen oder zumindest sehr erschweren. So braucht es beispielsweise regelrechte Zubringerprozesse, um auf gemeinsame globale Datenstrukturen zuzugreifen. Die dabei oftmals notwendigen Prozesszustandsumschaltungen verschlechtern die Leistungsfähigkeit des Prozesssystemes, insbesondere dann, wenn es auf Einzelrechnersystemen abläuft. Nun wurde das Konzept natürlich nicht für Einzelrechnerarchitekturen entwickelt. Message Passing eignet sich besonders für Interaktionen über Bussysteme, wo in der Regel nur Steuerinformationen zwischen den einzelnen Prozessorsystemen ausgetauscht werden müssen. Vorteilhaft wirkt sich dabei auch noch die Tatsache aus, dass sehr viele Bussysteme das Message Passing mit ihrer Hardware unterstützen. Es fördert damit besser als jedes andere Koordinierungsmittel auch auf logischer Ebene dezentrale Rechnerarchitekturen.

Rendez-Vous

Das letzte Koordinierungsmittel, das wir uns betrachten werden, ist sprachspezifisch. Lediglich ADA, die im Auftrag des amerikanischen Verteidigungsministeriums DoD entwickelte Sprache, kennt dieses Konzept [3].

Bei einem Rendez-Vous fordert ein Klient-Prozess mittels eines sogenannten ENTRY-Calls eine Dienstleistung eines Bediener-Prozesses an. Der Bediener-Prozess erwartet in einer ACCEPT-Anweisung den Aufruf des Klienten-Prozesses und führt nach erfolgter Synchronisation die im ACCEPT-Statement codierte Dienstleistung aus. Am Ein- und Ausgang des Rendez-Vous können dabei Parameter via Message Passing übertragen werden. Abb. 9 zeigt den Versuch, ein derartiges Rendez-Vous grafisch zu verdeutlichen.

Bei den Dienstleistungen des Bediener-Prozeses handelt es sich natürlich um die bereits wohlbekannten kritischen Aktionen. Betrachten wir uns dazu das folgende

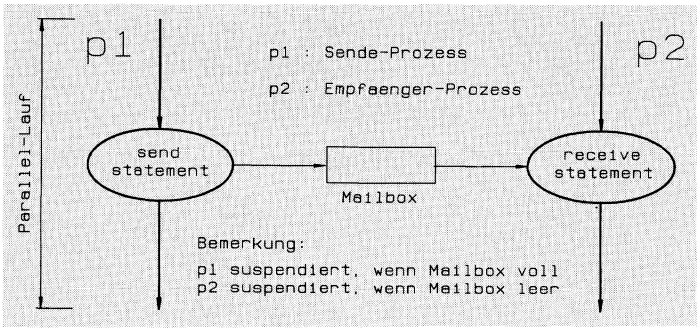


Abb. 8: Mailbox-Konzept

Programmfragment mit einem Editor als Klient-Prozess und einem Drucker-Driver als Dienstleistungs-Prozess, wobei hier für einmal nicht ein Datensatz, sondern ein Drucker das zu koordinierende Objekt ist:

```
Klient-Prozess : editor()

 LOOP
  "Dokument doc.txt editieren und abspeichern"
  ENTRY printer_driver.print (OUT doc.txt; IN confirmation)
  IF confirmation <> okay THEN "Sonderbehandlung" END
 END

Bediener-Prozess : printer_driver()

 LOOP
  ACCEPT print (IN text_file; OUT confirmation) DO
   "Dokument text_file prüfen"
   "confirmation gemäss Resultat der Prüfung aufsetzen"
  END ACCEPT
  IF okay THEN "Dokument text_file ausdrucken" END
 END
```

Der Editor-Prozess bekundet mit dem ENTRY-Call seine Absicht zu einem Rendez-Vous mit dem Printer-Driver-Prozess, um diesem das fertiggestellte Dokument zum Ausdrucken zu übergeben. Der Printer-Driver-Prozess seinerseits erwartet im ACCEPT-Statement auf neue Aufträge. Sind beide Prozesse zum Rendez-Vous bereit, d.h der Editor führt einen ENTRY-Call aus und der Printer-Driver wartet bereits im ACCEPT-Statement, kommt es zur Synchronisation. Der Editor übermittelt dem Printer-Driver das Dokument (meistens in Form der Bufferadresse oder des Filenamens), welches von diesem nach verschiedenen Kriterien überprüft wird. Am Schluss der Ueberprüfung übermittelt er das Ergebnis an den Editor und druckt, sofern alles in Ordnung ist, das Dokument aus. Der Editor erkennt am Inhalt der Rückmeldung, ob das Dokument vom Printer-Driver akzeptiert worden ist oder ob es noch einer nachträgliche Korrektur unterzogen werden muss.

Die ACCEPT-Anweisung lässt sich in gewisser Weise mit den Semaphor-Operationen P() und V() vergleichen, die ja ebenfalls einen kritischen Abschnitt einschliessen. Aus Abb. 10 ist ersichtlich, dass sich mit dem Rendez-Vous-Konzept sogar ein Monitor realisieren lässt. Dennoch unterscheidet es sich doch recht erheblich von diesen beiden. Der Unterschied resultiert hauptsächlich aus der Tatsache, dass beim Rendez-Vous-Konzept der kritische Abschnitt in Prozesse verpackt wird. Dies bedingt, dass man in ADA eine spezielle Klasse von Prozessen benötigt, welche die Benutzung des jeweiligen Mediums oder den Zugriff auf die entsprechenden Datensätze gestatten. Mit anderen Worten: Was bei den Semaphoren und Monitoren mit einem simplen Prozeduraufruf erreicht wird, erfordert im Rendez-Vous-Konzept einen regelrechten Dienstleistungs- oder Zubringerprozess, der kritische Aktionen in seiner ACCEPT-Anweisung schützt. Die folgenden Programmfragmente sollen dies verdeutlichen:

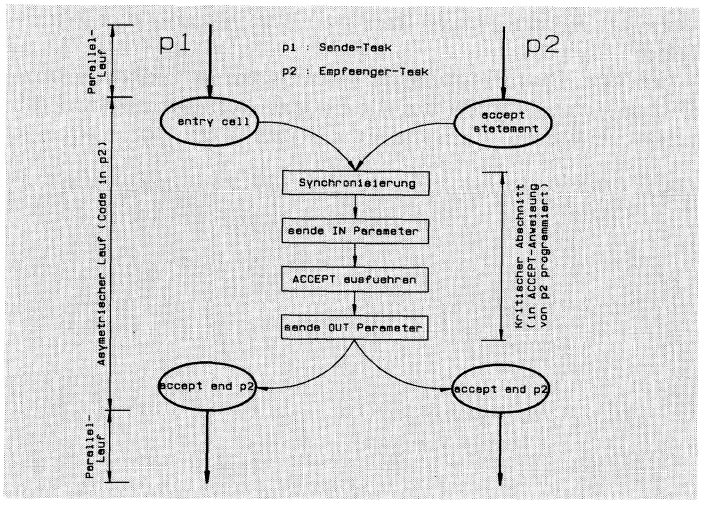


Abb. 9: Rendez-vous-Konzept

```
task body BoundedBuffer is

   N      : constant integer := AnyValue;
   Buffer : array (0..N-1) of Portion;
   In     : integer := 0;
   Out    : integer := 0;
   n      : integer := 0;

   begin
    loop
     select
      when n < N =>
       accept Append (x : in Portion) do
        Buffer(In) := x;
        n := n + 1;
        In := (In + 1) mod N;
       end;
     or
      when n > 0 =>
       accept Remove (x : out Portion) do
        x := Buffer(Out);
        n := n - 1;
        Out := (Out + 1) mod N;
       end;
     or
      terminate;
     end select;
    end loop;
   end BoundedBuffer;
```

Abb. 10: Task-Fragment BoundedBuffer

```
VAR GemeinsameDaten : BUFFER

PROCEDURE Zugriff (* Zugriffsprozedur für GemeinsameDaten *)

 Es gilt :
  - a) Buffer und Zugriff global deklariert
  - b) Buffer und Zugriff lokal in Monitor M deklariert
  - c) Buffer und Zugriff lokal in Prozess pz deklariert

 a) Semaphor-Konzept:

    p1: LOOP                    p2: LOOP
          |                           |
         P(s)                        P(s)
          Zugriff                     Zugriff
         V(s)                        V(s)
          |                           |
        END                         END


 b) Monitor-Konzept

    p1: LOOP                    p2: LOOP
          |                           |
         M.Zugriff                   M.Zugriff
          |                           |
        END                         END


 c) Rendez-Vous-Konzept

    p1: LOOP              pz: LOOP              p2: LOOP
          |                     |                     |
         ENTRY pz.get()        ACCEPT get()          ENTRY pz.get()
          |                     Zugriff               |
          |                    END ACCEPT             |
          |                     |                     |
        END                   END                   END
```

Das Rendez-Vous-Konzept in ADA stellt einen Kompromiss dar. Es eignet sich speziell für dezentralisierte Rechnerarchitekturen, wobei es das reine Message Passing einschliesst und für diese Art Architektur vorteilhaft erweitert. Ansonsten, zumal auf einem Einprozessorsystem, sind die gleichen Schwachstellen wie beim Message Passing zu bemängeln, also oftmals unnötige Synchronisierung und Zeiteinbussen beim Zugriff auf globale Datensätze oder Medien durch die Verkapslung in aktive Elemente wie Dienstleistungsprozesse. Allerdings wirken sich solche Dienstleistungsprozesse positiv auf die Les-, Test- und Wartbarkeit aus, da Prozessinteraktionen auf wohldefinierte Orte konzentriert werden. Im Uebrigen gestattet das Konzept eine weitgehende Prüfung der Anwendung durch den Compiler.

Zusammengefasst darf gesagt werden, dass das Rendez-Vous-Konzept alle Koordinierungsgründe unter Einschluss grösstmöglicher Compiler-Ueberprüfung der Anwendung beherrscht. Unbestritten hat es damit Vorteile gegenüber dem Semaphorenkonzept (fehlende Ueberprüfungsmöglichkeiten), dem Monitorkonzept (beschränktes Anwendungsfeld) und dem Message Passing (Subset des Rendez-Vous-Konzeptes). Nachteilig ist, dass nicht alle Anwendungen derart effizient abgedeckt werden können, wie das mit dem dazu besten Konzept zu erreichen wäre. Doch auch dies ist nicht unbedingt ein Nachteil, wenn man bedenkt, dass eine Sprache wie CHILL zwar praktisch alle Konzepte integriert hat und deshalb noch von ein paar wenigen Profis in ihrer Gesamtheit beherrscht wird.

Schlussbetrachtung

Es ist nicht trivial, Prozesse derart zu koordinieren, dass sie harmonisch zusammenwirken. Im vorliegenden Artikel wurden einige Techniken aufgezeigt, mit denen sich Konfliktsituationen prinzipiell meistern lassen. Nicht gezeigt wurde allerdings, wie Prozesssysteme zu modellieren sind. Dieses Gebiet wurde ausgespart, da es keine allgemeinen Modellierungsmethoden oder -vorschriften gibt, sondern nur Richtlininien in Bezug auf mögliche Deadlocks. So hängt es auch heute noch ganz einfach von der Erfahrung und Intuition des jeweiligen Programmierers ab, ob das Programm letztlich die geforderten Leistungsmerkmale erbringen wird oder nicht. Offengelassen wurde auch die Frage, wie Prozesssysteme auf Konsistenz geprüft werden können. Wohl existieren Verfahren für die Ueberprüfung des korrekten Zusammenwirkens von Prozessen, diese sind allerdings derart umfangreich und komplex, dass sie den Rahmen dieses Artikels bei weitem gesprengt hätten. □

Literatur

[1] E.W. Dijkstra; Co-operating Sequential Processes, Programming Languages, Academic Press, 1968

[2] C.A.R. Hoare; Monitors: An Operating System Structuring Concept, Communication of the ACM, Vol. 17, Number 10, October 1974

[3] P. Wegner, A.S. Smolka; Processes, Tasks, and Monitors: A Comparative Study of Concurrent Programming Primitives, IEEE Transaction on Software Engineering., Vol. SE-9, No. 4, July 1983

[4] W. Reisig, Petrinetze; Eine Einführung Springer Verlag, 1986 (2. Auflage)

[5] CHILL - A Self-Introduction Manual Vol. I,II,III Philips, 1983

[6] N. Wirth; Programming in Modula-2, Third corrected edition, Springer Verlag, 1985

[7] Modula-2/86 - User's Manual Logitech SA, 1984

[8] Disk Operating System Version 3.00 - Technical Reference International Business Machines Corp., 1984

[9] Personal Computer - Technical Reference International Business Machines Corp., 1983

[10] Introduction to CHILL CCITT, Study Croup XI, 1980

[11] CHILL Language Definition CCITT, Rec. Z.200 (1984), 1984

Vom Umgang mit dBase III PLUS (5)

Die verschiedenen Techniken und Befehle zum Editieren von Datensätzen haben wir in M+K 87-5 grundlegend besprochen. Nachdem so alle Voraussetzungen geschaffen sind, werden wir nun die erarbeiteten Befehle zu einem kompletten Programm miteinander verknüpfen und ein komfortables Mutationsprogramm erstellen. Im zweiten Teil dieses Beitrags wird dann die Funktionsweise des Clipper, der am Anfang dieses Lehrgangs erwähnt wurde, näher betrachtet.

Heinz Kastien

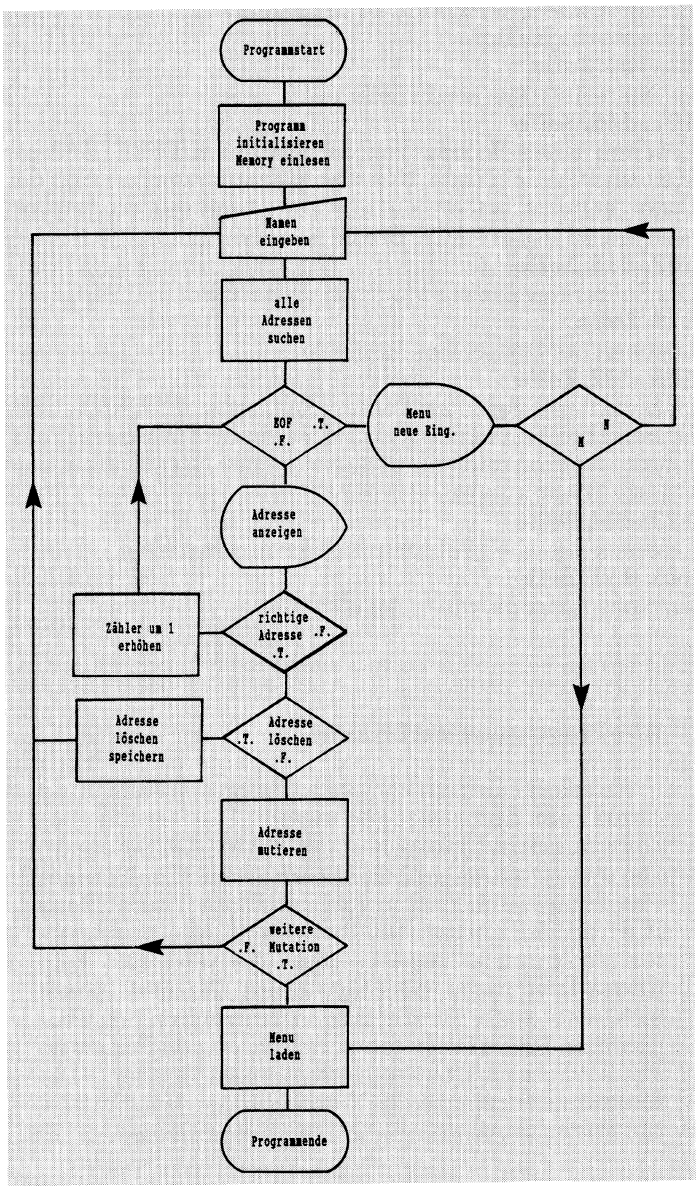
Mutationsprogramm

Das Editieren von Datensätzen umfasst: Suchen nach einem Schlüsselbegriff (Namen), Auswahl aus einer Reihe gleichartiger Schlüsselbegriffe, Mutieren der Datenfelder, eventuelles Löschen der Datensätze, Abspeichern der mutierten Datensätze und Rückkehr zum Menü.

Da dBase III PLUS weder einen Sprungbefehl noch Subroutinen im Sinne des BASIC kennt, sondern dBase-Prozeduren strukturiert aufgebaut sind, muss das Programm gut geplant sein, um unnötige Programmschritte zu vermeiden. Das nebenstehende Flowchart zeigt das Mutationsprogramm.

Im Programm erübrigt sich eine Beschreibung der Einzelbefehle, da diese bereits in der letzten Folge eingehend erläutert worden sind.

1. Zeile
Remark
2. Zeile
Initialisierung der Datenbank «Mitglied» und der Indexdatei «NAMIND»
3. Zeile
Einlesen von Variablen aus der Memorydatei «DATEN», es sind dies die Vorbelegungen der RANGE
4. Zeile
Beginn der DO....WHILE-Schleife zu Zeile 161
5. Zeile
Definition der Variablen SNAME mit 40 Leerzeichen
6. Zeile
Definition der Variablen «ANTWORT»
7.-42. Zeile
Bildschirmaske. Diese Maske wurde aus dem Eingabeprogramm kopiert und ist mit diesem identisch.
43.-45. Zeile
In der Bildschirmzeile Mitgliedername wird mit GET SNAME der Schlüsselbegriff, in diesem Fall der Name, erfragt und mit READ übernommen.
46. Zeile
FIND sucht auf Grund der Indexdatei NAMIND nach dem ersten Schlüsselbegriff. Wird vor den Schlüsselbegriff ein «&» Zeichen gesetzt, so werden alle Datensätze dieses Schlüsselbegriffs gesucht.
47. Zeile
IF...ENDIF-Schleife zu Zeile 89. Mit dieser Routine wird die Suche abgebrochen, sofern das Dateiende erreicht ist, also EOF gleich .T. wird.
48. Zeile
Wird der gesuchte Schlüsselbegriff bis zum Erreichen des Dateiendes nicht gefunden, erfolgt die entsprechende Meldung.
49. Zeile
DO...WHILE-Schleife zur Rückkehr ins Menü «M», oder Neubeginn «N», bei anderen Eingaben wird die Schleife erneut durchlaufen. Zu dieser Schleife gehört der ENDDO-Befehl in Zeile 60.
50.-51. Zeile
Eingabe von «N» Neubeginn, oder «M» für die Menüwahl.
52.-59. Zeile
DO...CASE-Routine zur obigen Abfrage. Wird mit «M» die Menüwahl getätigt, kehrt das Programm mit RETURN zum Hauptmenü zurück (Zeile 53-54). Bei einem Neubeginn mit «N» (Zeile 53-54) verlässt das Programm die DO... WHILE Schleife und kehrt durch den LOOP-Befehl in Zeile 61 zum Anfang des Programms zurück. Alle anderen Eingaben werden ignoriert, indem das Programm mit Zeile 57-58 die Schleife wiederholt.
62. Zeile
Wird ein Name gefunden und ist das Dateiende nicht erreicht, verzweigt das Programm mit ELSE in Zeile 62 nach Zeile 63.



63. Zeile
Diese DO...WHILE-Schleife ist an zwei Bedingungen geknüpft, nämlich dass der gesuchte Schlüsselbegriff SNAME gefunden wurde, also SNAME=NAME ist und EOF ungleich .T. ist.
64.-75. Zeile
Darstellung aller Felder auf dem Bildschirm.
76.-82. Zeile
Sofern der Schlüsselbegriff nicht eindeutig ist, also der gesuchte Name mehrfach vorkommt, kann hier ein weiterer Record aufgerufen werden. Diese DO...WHILE-Schleife bewirkt eine Ja/Nein-Abfrage und die entsprechenden Verzweigung. Dieser Programmteil ist mit den Zeilen 93-100 und 145-151 identisch. Es handelt sich um eine DO...WHILE-Schleife mit der Bedingung ANTWORT <> «J» oder <> «N». Ist die Antwort «J» oder «N», wird die Schleife mit der IF...ENDIF-Verzweigung und dem EXIT-Befehl verlassen.
83. Zeile
Im Falle einer «Ja»-Antwort wird die DO...WHILE-Schleife in Zeile 85 verlassen und das Programm in Zeile 93 fortgesetzt.
87. Zeile
Bei einer «Nein»-Antwort wird der Recordzähler um 1 erhöht. Wird EOF()= .T. kehrt das Programm in Zeile 88 zu Zeile 4 zurück, wird der gleiche Schlüsselbegriff noch einmal gefunden, wird nach Zeile 63 verzweigt und die neue Adresse angezeigt.
93.-100. Zeile
Ja/Nein-Abfrage, wie in Zeile 76-82.
101.-106. Zeile
Löschen eines Records mit den Befehlen DELETE (Zeile 102) und PACK (Zeile 103). Die Records werden aus der Datei entfernt; sollen sie nur zur Löschung vorgemerkt werden, so ist der PACK-Befehl zu entfernen.
107.-118. Zeile
Die Felder werden zur Mutation umbenannt.
119. Zeile
Löschen der Bildschirm-Fusszeile.
120.-132. Zeile
Uebernahme der alten Feldinhalte mit <RETURN> oder Ueberschreiben mit dem neuen Feldinhalt, die alten Feldinhalte werden auch dann übernommen, wenn sie mit dem Cursor übersprungen werden.
133.-144. Zeile
Ersetzen des alten Feldinhaltes durch den neuen.
145.-151. Zeile
Ja/Nein-Abfrage, wie in Zeile 76-82.

Alle Programme auf Diskette

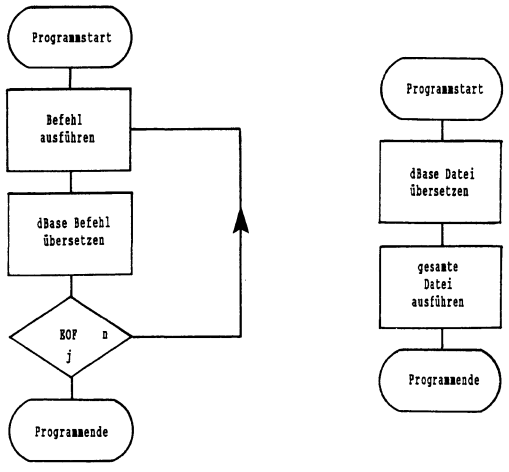
Ab sofort kann die komplette Programmsammlung direkt beim Autor bezogen werden. Die Sendung besteht aus zwei Disketten im 360 KB MS-DOS-Format. Die erste Diskette enthält alle abgedruckten Source-Programme und läuft sowohl mit dBase III und dBase III PLUS. Auf der zweiten Diskette sind die compilierten Programme abgespeichert. Es wird also kein dBase-Interpreter benötigt, diese Diskette eignet sich also auch für die reinen Anwender des Programms. Beide Disketten benötigen einen Printer im ESC/P-Format. Das Programm ist selbststartend. Die Lieferung erfolgt gegen Voreinzahlung von Fr. 50.-- auf Postscheckkonto 60-42710-5, Heinz Kastien. □

152.-160. Zeile
DO...CASE-Routine. Die Abfrage «Ja» kehrt zu Zeile 4 zurück und leitet eine neue Mutation ein. Mit der Antwort «Nein» schliesst das Programm die Datei und kehrt mit RETURN zum Menü zurück.
161. Zeile
ENDDO zu DO...WHILE aus Zeile 4.

Der Programmablauf selbst bedarf keiner weiteren Erklärungen, da er mit den notwendigen Kommentaren versehen ist. Das Programm sollte, wie auch alle anderen bisher beschriebenen Programmteile, immer vom Menü aufgerufen werden, da sonst ECHO, STATUS und HEADING aktiv sind. Das Indexregister NAMIND wird automatisch nachgeführt.

Compiler

Schon zu Beginn dieses Lehrganges wurde die Compilierung von Programmen erwähnt. Grundsätzlich versteht man hierunter die Uebersetzung einer Hochsprache, also in diesem Fall dBase, in Maschinensprache. Im dBase übernimmt ein Interpreter diese Aufgabe. Nach dem Programmstart übersetzt der Interpreter Zeile für Zeile das dBase-Programm in den Maschinencode, der dann ausgeführt wird. Liegen im dBase-Programm Syntax-Fehler vor, so werden diese erst bei Erreichen der entsprechenden Zeile auf dem Bildschirm angezeigt und das Programm abgebrochen. Dies ist umso ärgerlicher, je später der Fehler im Ablauf eines langen Programms auftritt. Ausserdem ist diese Methode sehr Zeit-aufwendig, da die Uebersetzung während jedes Programmablaufes neu vorgenommen wird. Der Vorteil dieser Sourceprogramme ist die leichte Editierbarkeit. Beim Auftreten eines Fehlers kann dieser mit dem Editor behoben und das Programm neu gestartet werden. Die dBase-Programme benötigen also auch beim reinen Programmablauf die Anwesenheit des Interpreters.



Interpreter *Compiler*

Der Compiler unterscheidet sich vom beschriebenen Interpreter wesentlich, zwar übersetzt auch er die Hochsprache in den Maschinencode, jedoch erfolgt die Uebersetzung unabhängig vom Programmablauf. Meist wird das gesamte Programm in eine Quelldatei geschrieben und nach deren Test compiliert. Durch die Compilierung liegt schon beim Programmstart ein Maschinenprogramm vor, dadurch erhöht sich die Geschwindigkeit der Programmausführung um ein Vielfaches. Ausserdem verlangt ein derartig compiliertes Programm sehr grosse Kenntnis-

se, um es lesen zu können, daher sind die compilierten Programm vor unbefugtem Zugriff fast hundertprozentig geschützt. Der Clipper verfügt über einen Debugger, der in das Programm eingebunden werden kann. Der Debugger erlaubt: Eine Untersuchung von Speichervariablen und deren Inhalt; Veränderung des Inhaltes der Speichervariablen; Anzeige des Status der Programmumgebung; Unterbrechungspunkte setzen; Ausdrücke betrachten und verändern; Umleiten von Debuggerausgaben an den Drucker.

Obwohl der Debugger eine Reihe von Veränderungen der compilierten Programmversion zulässt, ist der Umgang mit dieser Option relativ kompliziert und es ist daher empfehlenswert, das Sourceprogramm aufzubewahren und bei erforderlichen Aenderungen, diese am Sourceprogramm vorzunehmen und dieses anschliessend neu zu compilieren. Der Compiler ist selbsttragend, d.h. er benötigt zur Ausführung keinen zusätzlichen Editor. Clipper verfügt jedoch noch über eine ganze Reihe von Eigenschaften, die denen des dBase klar überlegen sind. Während dBase III PLUS 256 Speichervariablen und maximal 128 Felder pro Datenbank zulässt, beschränkt der Clipper die maximale Anzahl der Speichervariablen auf 64'000 und die Anzahl der Felder pro Datenbank auf 1'024. Es gibt aber noch eine ganze Reihe weiterer Vorteile, die die Anschaffung des Clippers rechtfertigen, so ist es möglich, selbstentworfene Hilfsprogramme zu verwenden, die vom Anwender im Bedarfsfall leicht aufgerufen werden können. Ausserdem können MEMO-Felder, deren Bearbeitung im dBase nicht ganz problemlos ist, in Stringvariablen umgewandelt werden. Der Anwender kann darüber hinaus eigene Funktionen definieren und die Funktionsbibliothek im Linklauf in das Programm einbauen. Wichtige dBase-Befehle, wie CREATE, LABEL und REPORT sind beim Clipper in eigenen Hilfprogrammen installiert. Einige dBase-Befehle werden vom Clipper nicht unterstützt, dies sind vor allem die interaktiven Befehle

- ASSIST und HELP
- BROWSE, CHANGE, EDIT, INSERT und MODIFY COMMAND
- SET, SET HELP, SET HEADING, SET MENU, SET SAFETY und SET TALK
- RETURN TO MASTER

Ausserdem haben einige wenige Befehle, wie DISPLAY, LIST, AVERAGE und SUM einen anderen Befehlssyntax.

Normalerweise wird ein Programm sofort nach seiner Fertigstellung compiliert. Die Compilierung erfolgt in zwei Stufen. In der ersten Stufe wird das ASCII-File in ein Maschinencode-ähnliches Objektfile umgewandelt. Hierbei werden alle Syntax-Fehler, die noch im Programm enthalten sind, erkannt und entweder auf dem Bildschirm angezeigt, oder in eine Fehlerdatei auf Disk geschrieben. Das Objektfile ist aber für sich allein noch nicht lauffähig, sondern muss noch in einem zweiten Arbeitsgang mit einem Laufzeitmodul versehen werden. Diesen Vorgang, der Linken genannt wird, wandelt das Objektfile in ein .EXE-Programm um, das direkt aus der Betriebssystemebene gestartet wird. Gleichzeitig bindet der Linkvorgang auch zusammengehörige Programmteile ein, so werden z.B. alle Programmteile, die sich vom Menü aufrufen lassen, in einem einzigen Programm verknüpft. Die gelinkten Programme können sehr gross werden, vor allem wenn man bedenkt, dass z.B. ein einziger IF...-Befehl bis zu 20 Maschinenbefehlen entspricht. Werden die Programme zu gross oder wird diese Technik nicht gewünscht, so können OVERLAY-Dateien erstellt werden. Bei dieser Programmiertechnik wird nur ein Teilprogramm in den Arbeitsspeicher geladen und alle anderen Programmteile, je

```
*** M+K Vereinsadressmutationsprogramm by H. Kastien 19.08.1987 ***
USE MITGLIED INDEX NAMIND
RESTORE FROM DATEN
DO WHILE .T.
SNAME=SPACE(40)
ANTWORT = "M"
SET COLOR TO W/
CLEAR
@ 0, 0 say "┌──────────────────────────────────────────────────────────────────────────────┐"
@ 1, 0 say "│      M+K      │"
SET COLOR TO B/
@ 1,30 say "Vereinsadressverwaltung"
SET COLOR TO W/
@ 1,67 say "│ H.Kastien ┃"
@ 2, 0 say "└──────────────────────────────┴──────────────────────────────────────────────┛"
@ 3, 0 say "┃"
@ 4, 0 say "┃"
SET COLOR TO +W/
@ 4, 7 say "Mitgliedername "
SET COLOR TO W/
@ 4,28 say ":                                                          ┃"
@ 5, 0 say "┃                                                                            ┃"
@ 6, 0 say "┃      Vorname              :                                                ┃"
@ 7, 0 say "┃                                                                            ┃"
@ 8, 0 say "┃      Strasse              :                                                ┃"
@ 9, 0 say "┃                                                                            ┃"
@10, 0 say "┃      PLZ :           Ort :                                                 ┃"
@11, 0 say "┃                                                                            ┃"
@12, 0 say "┃      Geburtsdatum         :          Eintrittsdatum :                      ┃"
@13, 0 say "┃                                                                            ┃"
@14, 0 say "┃      Telefonnummer        :                                                ┃"
@15, 0 say "┃                                                                            ┃"
@16, 0 say "┃      Mitgliederart        :                                                ┃"
@17, 0 say "┃                                                                            ┃"
@18, 0 say "┃      Mitgliederbeitrag    :            Fr.  Zahlung :                      ┃"
@19, 0 say "┃                                                                            ┃"
@20, 0 say "┃      Bemerkung            :                                                ┃"
@21, 0 say "┃                                                                            ┃"
@22, 0 say "┃                                                                            ┃"
@23, 0 say "┃                                                                            ┃"
@24, 0 say "└────────────────────────────────────────────────────────────────────────────┘"
```

```
42. SET COLOR TO
43. SET COLOR TO W/,/W
44. @  4,30 GET SNAME
45. READ
46. FIND &SNAME
47. IF EOF()
48. @ 22,21  SAY "Der Name "+TRIM(SNAME)+" ist nicht in der Datei ! "
49. DO WHILE ANTWORT <> "M" .OR. ANTWORT <> "N"
50. @ 23,21 SAY "Weiter mit N (Neubeginn) oder M zum Menu !" GET ANTWORT
51. READ
52. DO CASE
53. CASE UPPER(ANTWORT) = "M"
54. RETURN
55. CASE UPPER(ANTWORT) = "N"
56. EXIT
57. CASE ANTWORT <> "M" .OR. ANTWORT <> "N"
58. LOOP
59. ENDCASE
60. ENDDO
61. LOOP
62. ELSE
63. DO WHILE SNAME = NAME .AND. .NOT.EOF()
64. @  4,30 SAY NAME
65. @  6,30 SAY VORN
66. @  8,30 SAY STRA
67. @ 10,13 SAY POLZ
68. @ 10,30 SAY ORTB
69. @ 12,30 SAY GDAT
70. @ 12,59 SAY EDAT
71. @ 14,30 SAY TELN
72. @ 16,30 SAY MITA
73. @ 18,30 SAY BEIT
74. @ 18,58 SAY ZAHL
75. @ 20,30 SAY BEME
76. ANTWORT="J"
77. DO WHILE ANTWORT <> "J" .OR. ANTWORT <> "N"
78. @ 22,21 SAY "   Ist dies die richtige Adresse ? " GET ANTWORT
79. READ
80. IF UPPER(ANTWORT)="J" .OR. UPPER(ANTWORT)="N"
81. EXIT
82. ENDIF
82. ENDDO
82. IF UPPER(ANTWORT)="J"
85. EXIT
86. ENDIF
87. SKIP
88. ENDDO && EOF()
89. ENDIF && EOF()
90. IF SNAME <> NAME
91. LOOP
91. ENDIF
93. ANTWORT = "N"
94. DO WHILE ANTWORT <> "J" .OR. ANTWORT <> "N"
95. @ 22,21 SAY "   Wollen Sie die Adresse loeschen ? " GET ANTWORT
96. READ
97. IF UPPER(ANTWORT)="J" .OR. UPPER(ANTWORT)="N"
98. EXIT
99. ENDIF
100. ENDDO
101. IF UPPER(ANTWORT)="J"
```

```
102. DELETE
103. PACK
104. ANTWORT=CHR(13)
105. LOOP
106. ENDIF
107. ENAME=NAME
108. EVORN=VORN
109. EPOLZ=POLZ
110. EORTB=ORTB
111. ESTRA=STRA
112. EGDAT=GDAT
113. EEDAT=EDAT
114. ETELN=TELN
115. EMITA=MITA
116. EBEIT=BEIT
117. EZAHL=ZAHL
118. EBEME=BEME
119. @ 22,21 SAY SPACE(40)
120. @  4,30 GET ENAME
121. @  6,30 GET EVORN
122. @  8,30 GET ESTRA
123. @ 10,13 GET EPOLZ RANGE POLZMIN,POLZMAX
124. @ 10,30 GET EORTB
125. @ 12,30 GET EGDAT
126. @ 12,59 GET EEDAT
127. @ 14,30 GET ETELN
128. @ 16,30 GET EMITA
129. @ 18,30 GET EBEIT RANGE BEITMIN,BEITMAX
130. @ 18,58 GET EZAHL
131. @ 20,30 GET EBEME
132. READ
133. REPLACE NAME WITH ENAME
134. REPLACE VORN WITH EVORN
135. REPLACE STRA WITH ESTRA
136. REPLACE POLZ WITH EPOLZ
137. REPLACE ORTB WITH EORTB
138. REPLACE GDAT WITH EGDAT
139. REPLACE EDAT WITH EEDAT
140. REPLACE BEME WITH EBEME
141. REPLACE TELN WITH ETELN
142. REPLACE MITA WITH EMITA
143. REPLACE BEIT WITH EBEIT
144. REPLACE ZAHL WITH EZAHL
145. DO WHILE ANTWORT <> "J" .OR. ANTWORT <> "N"
146. @ 22,21 SAY "   Wollen Sie weitere Mutationen (j/n)" GET ANTWORT
147. READ
148. IF UPPER(ANTWORT) = "J" .OR. UPPER(ANTWORT) = "N"
149. EXIT
150. ENDIF
151. ENDDO
152. DO CASE
153. CASE UPPER(ANTWORT)="J"
154. SNAME=SPACE(40)
155. LOOP
156. CASE UPPER(ANTWORT)="N"
157. USE
158. RETURN
159. ENDCASE
160. LOOP
161. ENDDO
```

nach Bedarf zugeladen. An den eigentlichen .DBF-Dateien wird nichts geändert, diese liegen in der unveränderten Form vor. Der Clipper ist ein spezieller dBase-Compiler. Die Prozeduren müssen aber nicht unbedingt mit dem dBase-Texteditor geschrieben sein, es eignet sich jede beliebige Textverarbeitung, die ein ASCII-File erstellt. Zur Umwandlung in das Objektfile wird der Clipper aus der MS-DOS-Ebene mit

CLIPPER «Programmname»

aufgerufen. Die Fehlermeldungen erfolgen in diesem Fall auf dem Bildschirm.

CLIPPER «Programmname» > Fehler

schreibt die erkannten Fehler bei der Compilierung in die Datei «Fehler» auf der Disk. Nach erfolgter Compilierung liegt auf der Disk neben dem Sourcefile «Programmname».PRG, ein Objektfile «Programmname».OBJ vor. Bei Fehlermeldungen wird die Compilierung nicht abgebrochen, jedoch muss das Sourceprogramm nach dem Editieren der Fehler neu compiliert werden. Das Linken des Objektfiles wird mit

PLINK86 FILE «Name des Objektfiles»

ausgelöst. Auf Grund der wesentlich höheren Ablaufgeschwindigkeit, des grösseren Feldvolumens der Dateien und der anderen, oben geschilderten Vorteile, kann zur Anschaffung eines Clippers nur geraten werden.

Mit der nächsten Folge beenden wir unseren dBase Kurzlehrgang. Es ist einleuchtend, dass in dieser kurzen Beschreibung nicht alle Feinheiten des dBase erklärt werden konnten. Vor allem mussten die speziellen Eigenschaften des dBase III PLUS in Netzwerken vernachlässigt werden. Dies soll jedoch in späteren Ausgaben innerhalb einer Beschreibung von Netzwerken nachgeholt werden. In der abschliessenden Folge in M+K 88-1 folgt eine statistische Auswertung der Datei nach der Mitgliederart, Anzahl der Mitgliedsjahre und Kontrolle der Beitragszahlungen. Schliesslich soll noch die grafische Darstellung der erwähnten Statistiken, die völlig neu auf dem Markt erschienen ist, unseren dBase-Lehrgang abrunden. □

Effiziente Prozeduren für die Benutzerschnittstelle

Wer oft mit verschiedenen Anwendungsprogrammen auf Computern gearbeitet hat, weiss um die höchst unterschiedlichen Benutzerschnittstellen. Dass ein Programm Fehlbedienungen erkennen und entsprechende Fehlermeldungen ausgeben soll, ist heute eigentlich selbstverständlich. Bei kommerziellen Softwarepaketen wird diesem Grundsatz im allgemeinen gut nachgelebt, denn nur ein leicht zu bedienendes Programm lässt sich bei der heutigen Konkurrenzsituation noch vermarkten.

```
program proc(input, output);
{Prozeduren fuer Ein- und Ausgabe von E. Pfenninger}


const
  bell=^G;                {bewirkt Warnton}
  up=#5;                  {Tastencode von Cursor nach oben}
  down=#24;               {Tastencode von Cursor nach unten}
  enter=#13;              {Tastencode der ENTER-Taste}
  befehlsZeile=23;        {Zeile, auf welcher die Befehle erscheinen}
  errorZeile=24;          {Zeile, auf der die Fehlermeldungen erscheinen}
  version='5.8.1986';     {nur fuer Demoprogramm verwendet}


type
  str13=string[13];
  str80=string[80];

var
  zahlText: str13;            {nur fuer Demoprogramm verwendet}
  wert, funktion: integer;    {nur fuer Demoprogramm verwendet}
  realWert: real;             {nur fuer Demoprogramm verwendet}
  abbruch: boolean;           {nur fuer Demoprogramm verwendet}


procedure befehl(text:str80);
{Der Inhalt von 'text' wird auf die Befehlszeile geschrieben.}
{Ist die Variable 'text' leer, wird die Befehlszeile geloescht.}
begin
  gotoXY(1,befehlsZeile); clrEol;
  gotoXY(1,befehlsZeile); write(text);
end; {befehl}


procedure error(text:str80);
{Der Inhalt von 'text' wird auf die Errorzeile geschrieben und}
{der Warnton wird ausgegeben.}
{Ist die Variable 'text' leer, wird die Errorzeile geloescht.}
begin
  if not (text='') then write(bell);
  gotoXY(1,errorZeile); clrEol;
  gotoXY(1,errorZeile); write(text);
end;  {error}


procedure format(argument:real; var zahlText:str13);
{Die Variable 'argument' wird in eine Zeichenkette 'zahlText'}
{umgewandelt. Es kommen nur Zehnerpotenzen mit Exponenten der}
{Dreierreihe vor. Es wird auf 6 Stellen gerundet.}
var
  code,exponent: integer;
  rest:byte;
  wert: real;
  hilfsStr: string[11];
begin
  wert:=abs(argument);
  str(wert:11,hilfsStr);
  if hilfsStr[1]='0' then zahlText:='       0'
  else
  begin
    if argument<0 then zahlText:='-' else zahlText:='';
    val(copy(hilfsStr,10,2),exponent,code);
    rest:=exponent mod 3;
    if hilfsStr[9]='-' then exponent:=-exponent;
    if (exponent<0) and (rest<>0) then rest:=3-rest;
    if rest=0 then zahlText:=zahlText+copy(hilfsStr,1,7)
    else
    begin
      zahlText:=zahlText+hilfsStr[1]+hilfsstr[3];
      if rest=1 then zahlText:=zahlText+'.'+copy(hilfsStr,4,4)
      else zahlText:=zahlText+hilfsStr[4]+'.'+copy(hilfsStr,5,3);
      exponent:=exponent-rest;
    end;
    if abs(exponent)>9 then str(exponent:3,hilfsStr)
      else str(exponent:2,hilfsStr);
    if exponent<>0 then zahlText:=zahlText+' E'+hilfsStr;
  end;
end; {format}
```

Bei privaten Programmen, welche in der Problemlösung oft ein sehr hohes Niveau erreichen, wird diesem Punkt oft zu wenig Beachtung geschenkt. Meist fehlt dort die Zeit oder das Interesse, dem Benutzer einen gewissen Komfort bei der Bedienung und allfälligen Bedienungsfehlern zu bieten. Dabei wäre mit geringem Zusatzaufwand die Benutzerschnittstelle zu verbessern. Im folgenden stellen wir einige Prozeduren in Pascal vor, welche in allen Programmen leicht eingebaut werden können und die Handhabung der Software erleichtern. Gleichzeitig kann auf diese Weise eine einheitlichere Bedienung erreicht werden, indem sowohl innerhalb des Programmes wie im Vergleich zu anderen Programmen dieselben Bedienungsprozeduren auftreten.

Ernst Pfenninger

Die Vorteile von Pascal zeigen sich hier deutlich, denn es lassen sich Funktionen und Prozeduren festlegen, welche eine klar definierte Schnittstelle nach aussen aufweisen und daher leicht einzubauen sind. Der innere Aufbau interessiert später nicht mehr, ja er kann sogar geändert werden ohne weitere Auswirkungen, falls dabei die Schnittstelle gleich bleibt. Das Listing zeigt einen Satz von Prozeduren in Turbo-Pascal, welche jedem Programm beigefügt werden können - zum Beispiel als Include-Datei. Durch die Verwendung dieser bewährten Prozeduren reduziert sich auch der Aufwand bei der Erstellung eines neuen Programmes erheblich.

Die Prozeduren

Für die Eingabe wird eine Befehlszeile definiert, auf welcher die Anweisungen an den Benutzer erscheinen. Es wird die Zeile 23 verwendet; dieser Wert kann bei Bedarf geändert werden. Die Prozedur «befehl» löscht die Befehlszeile und beschreibt sie mit der als Parameter übergebenen Zeichenkette. Besteht der Parameter aus einer Zeichenkette der Länge Null, wird die Befehlszeile nur gelöscht.

Für Fehlermeldungen wird eine eigene Zeile reserviert. Dies ist die Zeile 24. Auch dieser Wert kann angepasst werden. Die Prozedur «error» funktioniert im übrigen wie die Funktion «befehl», mit dem Unterschied, dass jedesmal das Glockenzeichen mit ausgegeben wird. Dies natürlich nur,

falls auch eine Fehlermeldung übergeben wurde.

Eine weitere Prozedur zur Vereinfachung der Ausgabe ist die Prozedur «format». Sie bewerkstelligt die Umwandlung einer beliebigen reellen Zahl in eine formatierte Zeichenkette. Dies ist eine oft benötigte Funktion, denn Pascal bietet leider nur wenig Möglichkeiten, die Ausgabe von Zahlen ansprechend zu gestalten. Eine reelle Zahl soll eine fixe Anzahl Ziffern aufweisen, um die Genauigkeit sicherzustellen. Der Exponent sollte ein Vielfaches der Zahl drei sein, denn nur solche Werte können vernünftig interpretiert werden. In der Elektrotechnik wird beispielsweise mit den Grössen Mikro; Milli, Kilo, Mega usw. gerechnet, so dass ein Progrmm die Werte ebenfalls in dieser Reihe - allerdings als Exponent - angeben soll. Die Prozedur erhält als Parameter die Ausgangszahl des Typs «Real», worauf dem zweiten Parameter die Zeichenkette der formatierten Zahl übergeben wird. Dieser ist vom Typ «String[13]».

Die Funktion «menuWahl» dient zur leichten Generierung eines Menüs. Die Menüeinträge müssen zuerst ab der dritten Zeile zeilenweise auf den Bildschirm gebracht werden. Die Zeilen 1 und 2 können für den Menütitel verwendet werden. Am rechten Rand sind die Spalten 1-6 frei zu lassen, denn dort plaziert die Funktion «menuWahl» den Zeiger. Wenn die Menüeinträge auf dem Bildschirm sind, muss die Funktion «menuWahl» aufgerufen werden. Als Parameter ist die Anzahl Menüeinträge zu übergeben. Die Funktion liefert als Wert die Nummer des gewählten Eintrages. Die Auswahl des Eintrages geschieht durch die Plazierung des Zeigers neben dem gewünschten Eintrag, was die Wahrscheinlichkeit von Fehleingaben klein hält. Der Benutzer hat den gewählten Eintrag vor Augen und kann beliebig korrigieren, was wesentlich einfacher ist als die Eingabe einer blossen Zahl.

Die Prozedur «warteAufTaste» macht nichts anderes als zu warten, bis eine beliebige Taste betätigt wird. Die Funktion «positiveAntwort» ist nützlich, falls eine Frage mit «ja» oder «nein» beantwortet werden soll. Zuerst ist auf dem Bildschirm die Frage anzuzeigen. Danach wird die Funktion «positiveAntwort» aufgerufen. Sie schreibt die beiden Antwortmöglichkeiten auf den Bildschirm und wartet danach auf die Eingabe von «j» oder «n». Falls die Taste «j» gedrückt wird, ist der Wert der Funktion TRUE, bei Betätigung der Taste «n»

```
function menuWahl(anz:integer):integer;
{Die Funktion gibt die Nummer des gewaehlten Menueintrages als}
{Wert zurueck. Es sind 'anz' Menueintraege insgesamt vorhanden.}
{Diese sind vorgaengig auf den Bildschirm zu bringen ab der dritten}
{Zeile. Rechts ist ein freier Rand der Breite 6 zu lassen.}
var zahl, altZahl: integer;
    eingabe: char;
begin
  befehl('Cursor fuer auf, ab    ENTER fuer Wahl');
  zahl:=1; altZahl:=zahl;
  repeat
    gotoXY(1,altZahl+2); write('    '); altZahl:=zahl;
    gotoXY(1,zahl+2); write('--->'); gotoXY(80,zahl+2);
    read(kbd, eingabe);
    error('');
    case eingabe of
      up:   if zahl>1 then zahl:=pred(zahl);
      down: if zahl<anz then zahl:=succ(zahl);
      enter: begin end;
    else
      error('Nur Cursortasten fuer auf, ab und ENTER-Taste sind erlaubt!');
    end;  {case}
  until eingabe=enter;
  menuWahl:=zahl;
end;  {menuWahl}


procedure warteAufTaste;
{Die Prozedur wartet auf die Eingabe einer beliebigen Taste.}
var zeichen: char;
begin
  befehl('Weiter mit beliebiger Taste!  ');
  read(KBD,zeichen);
  befehl('');
end;  {warteAufTaste}


function positiveAntwort:boolean;
{Die Funktion verlangt nach der Eingabe von 'j' oder 'n'.}
{Falls 'j' eingegeben wird, ist der Wert der Funktion TRUE.}
var zeichen: char;
    ok: boolean;
begin
  write('(j/n)  ');
  repeat
    read(KBD,zeichen);
    ok:=(upCase(zeichen) in ['J','N']);
    if not ok then write(bell);
  until ok;
  write(zeichen);
  positiveAntwort:=(upCase(zeichen)='J');
end;  {positiveAntwort}


function inputInteger(uGrenze,oGrenze,zeile, spalte:integer):integer;
{Es wird eine ganze Zahl zwischen 'uGrenze' und 'oGrenze' als Eingabe}
{akzeptiert. Die Eingabe erfolgt bei den Bildschirmkoordinaten}
{'zeile', 'spalte'. Der Wert der Funktion ist gleich der eingegebenen}
{Zahl.}
var zahl: integer;
    ok: boolean;
begin
  befehl('Eingabe einer ganzen Zahl');
  repeat
    gotoXY(spalte,zeile); write('      '); gotoXY(spalte,zeile);
    bufLen:=6; {$I-} read(zahl) {$I+};
    ok:=((IOResult=0) and (zahl>=uGrenze) and (zahl<=oGrenze));
    gotoXY(1,errorZeile);
    if not ok then
    begin
      write(bell,'Die Zahl muss zwischen ',uGrenze,' und ',oGrenze,
            ' liegen!');
    end
    else clrEol;
  until ok;
  befehl('');
  inputInteger:=zahl;
end;  {inputInteger}


function inputReal(uGrenze,oGrenze:real; zeile,spalte:integer):real;
{Es wird eine reelle Zahl zwischen 'uGrenze' und 'oGrenze' als Eingabe}
{akzeptiert. Die Eingabe erfolgt bei den Bildschirmkoordinaten}
{'zeile', 'spalte'. Der Wert der Funktion ist gleich der eingegebenen}
{Zahl.}
var zahl: real;
    ok: boolean;
    zahlText: str13;
begin
  befehl('Eingabe einer reellen Zahl');
  repeat
    gotoXY(spalte,zeile); write('                ');
    gotoXY(spalte,zeile);
    bufLen:=17; {$I-} read(zahl) {$I+};
    ok:=((IOResult=0) and (zahl>=uGrenze) and (zahl<=oGrenze));
    gotoXY(1,errorZeile);
    if not ok then
```

```
    begin
      format(uGrenze,zahlText);
      write(bell,'Die Zahl muss zwischen ',zahlText,' und ');
      format(oGrenze,zahlText);
      write(zahlText,' liegen!');
    end
    else clrEol;
  until ok;
  befehl('');
  inputReal:=zahl;
end;  {inputReal}


{Hauptprogramm zur Demonstration der obigen Prozeduren und Funktionen}
begin
  repeat
    abbruch:=false;
    clrScr; write('Version ',version); gotoXY(1,3);
    writeln('     Eingabe einer ganzen Zahl zwischen -1 und 17');
    writeln('     Eingabe einer reellen Zahl zwischen -10 E 9 und 10 E 12');
    writeln('     Es geht gleich weiter ...');
    writeln('     Abbruch des Demonstrationsprogrammes');
    funktion:=menuWahl(4);
    case funktion of
      1: begin
           gotoXY(1,10); write('Zahlwert?');
           wert:=inputInteger(-1,17,10,12);
         end;
      2: begin
           gotoXY(1,12); write('Zahlwert?');
           realWert:=inputReal(-10E9,10E12,12,12);
           format(realWert,zahlText);
           gotoXY(40,12); writeln; writeln(zahlText);
           warteAufTaste;
         end;
      3: begin
           gotoXY(1,14); write('Es geht gleich weiter ...');
           warteAufTaste;
         end;
      4: begin
           befehl('');
           gotoXY(1,16); write('Wollen Sie das Programm beenden? ');
           abbruch:=positiveAntwort;
         end;
    end;  {case}
  until abbruch;
end.
```

ist sie FALSE. Bei Fehlereingaben ertönt die Glocke.

Die Funktionen «inputReal» und «inputInteger» dienen zur Eingabe einer Real-Zahl bzw. einer Integer-Zahl in einem gewissen Bereich. Beide Funktionen verlangen als Parameter die untere und obere Grenze des Wertes sowie den Ort (Bildschirmkoordinaten), wo die Eingabe erfolgen soll. Bei fehlerhaften Eingaben wird eine Fehlermeldung ausgegeben. Der Wert der Funktionen ist die eingegebene Zahl vom Typ «real» bzw. «integer».

Das Demoprogramm

Das Hauptprogramm dient zur Demonstration aller Prozeduren und soll die Anwendung und den Prozeduraufruf veranschaulichen. Der Leser wird leicht Anwendungen mit praktischem Nutzen finden, denn Ein- und Ausgabe sind zentrale Punkte in jedem Programm. Der Autor hat die Prozeduren schon in vielen Programmen verwendet; die Zeitersparnis ist enorm, wenn nicht jedesmal neue Ein- und Ausgaberoutinen geschrieben werden müssen. □

EPSON Laserdrucker GQ-3500

Der Unterschied:

EPSON ist der führende Drucker-Hersteller. Mit dem weltweit verbindlichen ESC/P Standard (EPSON Standard Code for Printers) erreichte EPSON die totale Kompatibilität mit dem Industriestandard der Software- und Computerhersteller und setzte damit entscheidende Akzente.

Mit dem neuen Laserdrucker GQ-3500 verwirklicht EPSON jetzt benutzerfreundliche Lasertechnologie.
Was die Zukunft noch offen lässt, ist bei EPSON immer bereits vorprogrammiert. Im Laserdrucker GQ-3500, zum Beispiel durch die EPSON Memory-Karte.
Damit erhalten Sie die unbeschränkte Laser-Drucker-Technologie in die eigene Hand.

Das ist der Unterschied von EPSON, damit Sie den Anschluss an die Zukunft nicht verpassen.

Kompakt leise

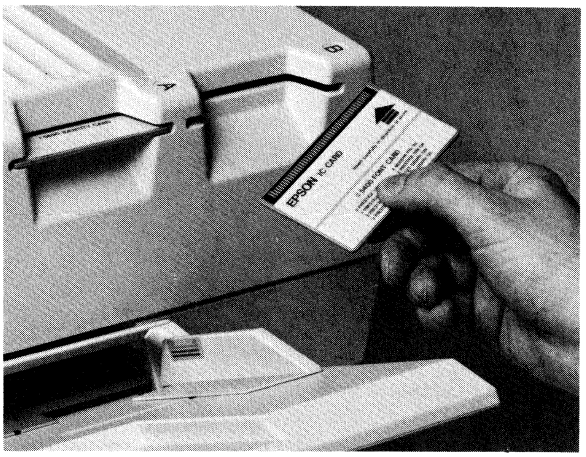
Der GQ-3500 kommt bei keinem Einsatz in die Quere. Weder durch seinen Platzbedarf, noch durch störende Arbeitsgeräusche. Mit seinen Abmessungen, 215 mm hoch, 640 mm breit, 405 mm tief, lässt er sich neben jedem PC plazieren. Dort arbeitet er so flüsterleise, dass er fast übersehen wird.

Gestochen scharf in vielen Schriften

Sieben eingebaute Schriften ergeben gestochen scharfe Schriftbilder. Über EPSON Memory-Karten können in Zukunft aus der Softwarebibliothek laufend weitere Schriften gewählt werden.

Rundum kompatibel

Punkto Kompatibilität verwirklicht der EPSON Laserdrucker GQ-3500 kühnste Wünsche. Klar, dass er mit seinem ESC/P (EPSON Standard Code for Printers) mit weitgehend jeder Software zusammenarbeitet. Neu, dass er mit den EPSON Memory-Karten auch den HP Laser Jet plus und den Diablo 630 emuliert. Mit diesen Memory-Karten lassen sich verschiedene Betriebssysteme individuell ansteuern und nachträglich laufend neue Schriften entwickeln.

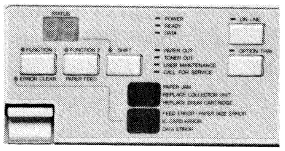


Einer der entscheidenden Unterschiede: Die EPSON Memory-Karte für den sicheren Anschluss an die Zukunft.

Kostengünstig und wartungsfreundlich

Die meisten Wartungen können beim GQ-3500 mit einfachen Handgriffen durch den Anwender durchgeführt werden. Selen-Trommel, Toner und vieles mehr, lassen sich einzeln ersetzen. Das ist einer der vielen Gründe, die den GQ-3500 so kostengünstig machen.

Sekundenschnell und benutzerfreundlich



Die benutzerfreundliche Bedieneroberfläche des neuen EPSON Laserdruckers GQ-3500.

Vergessen Sie Minuten. Die Aufwärmzeit des GQ-3500 dauert bis zur Betriebsbereitschaft nur 30 Sekunden. Nur 60 Sekunden benötigt er, um sechs A4-Seiten auszudrucken. Hoch oder quer, per Knopfdruck, über benutzerfreundliche Funktionstasten. Auf diese Weise können 11 verschiedene Drucker-Funktionen gesteuert werden. Und über die einstellbare Kopierablage liefert er die Kopien auf Wunsch automatisch in chronologischer Reihenfolge.

Farbenfroh handlich

Weil der GQ-3500 etwas gegen konsequente Schwarzseher hat, lässt er sich im Handumdrehen in einen Farbdrucker verwandeln. Mit seinem leicht auswechselbaren Toner-Modul bringt man ihn leicht dazu, auch in Rot, Blau, Grün oder Braun zu flüstern.

Ausbaufähiger Grafik-Speicher

Die Zeichenauflösung von 300 Punkten pro Zoll ist grafikfähig. Und wo der Speicher der Basisausführung von 640 KB für umfangreichere Grafiken oder Desktop Publishing nicht ausreicht, lässt sich der GQ-3500 auf 1,5 MB ausbauen.

Basismodell mit 640 KB nur Fr. 4990.–

EPSON
Technologie, die Zeichen setzt.

Am vorteilhaftesten ist es natürlich, wenn Sie sich den EPSON Laserprinter GQ-3500 bald vorführen lassen, weil Sie sich dann persönlich von diesen entscheidenden Unterschieden überzeugen können.

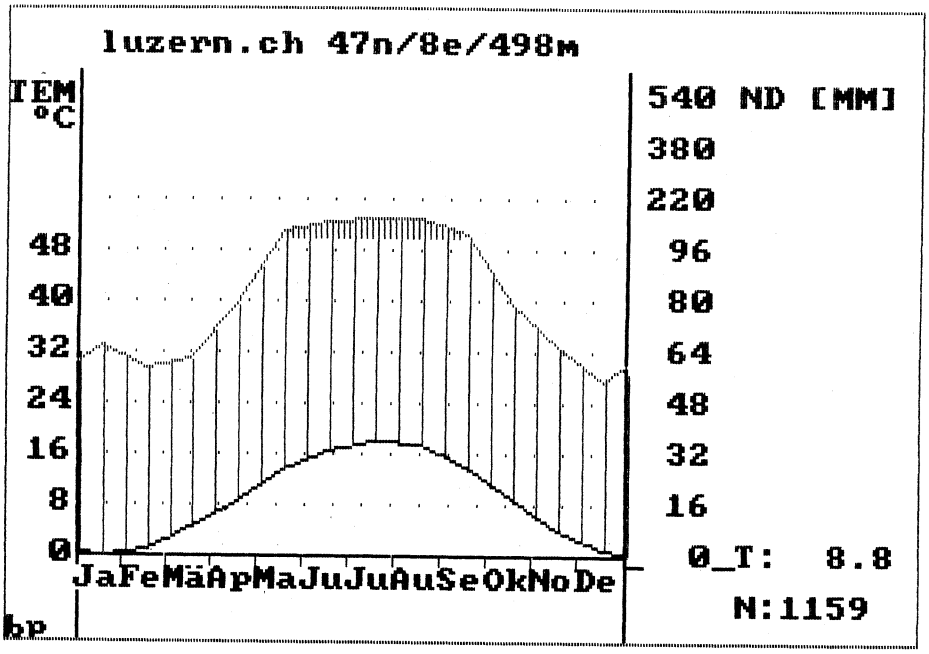
Beratung, Verkauf durch unsere Wiederverkäufer.
Oder besuchen Sie das **EPSON INFORMATION CENTER** am Stauffacher in Zürich.
Generalvertretung für die Schweiz:
Excom AG, Moosacherstr. 6, Au, 8820 Wädenswil, Telefon 01/780 74 14

EXCOM

ADWORK

Klimaauswertung auf dem PC

Jeder, der in fremden Ländern oder an unbekannten Orten Geld oder Zeit investieren möchte, sollte sich über den Standortfaktor Klima möglichst umfassend informieren. Das folgende Programm soll Ihnen helfen, Klimainformationen auszuwerten und grafisch darzustellen. Nebenbei kann der PC-Nutzer seine private Klimadatenbank aufbauen, die ihm dann jederzeit bei einer Standortentscheidung behilflich sein kann.




```
  Type RealFeld=  Array[0..14] of Real;
       Monate=    Array[1..12] of String[15];
       Wort=      String[35];

  Const M:Monate=('Januar........',
                  'Februar.......',
                  'März..........',
                  'April.........',
                  'Mai...........',
                  'Juni..........',
                  'Juli..........',
                  'August........',
                  'September.....',
                  'Oktober.......',
                  'November......',
                  'Dezember......');

  Var A,T,N:      RealFeld;
      H:          Char;
      SN,HW:      Wort;
      Mt,Ma,
      MaxTempDiff:Real;
      Fehler:     Boolean;
      t1:         Integer;

(*-------------------------------------------------------------*)

procedure WarteSchleife;
Var h: Char;
begin
  h:='i';
  Repeat
    Read(kbd,h)
  Until (h=#13) or (h=#32)
end;
(*-------------------------------------------------------------*)

Procedure LiesDaten(Var MA,MT,MaxTempDiff: Real;VAR A,T,N:RealFeld;
Var SN,HW:Wort;Var Fehler:Boolean);
Var
```

Wohl gemerkt: Klima ist nicht das gleiche wie Wetter! Auch wenn das Klimadiagramm einen extrem feuchten August anzeigt, kann innerhalb von zwei Wochen Urlaub stets die Sonne scheinen - oder umgekehrt! Klima ist letztlich eine rein mathematische Grösse: die langjährigen Durchschnitte aus allen relevanten Klimawerten.

Da in der Praxis lediglich die Werte für die Monatsmitteltemperatur und die Monatsniederschlagsmengen über lange Jahre hinweg ausreichend genau aufgezeichnet werden, beschränkt man sich bei Klimadiagrammen auf diese beiden Messgrössen. Durch geschickte Anordnung der Temperatur- und Niederschlagsskala (10 Grad Celsius entsprechen 20 mm Niederschlag) erzielt man darüberhinaus noch eine zusätzliche Infor-

Bruno Pohl

mation über Aridität (= Trockenheit) und Humidität (= Feuchtigkeit). In ariden Gebieten (Trockengebieten) könnte eigentlich mehr Wasser verdunsten als fällt, in humiden Gebieten (wie z.B. der Schweiz) fällt mehr Niederschlag als verdunsten kann.

Damit Aridität und Humidität sofort im Diagramm unterscheidbar sind, wird der Raum zwischen Temperatur- und Niederschlagskurve schraffiert oder gepunktet, je nachdem ob der Bereich humid oder arid ist. Damit extrem hohe Niederschlagsmengen das Diagramm nicht ins Kraut schiessen lassen, wird bei Niederschlagsmengen über 100 mm pro Monat (per-humider Bereich) die Niederschlagsachse auf ein Zehntel reduziert; dies wird im Diagramm dadurch gekennzeichnet, dass der Bereich dicht schraffiert oder vollständig ausgefüllt wird.

Das Diagramm ist vollständig, sobald der Rahmen gezeichnet ist; es kann dann mit Hilfe einer Hardcopy-Routine auf Papier ausgegeben werden. Hierzu wird ein grafikfähiger Drucker benötigt. Durch Drücken der RETURN-Taste oder der Leer-Taste erreicht man wieder das Hauptmenü.

Wurden Daten von der Tastatur eingegeben, so werden sie automatisch gespeichert und können anschliessend über den Programmteil «Daten von Disk» als Diagramm gezeichnet werden.

Programmbeschreibung

Das Programm zeichnet nach Eingabe der Niederschlags- und Tempe-

raturwerte ein Klimadiagramm nach Walter-Lieth in hochauflösender Grafik (320x200 Punkte) auf den Bildschirm. Von dort kann das Diagramm als Hardcopy ausgedruckt werden.

Alle neu eingegebenen Werte werden auf Diskette (aktuelles Laufwerk) gespeichert und können so später ohne erneute Tipparbeit genutzt werden. Als Filename empfielt sich der abgekürzte Stationsname und als Namenserweiterung das internationale Automobilkennzeichen zur leichten Einordnung der Dateien (Beispiel: ZUERICH.CH).

Das Programm ist in Turbo-Pascal geschrieben und auf allen PCs lauffähig, die nach dem Industriestandard arbeiten (MS-DOS 2.xx).

Variablenvereinbarung

Das Feld T enthält die zwölf Temperatur-Monatsmittel, T[14] die Jahresmitteltemperatur. T[13] wird als Zwischenspeicher benötigt; es enthält den Mittelwert der Januar- und Dezember-Temperaturen. T[13] dient zur Verstetigung der Kurve über den Jahreswechsel hinaus.

Das Array A enthält die original Niederschlagswerte, N dagegen die bereits für perhumide Bereiche reduzierten Zahlen.

Die Monatsnamen werden in einer Prozedur «Anfang» initialisiert; dies erleichtert die Anpassung an andere Sprachen.

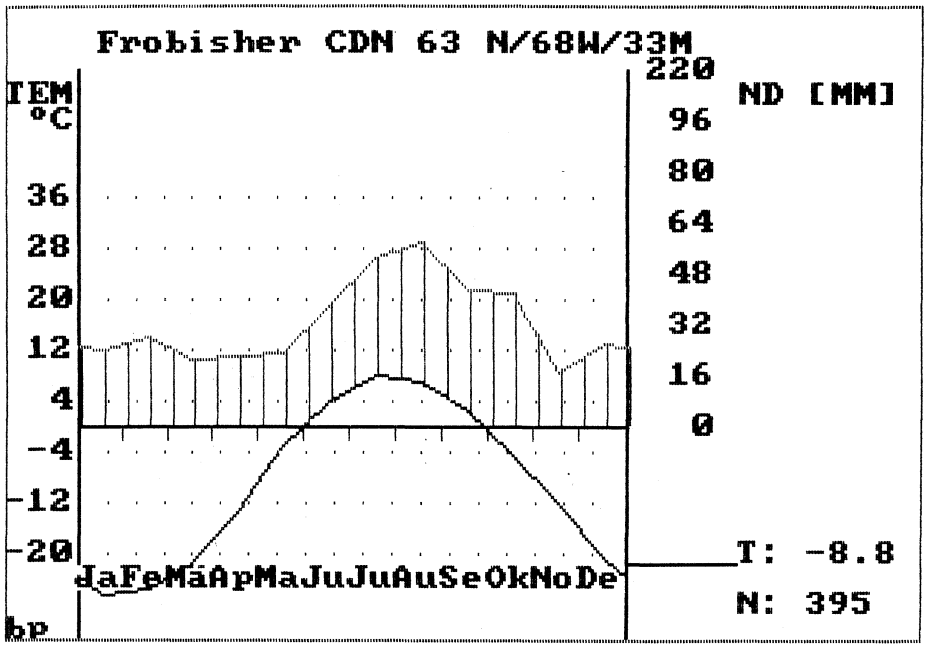
«LiesDaten» sucht nach Eingabe des Stationsnamens die Daten auf der Diskette und liest sie in die Felder A und T ein. Gleichzeitig wird das Feld N berechnet und die kleinste bzw. grösste Monatsmitteltemperatur bestimmt (MaxT und MinT).

Das Zentrum des Programmes stellt die Prozedur «DruckeDiagramm» dar. Zuerst muss noch ein Höhenfaktor bestimmt werden. Dieser dient dazu, das Diagramm bei sehr hohen Niederschlagswerten (z.B. Monsungebieten) in Richtung der Y-Achse zu reduzieren (F:=0.5, sonst F:=1). Danach wird die Nullinie in Abhängigkeit von der Minimaltemperatur bestimmt. Nur so ist es möglich, auch extreme Tieftemperaturen (z.B. kanadische NW-Territories) darzustellen. Das Minimum für die Nullinie liegt bei 8 Pixels entsprechend zwei Zeilen im Textformat.

Anschliessend werden die Achsen gezeichnet. Niederschlags- und Temperaturkurven und die Gesamtbeschriftung runden den ersten Prozedurteil ab.

In getrennten Schleifen werden die beiden Achsen beschriftet, wobei be-

```
    fn: Text;
    S,MinT,MaxT: Real;
    I:Integer;
    SName: Wort;
    OK:  Boolean;
begin
  Repeat
    WriteLn;
    WriteLn('Stationsname');
    ReadLn(SName);
    Assign(fn,SName);  (*prüfe, ob Station existiert*)
    {$I-} Reset (fn)   {$I+};
    OK:=(IoResult=0);
    Fehler:=(SName='');
    If Not OK Then WriteLn('Falsche Station');
  Until Ok  Or Fehler;

  If Not Fehler then  (*lies Temp und Nd*)
    begin
     S:=0;
     MA:=0;
     ReadLn(fn,SN);
     ReadLn(fn,HW);
     FOR I:=1 TO 12 Do
     begin
       ReadLn(fn,A[I]);
       N[I]:=A[I];
       S:=S+N[I];
       IF (N[I]>100) Then N[I]:=100+ ((N[I]-100)/10); (*auf 1/10 verkürzen*)
       IF (N[I]>MA) Then MA:=N[I];                    (*max Niederschlag*)
     end;
     N[14]:=S;          (*Jahresniederschlagsmenge*)
     S:=0;
     MT:=0;
     FOR I:=1 TO 12 Do
        begin
        ReadLn(fn,T[I]);
        IF (T[I]<MT) Then MT:=T[I];   (*Min Temp*)
        S:=S+T[I];
     end;
     T[14]:=INT(10*S/12+0.5)/10;          (*Jahresmittel der Temp*)
     CLOSE (fn);
     (*Min Max TempDiff*)
     MinT:=T[1];
     MaxT:=T[1];
     For I:=2 To 12 Do
             if t[i]>MaxT Then MaxT:=t[I] else If t[i]<MinT Then MinT:=t[i];
     MaxTempDiff:=MaxT-MinT;

  end; (*Notausgang*)
  end;
(*-------------------------------------------------------------*)

Procedure Interpoliere(X,X1,X2:Integer;Y1,Y2,Y3,Y4:Real; Var YT,YN:Integer);
begin
      YT:=trunc((Y2-Y1)/(x2-x1)*(X-x1)+Y1);
      IF (YT<0) THEN YT:=0;
      YN:=trunc(((Y4-Y3)/(X2-x1))*(x-x1)+Y3);
end;
(*-------------------------------------------------------------*)

Procedure Perhumid (B,X,DX,X1,X2,Nullinie:Integer;F1,Y1,Y2,Y3,Y4:Real);
Var X3,XInt,YT,YN:Integer;
        Begin
          (*Perhumid*)
          X3:=X;
          Repeat
            X3:=X3+2;
            XInt:=X3+16;
            Interpoliere(X3,X1,X2,Y1,Y2,Y3,Y4,YT,YN);
            IF (YN>100*F1) and (x3>6) THEN
                  Draw(XInt,Nullinie-trunc(100*F1),XInt,Nullinie-YN,1);
          Until (X3>X+dx-2) or (X3>B*12.5);
        end; (*Perhumid*)
(*-------------------------------------------------------------*)

procedure Schraffur(Var N,T:RealFeld;B,NL:Integer;F1,F2:Real);
    Const DX=8;
    Var X1,X2,X3,X,I,J,K,XInt,YT,YN,YH,Nullinie: Integer;
        y1,y2,y3,y4: Real;
    Begin
    N[13]:=(N[1]+N[12])/2;
    N[0]:=N[13];
```




```
    T[13]:=(T[1]+T[12])/2;
    T[0]:=T[13];
    Nullinie:=180-NL;
    For I:=1 TO 13 dO
    begin
      X1:=(I-1)*B; (*linke ecke*)
      X2:=I*B;     (*rechte ecke*)
      Y1:=T[I-1]*F2;
      Y2:=T[I]*F2;
      Y3:=N[I-1]*F1;
      Y4:=N[I]*F1;
      X:=x1;
      While X<X2 do
      Begin
        IF ((N[I-1]>100) or (N[I]>100))  THEN
              PerHumid(B,X,DX,X1,X2,Nullinie,F1,Y1,Y2,Y3,Y4);
        Interpoliere(X,X1,X2,Y1,Y2,Y3,Y4,YT,YN);
        IF (X<B*12+10) AND (X>=8)  THEN
        begin
          XInt:=X+18;
          If YT<YN Then
          Begin
            Draw(XInt,Nullinie-YT-1,XInt,Nullinie-YN,1);  (*humid*)
          End
          Else
          Begin    (*arid*)
            YH:=trunc(Nullinie-YT-3);
            Repeat
              YH:= YH+3;
              Plot (XInt,YH,1);
            Until YH>=Nullinie-YN-2;
          end; (*else*)
        end; (*If X...*)
        X:=X+DX;
      end; (*While*)
    end; (*I*)

    DRaw (0,0,0,199,1);
    Draw (0,199,319,199,1);
    Draw (319,199,319,0,1);
    Draw (319,0,0,0,1);  (*Rahmen*)
  end; (*Proc Schraffur*)
(*-------------------------------------------------------*)

Procedure DruckeDiagramm(MA,MT:Real; M:Monate; N,T:Realfeld;Sn,HW:Wort);
const B=16; (*Breite*)
Var F1,F2,H: real;
    X,Y,DY,I,NL: Integer;
Begin
ClrScr;
GraphColorMode;
F1:=1;
IF (MA>400) THEN F1:=0.5 ;  (*verkürze, falls Max>400*)
F2:=F1*2; (*Höhenfaktor T*)
```

sonders auf die bei Werten über 100 mm/Monat verkürzten Bereich Wert gelegt wird.

Am Ende der Prozedur «Drucke Diagramm» wird die Prozedur «Schraffur» aufgerufen, die ihrerseits wiederum die beiden Prozeduren «Interpoliere» und «Perhumid» aufruft.

«Interpoliere» berechnet die Y-Koordinate der Temperatur- oder Niederschlagskurve zwischen zwei zu den Monatsmitteln gehörenden X-Werten.

«Perhumid» schraffiert den über 100 mm gelegenen Bereich des Diagramms sehr dicht. Auf eine Füllung dieses Bereiches wurde zugunsten der Verarbeitungsgeschwindigkeit verzichtet.

Neue Daten werden über die Tastatur mit Hilfe der Prozedur «Daten VonTast» eingegeben (DatenLesen) und nach einer kurzen Ueberprüfung des Stationnamens auf Diskette geschrieben (DatenSchreiben). Ein kurzes Menü (Auswahl) und das Hauptprogramm beschliessen das Programm. □

Computer-Splitter

1 Million PS/2 ausgeliefert

Die IBM hat seit der Ankündigung des IBM Personal Systems/2 vor rund einem halben Jahr weltweit bereits eine Million Modelle dieser neuen Personal Computer-Familie ausgeliefert. Dieses Resultat gab die IBM an der COMDEX bekannt, einer Handelsmesse für Mikrocomputer, die kürzlich in Las Vegas stattfand. Danach liefert die IBM derzeit weltweit täglich rund 9'000 Personal System/2 aus. Bei den ersten PC-Modellen der IBM waren bis zur Auslieferung von 1 Million Stück 28 Monate verstrichen. Info: IBM Schweiz, General Guisan-Quai 26, 8002 Zürich, Tel. 01/ 207'21'11. □

Büfa Romande 1988

(546/fp) Ein klarer Rekord für die Büfa im Basler Messeherbst 1987. Die Branche bereitet sich schon für die Büfa Romande in der Westschweiz vor, welche die Basler Büfa im Zweijahres-Turnus ablösen wird und vom 4. bis zum 8. Oktober 1988 im Palexpo in Genf stattfinden wird. In Basel wird statt der Büfa die Swissdata stattfinden. □

```
NL:=trunc(8*INT(ABS(MT)*F2/8)); (*Null-Linie*)
IF (NL<=8) THEN NL:=8;          (*minimale Null-Linie*)
Draw(25,20,25,199,3);           (*Achsen*)
Draw(B*12+25,20,B*12+25,199,3);
Draw(25,180-NL,12*B+25,180-NL,3);
Draw(12*B+25,175,270,175,3);
GotoXY(1,25);
Write('bp');
FOR I:=1 TO 11 Do Draw(I*B+24,180-NL,I*B+24,184-NL,1);     (*Monatsabgrenzung*)
FOR I:=1 TO 12 Do (*beschrifte X-Achse*)
Begin
  GotoXY(trunc(I*B/8+2),23);
  Write(copy (M[i],1,2));
end; (*For*)
(*drucke Niederschlag*)
H:=(N[1]+N[12])/2;
Draw(12*B+25,180-trunc(H*F1)-NL,trunc(11.5*B)+25,180-trunc(F1*N[12])-NL,1);
FOR I:=11 DownTO 1 Do
       Draw(trunc((I-0.5)*B+25),trunc(180- F1*N[I]-NL),trunc((I+0.5)*B+25),
       trunc(180-F1*N[I+1]-NL),1);
Draw(25,trunc(180-F1*H-NL),trunc(0.5*B+25),trunc(180-N[1]*F1-NL),1);

(*Drucke Temp*)
H:=(T[1]+T[12])/2;
Draw(25,trunc(180-H*F2-NL), trunc(0.5*B+25),trunc(180-F2*T[1]-NL),2);
FOR I:=11 DownTO 1 Do
       Draw(trunc((I-0.5)*B+25),trunc(180- F2*T[I]-NL), trunc((I+0.5)*B+25),
       trunc(180-F2*T[I+1]-NL),2);
Draw(12*B+25,trunc(180-H*F2-NL),trunc(11.5*B+25),trunc(180-F2*T[12]-NL),2);

(*Beschriftung*)
GotoXY (5,2); Write(SN+' '+HW);
GotoXY(33,22);Write('T:',T[14]:5:1);  (*Jahresmittel*)
GotoXY(33,24);Write('N:',N[14]:4:0);  (*Jahresniederschlagssumme*)

(*Masstab Temp*)
I:=6;
Repeat
  I:=I+2;
  GotoXY(1,I);
  X:=trunc((8*(23-I)-NL)/F2);
  If X<50 Then Write(X:3);
  X:=25;
  Repeat  (*Hilfslinie punktieren*)
    X:=X+10;
    Plot (X,I*8-4,1);
  until X>=12*B+10;

Until I>=22;

(*Masstab  ND*)
X :=trunc(23-NL/F1/8) ;
DY:=trunc(8/F1);
Y:=0;
WHILE (Y<=96) AND (X>4) Do
  begin
  GotoXY(29,X);
  Write(Y:3);
  X:=X-2;
  Y:=Y+2*DY;
END; (*While*)
I:= X+2;   (*letzte Zeile*)
Y:=trunc(100+12*10/F1); (*nächster Wert > 100 mm *)

Repeat  (*perhumider Bereich*)


   I:=I-2;
   GotoXY(29,I);
   Write(Y:3);
   Y:=Y+20*DY;
Until I<=4;

(*Titelei*)
GotoXY(33,4);
WriteLn('ND [MM]');
GotoXY(1,4);
WriteLn('TEM');
GotoXY(2,5);
WriteLn(CHR(248)+'C');
Schraffur(N,T,B,NL,F1,F2);
end; (*procedure diagramm*)
(*-------------------------------------------------*)
```

```
Procedure DatenVonDisk;
Var MA,MT,MaxTempDiff: Real;
    A,T,N:RealFeld;
    SN,HW:Wort;
    Fehler: boolean;
Begin
  LiesDaten(MA,MT,MaxTempDiff,A,T,N,SN,HW,Fehler);
  If Not Fehler Then
    begin
    DruckeDiagramm(MA,MT, M, N,T,SN,HW);
    WarteSchleife;
    ClrScr;
    TextMode(C80);
  end; (*Not Fehler*)

END;
(*---------------Ende Procedure DatenVonDisk----------------------*)

procedure DatenLesen(Var T,N:RealFeld;M:Monate);
Var I: Integer;
begin
ClrScr;
WriteLN('Hinweise, z.B. Lage im Gradnetz:');
ReadLn(HW);

WriteLn('Geben Sie nun die Niederschläge ein.');
For i:=1 to 12 Do
   begin
   Repeat
     Write(M[i]);
     ReadLn(N[I]);
   Until (N[I]<1000) and (N[i]>=0);  (*Plausibilitätsprüfung*)
   end;

WriteLn('Geben Sie nun die Temperaturen ein.');
 For i:=1 to 12 Do
   begin
   Repeat
     Write(M[i]);
     ReadLn(T[I]);
   Until (t[I]<75) and (t[i]>-75);    (*Plausibilitätsprüfung*)
   end;
end;
(*-----------------------------------------------------*)

procedure DatenSchreiben(Var Fn:Text;SN,HW:Wort;N,T:RealFeld);
Var I:Integer;
begin
    ReWrite(fn);
    WriteLn(fn,SN);
    WriteLn(fn,HW);
    FOR I:=1 TO 12 Do  WriteLn(fn,N[I]);
    For I:=1 to 12 Do  WriteLn(fn,T[I]);
    CLOSE (fn);
end;
(*-----------------------------------------------*)

Procedure DatenVonTast(M:Monate);
Var T,N:RealFeld;
    SN,HW:Wort;
    I: Integer;
    OK,Fehler: Boolean;
    SName: Wort;
    FN:Text;
Begin
  Repeat
    WriteLn;
    WriteLn('Stationsname:');
    ReadLn(SName);
    Assign(fn,SName);
    {$I-} Reset (fn)   {$I+};
    OK:=(IoResult=0);
    Fehler:=(SName='');
    If OK Then
      begin
       WriteLn('Station existiert bereits');
      End; (*If*)
  Until not Ok or Fehler;
  If (not Ok) and (not Fehler) Then
    begin
      SN:= SName;
      DatenLesen(T,N,M);;
```

Computer-Splitter

Bildverarbeitungskarte

LFP-Baby ist ein Bildspeicher/Verarbeitungssystem, realisiert als Einsteckkarte für PCs vom Typ IBM AT oder kompatible. In Echtzeit können Videobilder digitalisiert und in den Bildfomaten 512x512x8 Bit oder 512x256x8 Bit abgespeichert werden. Mittels einer 2Kx8Bit Look-up-Table ist es möglich, die Bilder vor der Speicherung in Echtzeit mit beliebigen Funktionen luminanzmässig zu verändern (z.B. Binarisierung). Um eine möglichst leistungsfähige Bildverarbeitung zu ermöglichen, kann ein beliebiger Bildausschnitt vom Format 256x256x8 (64 KByte) in den MS-DOS-Speicherbereich des IBM AT gemappt werden. Somit entfällt die zeitraubende Datenübertragung zwischen Bildspeicher und Arbeitsspeicher des PC. Dem Bildspeicher ist ein zusätzlicher Overlay-Speicher vom Format 512x512x4 Bit bzw. 512x256x4 Bit überlagert, der ebenfalls in den MS-DOS-Speicherbereich des PC gemappt werden kann. Ein programmierbarer Farbausgang (RGB; 3x8 Bit DAC) erlaubt somit die Darstellung der gespeicherten Bilder in schwarz/weiss oder Falschfarbe und deren partielle Ueberblendung mit verschiedenfarbigem Text oder Grafik, ohne den Bildinhalt zu zerstören. Je vier digitale, optisch getrennte Steuereingänge/ausgänge ermöglichen sofortige, systemsynchronisierte Aktionen und Reaktionen zur Aussenwelt. Info: Leutron AG, Kanalstrasse 15, 8152 Glattbrugg, Tel. 01/810'06'76. □

Gut gedruckt...

sind die Outputs des Schrift- und Grafik-Programmpaketes «Headline». Das Produkt arbeitet mit XT- und AT-Geräten und erzeugt hochwertige Ausdrucke auf grafikfähigen Matrix- und Laser-Druckern. Das Paket umfasst standardmässig über 50 Fonts mit verschiedenen Schriften, Sonderzeichen und Barcodes. Ein integrierter Graphic-Editor gestattet dem Benutzer den Entwurf eigener Zeichen und Schriften (Sonderzeichen, Embleme, Firmenzeichen usw...). Durch Verwendung virtueller Speicher-Techniken können Outputs in fast jeder beliebigen Grösse angefertigt werden: Von winzigen Etiketten in

Grösse einer Briefmarke bis zum Posterformat und meterlangen Spruchbändern. Dabei werden nur 256 KByte Hauptspeicher belegt, so dass «Headline» auch von anderen Programmen aus aufgerufen werden kann. Hierdurch ergeben sich neue Möglichkeiten wie: Formular-Druck, automatisierter Druck von Etiketten, Barcode-Anwendungen, Plots u.a. Durch eine Reihe von Zeichen-Modifikationen entstehen aus den Standard-Schriften hunderte weiterer neuer Schriftarten, die alle in variabler Grösse darstellbar sind. Die Zeichen-Auflösung liegt mit 240x240 Pixeln weit über dem der üblichen NLQ-Schriften (24x36, 18x24...). Einsatzgebiete von «Headline» sind der Etiketten-Druck, Schilder, gut lesbare, grosse Beschriftungen, Formular-Druck, Ersatz von Anreibe-Symbolen und dergleichen. Die integrierte Plotter-Emulation bietet Linien-Grafik in unterschiedlichen Strich-Stärken und Arten, Inversionen und frei definierbare grafische Muster können in freier Kombination mit Texten verwendet werden. Geschrieben werden kann in allen vier Himmelsrichtungen. Info: Ing.-Büro Wilke, Adalbertsteinweg 26, D-5100 Aachen. □

```
      DatenSchreiben(Fn,SN,HW,N,T);
    end; (*If*)
End;
(*-----------------------------------------------*)

procedure Auswahl(var t1:integer);
begin
   ClrScr;
   GotoXY(1,10);
   WriteLn('Bitte wählen Sie:');
   WriteLn;
   WriteLn('1:  Daten von Diskette');
   WriteLn('2:  Daten von Tastatur');
   WriteLn('3:  Programm beenden');
   t1:=0;
   Repeat
      ReadLn(t1);
   Until (t1>0) and (t1<4);
end;
(*===============================================*)
(*Hauptprogramm*)
begin
repeat
   ClrScr;
   NormVideo;
   GotoXY(10,5);
   WriteLn('Klimadiagramm');
   GotoXY(10,7);
   WriteLn('Ein Programm von Bruno Pohl, 1987');
   WarteSchleife;
   Auswahl(t1);
   case t1 of
   1: DatenVonDisk;
   2: DatenVonTast(M);
   end; (*case*)
 until t1=3;
ClrScr;
NormVideo;
END.
```

Lineare Mehrfach-Regression und -korrelation

Die Messwerte x_i, y_i eines linear angenommenen Zusammenhanges liegen meist nicht exakt auf einer Geraden. Da sie Fremdeinflüsse, statistische Schwankungen, Messfehler und dergleichen enthalten, «streuen» sie mehr oder weniger. Das Punktfeld lässt sich durch eine Regressionsgerade x=mx+t ausgleichen (Bilder 3,4); ihre Parameter m und t werden aus den Punktkoordinaten nach dem Prinzip der kleinsten Quadratsumme berechnet. Zum Auffinden von Zusammenhängen eignen sich besonders Mehrfachkorrelationen. Dieses Programm ist für den C128 mit Plotter VC 1520 geschrieben, man kann es aber relativ leicht auf andere Modelle mit Grafik und genügend verbleibendem Speicherplatz übersetzen.

```
LINEARE MEHRFACH-KORRELATION :

ANZAHL VARIABLEN =? 5
```

NR.	X1	X2	X3	X4	X5
1	? 70	? 5.5	? 4.0	? 17.7	? 10.5
2	? 89	? 5.5	? 4.7	? 20.4	? 12.6
3	? 52	? 2.6	? 2.0	? 4.2	? 5.8
4	? 64	? 2.1	? 2.1	? 3.5	? 5.9
5	? 116	? 5.3	? 3.8	? 16.3	? 8.9
6	? 49	? 4.2	? 3.2	? 10.8	? 10.2
7	? 190	? 6.8	? 5.0	? 27.3	? 16.8
8	? 150	? 5.8	? 4.0	? 18.9	? 15.0
9	? 130	? 5.9	? 4.7	? 22.1	? 13.9
10	? 215	? 7.2	? 6.8	? 38.5	? 19.8
11	? E				

Bild 1: Eingabemaske für n=5. Zur Erhöhung der Uebersichtlichkeit sind die Fragezeichen auf dem Bildschirm unsichtbar (Hintergrundfarbe).

NR	REGRESSIONS-GLEICHUNG		KORR.KOEFF.
0	X(2)= 0.023X(1)+2.462		0.820
1	X(3)= 0.020X(1)+1.740	G	0.837
2	X(4)= 0.160X(1)-0.043	P	0.887
3	X(5)= 0.070X(1)+4.068		0.893
4	X(3)= 0.844X(2)-0.266	P	0.934
5	X(4)= 6.653X(2)-15.89		0.944
6	X(5)= 2.940X(2)-3.023		0.917
7	X(4)= 7.436X(3)-12.00	G	0.992
8	X(5)= 3.405X(3)-1.784		0.935
9	X(5)= 0.422X(4)+4.365	G	0.952

```
MAX. KORR.KOEFF. = R(4,3) =  0.992

WELCHE KOMBINATION WILLST DU (0-9) ? 4

PLOTTEN ODER ANDERE GRAFIK (P/G) ? P
```

Bild 2: Bildschirmdarstellung der Regressionsgleichungen und Korrelationskoeffizienten mit Fragemenü. Die m, k und r werden auf vier Stellen angegeben, unabhängig von der Kommaposition.

Während vielen Publikationen, praktisch von der Annahmen ausgehen, die Abweichungen seien nur in den y_i zu suchen, bestehen in der Regel Ungenauigkeiten auch in den x_i. Anstelle der Bedingung

$$\sum_{i=1}^{n} (\Delta y_i)^2 = \text{Minimum}$$

soll die Regressionsgleichung die Bedingung

$$\sum_{i=1}^{n} [(\Delta x_i)^2 + (\Delta y_i)^2] = \text{Minimum}$$

erfüllen, worin gilt:

$\Delta x_i = x_i - \bar{x}$,

$\bar{x} = \sum_{i=1}^{n} x_i / n$, $\Delta y_i = y_i - \bar{y}$,

$\bar{y} = \sum_{i=1}^{n} y_i / n$.

Anstelle der Formel

$$m = (\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}) / (\sum_{i=1}^{n} x_i^2 - n\bar{x}^2)$$

folgt entsprechend [1] der genauere Wert

$$m = \tan[0.5 \cdot \arctan (2 \sum_{i=1}^{n} \Delta x_i \Delta y_i / (\sum_{i=1}^{n} \Delta x_i^2 - \sum_{i=1}^{n} \Delta y_i^2))].$$

Für den Korrelationskoeffizienten gilt die allgem ein anerkannte Formel

$$r = \sum_{i=1}^{n} \Delta x_i \Delta y_i / \sqrt{(\sum_{i=1}^{n} \Delta x_i^2)(\sum_{i=1}^{n} \Delta y_i^2)}.$$

Anwendungen gibt es beim Eichen von Messgeräten; aber auch in Biologie, Psychologie, Statistik, in Wirtschafts-, Geschichts- und Sozialwissenschaften berechnet man Regressionsgleichungen und Korrelationskoeffizienten, um die Parameter und die Zuverlässigkeit linearer Zusammenhänge zu ermitteln.

Christoph Reuss

Eine deutliche Arbeitsersparnis bietet die Möglichkeit, Mehrfachkorrelationen - hier bis zu fünf - einzugeben und simultan zu rechnen: Sind Beziehungen zwischen den Messdaten von mehr als zwei, z.B. n=5 Variablen zu vermuten, dann sind n*(n-1)/2=10 Zweierkombinationen möglich. Müsste man sie einzeln zu je zwei Kolonnen eingeben, so wären 20 Kolonnen einzutippen. Mit der Option, die Kombinationen mehrerer Kolonnen gleichzeitig zu rechnen, sind bei n=5 nur fünf Kolonnen einzugeben; anstelle von x und y wird x_1, x_2, ... x_n verwendet. In den resultierenden Regressionsgleichungen $x_i = m_{i,k} * x_k + t_{i,k}$ (Bild 2) übernimmt y jeweils die Rolle des x mit dem grösseren Index (i>k).

Eine nicht sehr typische, aber anschauliche Anwendung einer Mehrfachkorrelation ist die zwischen Merkmalen einer Stengelpflanze, bei der wir z.B. folgende fünf Variablen wählen können: x_1 = Alter [Tage], x_2 = Stengeldicke am Boden [mm], x_3 = Stengelhöhe [cm], x_4 = durchschnittliche Blattlänge oben, x_5 = Fruchtgewicht [g]. Um hinreichend genaue Korrelationen zu erhalten, wird man diese Daten an mindestens zehn Pflanzen messen und eingeben müssen.

Bei einer Erweiterung auf n>5 liesse sich der Daten-I/O nach entsprechender Programmänderung nur noch auf dem Drucker darstellen (ab n=7 haben die Gleichungen nicht mehr genug Platz auf dem Bildschirm); anstelle der bisherigen INPUT-Abfrage empfiehlt sich dann die Ablage der Eingabedaten in Datenzeilen. Damit liessen sich z.B. die Ursachen des Waldsterbens und allfällige Zusammenhänge zwischen ihnen genauer analysieren.

Ein weiteres Beispiel wäre der Zusammenhang zwischen Beträgen der an verschiedenen Stellen einer Grosstadt gleichzeitig gemessenen Verkehrsdichte. An den Korrelationskoeffizienten (Bild 2) ist abzulesen, welche Variablenpaare deutlich korrelieren (0.9<|r|<1), und welche wahrscheinlich nicht zusammengehören (|r|<0.5).

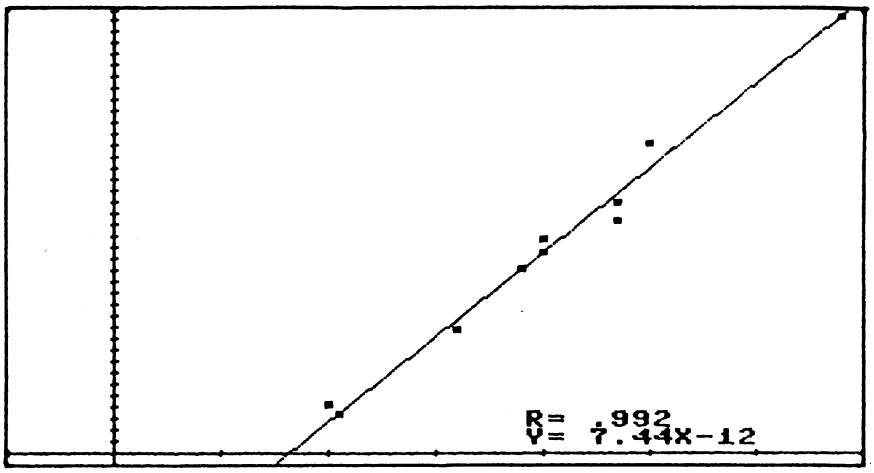


Bild 3: Grafikdarstellung der Kombination (4,3).

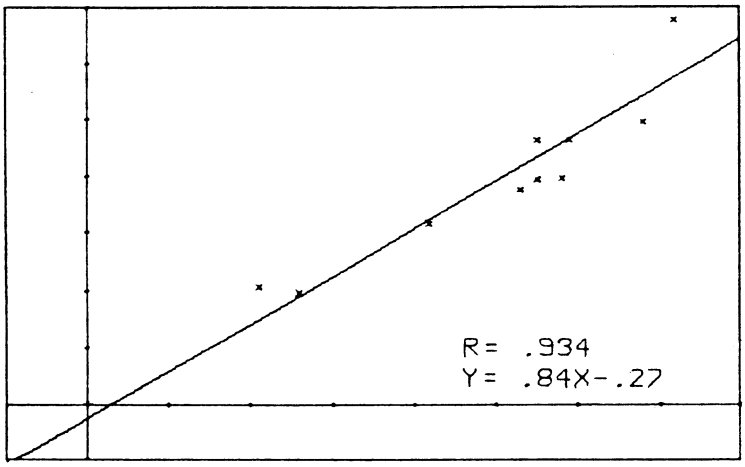


Bild 4: Plotdarstellung der Kombination (3,2).

Programmablauf

Zuerst wird gefragt, zwischen wievielen Variablen korreliert werden soll (2..5). Dann erscheint die Bildmaske, in welche die Punktkoordinaten eingegeben werden (Bild 1). Zur Erhöhung der Uebersichtlichkeit haben die INPUT-Fragezeichen die Farbe des Hintergrundes, sind also «unsichtbar». Gewertet werden die ersten acht Zeichen (vier Vor-, drei Nachkommastellen), aus Platzmangel werden die Zahlen im Bild nach der 4. Stelle abgeschnitten. Nach einer allfälligen Fehleingabe lässt sich mit «K» <RETUREN> der Cursor zwecks Korrektur je eine Stelle nach oben (in der x_1-Spalte) zurückversetzen. Die korrekten Eingaben können dann mit <RETURN> bestätigt, die falschen korrigiert werden. Das Ende der Eingaben (max. 100 Punkte) signalisiert man entweder mit «E» oder mit «D», was zur Berechnung der m, t und r aller Kombinationen führt. Bei «E» erscheinen danach die Regressionsgleichungen und Korrelationskoeffizienten aller Kombinationen auf

```
10 PRINT"■":CLR:DIMZ(100,5),DX(100,5),K(5,5),M(5,5),T(5,5),X(100),Y(100)
20 FORI=0TO9:A$(I)=CHR$(I+48):NEXTI
30 COLOR0,2:COLOR4,2
40 PRINT"■■   LINEARE MEHRFACH-KORRELATION :"
50 PRINT"                                    "
60 INPUT"■■■ANZAHL VARIABLEN =";AV:IFAV<2ORAV>5THEN60
70 PRINT"■──";:FORI=1TOAV:PRINT"┬──────";:NEXTI:PRINT"┐"
80 PRINT"■NR.■|";
90 FORI=1TOAV:PRINT"  ■X"A$(I)"  ■|";:NEXTI:PRINT
100 PRINT"■──";:FORI=1TOAV:PRINT"┼──────";:NEXTI:PRINT"┤"
110 FORO=1TO100:GOTO130
120 O=O-1:PRINT"□□";
130 PRINTUSING"■###■|■";O;
140 FOROO=1TOAV:POKE208,1:POKE842,31
150 Z$="":INPUTZ$:Z(O,OO)=VAL(LEFT$(Z$,8))
160 IFO>1ANDLEFT$(Z$,1)="K"THEN120
170 IFO>2ANDOO=1ANDLEFT$(Z$,1)="E"THEN260
180 IFLEFT$(Z$,1)="D"THEN210
190 PRINTTAB(3+OO*7)"□■|■";
200 NEXTOO:PRINT:NEXTO
210 OPEN4,4:CMD4:PRINT"   NR. |";:FORI=1TOAV:PRINT"      X"A$(I)"      |";:NEXTI
220 PRINT:PRINT"──────┼";:FORI=1TOAV:PRINT"──────────┼";:NEXTI:PRINT
230 FORI=1TOO-1:PRINTUSING"  ###  |";I;:FORJ=1TOAV:PRINTUSING"  ####.###  |";Z(I
,J);:NEXTJ:PRINT:NEXTI
240 PRINT"──────┴";:FORI=1TOAV:PRINT"──────────┴";:NEXTI:PRINT:PRINT:PRINT
250 PRINT#4:CLOSE4
260 FAST:O=O-1:OO=AV
270 D$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■":L$="                              "
280 FORAI=1TO OO :FORN=1TO O :A=A+Z(N,AI):NEXTN:A(AI)=A/O:A=0:NEXTAI
290 FORAI=1TO OO :FORN=1TO O :DX(N,AI)=A(AI)-Z(N,AI):EI=EI+DX(N,AI)*DX(N,AI):NEX
TN:E(AI)=EI:EI=0:NEXTAI
300 FORAI=1TOOO-1
310 FORAK=AI+1TOOO
320 FORN=1TOO:IFAI=AKTHEN340
330 FA=FA+DX(N,AI)*DX(N,AK)
340 NEXTN
350 K(AI,AK)=FA/SQR(E(AI)*E(AK)):M(AI,AK)=ATN(2*FA/(E(AI)-E(AK)+1E-5)):FA=0
360 IFE(AI)<E(AK)THENM(AI,AK)=TAN(1.57+M(AI,AK))/2):ELSEM(AI,AK)=TAN(M(AI,AK)/2)
370 T(AI,AK)=A(AK)-A(AI)*M(AI,AK)
380 NEXTAK,AI
390 PRINT"■■■NR■■| ■REGRESSIONS-GLEICHUNG ■■| |■KORR.KOEFF."
400 PRINT"■■┼──────────────────┼┼──────────"
```

```
410 Z=-1:FORAI=1TO00-1
420 FORAK=AI+1TO00
430 Z=Z+1:AI(Z)=AI:AK(Z)=AK:IFABS(K(AI,AK))>ABS(MAX)THENMAX=K(AI,AK):AA=AI:KK=AK
440 GA$="######":PRINT"■"A$(Z)" ■| ■X("A$(AK)")=";
450 L=LEN(STR$(INT(M(AI,AK)))):U$=LEFT$(GA$,L)+"."+LEFT$(GA$,-(5-L)*(L<6))
460 PRINTUSINGU$;M(AI,AK);
470 PRINT"X("A$(AI)")";
480 L=LEN(STR$(INT(T(AI,AK)))):U$="+"+LEFT$(GA$,L-1)+"."+LEFT$(GA$,-(5-L)*(L<6))
490 PRINTUSINGU$;T(AI,AK);
500 PRINT" ■| | ■";:PRINTUSING"##.###";K(AI,AK)
510 NEXTAK,AI
520 PRINT"■┼──────────────────────────┼┼──────────":SLOW
530 PRINT"■■   MAX. KORR.KOEFF. = R("A$(KK)","A$(AA)") = ";
540 PRINTUSING"##.###";MAX
550 IFLEFT$(Z$,1)="D"THENZ$="":OPEN4,4:CMD4:GOTO390
560 IFZ$=""THENPRINT:PRINT:PRINT:PRINT#4:CLOSE4
```

Listing 1: Bildschirm-I/O, mit optionaler Druckausgabe (Hauptteil)

```
570 COLOR4,2:PRINT"■■ WELCHE KOMBINATION WILLST DU (0-"A$(Z)") ";:INPUTGG
580 IFGG<0ORGG>ZTHENPRINT"□□□□":GOTO570
590 FAST:AI=AI(GG):AK=AK(GG)
600 VX=0:VY=0:BX=0:BY=0:U=0
610 FORI=1TOO:X(I)=Z(I,AI):Y(I)=Z(I,AK):NEXT
620 M=M(AI,AK):K=K(AI,AK):T=T(AI,AK)
630 FORI=1TOO
640 IFX(I)<VXTHENVX=X(I)
650 IFX(I)>BXTHENBX=X(I)
660 IFY(I)<VYTHENVY=Y(I)
670 IFY(I)>BYTHENBY=Y(I)
680 NEXTI
690 XU=INT((BX-VX)/50+1)
700 YU=INT((BY-VY)/50+1)
710 VX=INT(ABS(VX)+.1)*SGN(VX)-XU
720 BX=INT(ABS(BX)+.1)*SGN(BX)+XU
730 VY=INT(ABS(VY)+.1)*SGN(VY)-YU
740 BY=INT(ABS(BY)+.1)*SGN(BY)+YU
750 GRAPHIC1,1:COLOR1,1:COLOR4,14
760 SX=BX-VX:SY=BY-VY:FX=319.5:FY=199.5
770 XE=ABS(FX/SX): XA=-VX*XE
780 YE=ABS(FY/SY): YA=FY+YE*VY :SLOW
790 DRAW1,0,YATO320,YA:IFSX>100THEN810
800 FORI=0TO320STEPXE:DRAW1,I,YA-1TOI,YA+1:NEXT
810 DRAW1,XA,0TOXA,200:IFSY>100THEN830
820 FORI=0TO200STEPYE:DRAW1,XA-1,ITOXA+1,I:NEXT
830 FORI=1TOO:LX=XA+XE*X(I):LY=YA-YE*Y(I):BOX1,LX-1,LY-1,LX+1,LY+1,,1:NEXT
840 A=YA-YE*(M*VX+T): B=XA+XE*(VY-T)/M
850 C=YA-YE*(M*BX+T): D=XA+XE*(BY-T)/M
860 IFA>=0ANDA<200THENMM(U)=0:NN(U)=A:U=U+1
870 IFB>=0ANDB<320THENMM(U)=B:NN(U)=FY:IFU=0THENU=U+1
880 IFC>=0ANDC<200THENMM(U)=FX:NN(U)=C:IFU=0THENU=U+1
890 IFD>=0ANDD<320THENMM(U)=D:NN(U)=0:IFU=0THENU=U+1
900 DRAW1,MM(0),NN(0)TOMM(1),NN(1)
910 K$=STR$(INT(K*1000+0.5)/1000)
920 M$=STR$(INT(M*100+0.5)/100)
930 T$=STR$(INT(T*100+0.5)/100):IFVAL(M$)=1THENM$=" "
940 IFVAL(M$)=-1THENM$="-"
950 IFM$=" 0"THENM$="":X$="":ELSEX$="X"
960 IFT>-.009ANDM$+X$=""THENT$="+"+RIGHT$(T$,LEN(T$)-1):IFT$="+0"THENT$=""
970 Y$="Y="+M$+X$+T$:W=2:IFM>0THENW=24
980 CHAR1,W,22,"R="+K$:CHAR1,W,23,Y$:W=8*(W+LEN(K$))+20
990 PRINT"■■"LEFT$(D$,GG+3)TAB(27)"■G■"LEFT$(D$,Z+10)
1000 GETKEYA$:GRAPHIC0:COLOR4,2
1010 PRINT"□□□□":GOTO570
```

Listing 2: Grafikteil (fakultativ)

```
1010 INPUT"■■ PLOTTEN ODER ANDERE GRAFIK (P/G) ";A$
1020 IFA$="P"THEN1040
1030 PRINT"□□□□□□□":GOTO570
1040 GRAPHIC1:COLOR4,14:V=299:F=1.5
1050 IFWU=0THENWU=1:OPEN1,6,1:OPEN2,6,2:OPEN3,6,3:OPEN10,6,0:PRINT#3,1
1060 PRINT#2,3:PRINT#3,1:FORI=1TO17:PRINT#10:NEXT:PRINT#1,"I",0,0
1070 XA=XA*F:YA=V-YA*F:XE=XE*F:YE=YE*F
1080 PRINT#1,"M",0,0:PRINT#1,"D",0,V:PRINT#1,"D",479,V:PRINT#1,"D",479,0
1090 PRINT#1,"D",0,0:PRINT#2,0
1100 PRINT#1,"M",0,YA:PRINT#1,"D",479,YA
1110 IFSX>160THEN1130
1120 FORI=0TO480STEPXE:PRINT#1,"M",I,YA+1:PRINT#1,"D",I,YA-1:NEXTI
1130 PRINT#1,"M",XA,V:PRINT#1,"D",XA,0
1140 IFSY>100THEN1160
1150 FORI=0TO300STEPYE:PRINT#1,"M",XA-1,V-I:PRINT#1,"D",XA+1,V-I:NEXTI
1160 PRINT#2,2:FORI=1TOO:LX=XA+XE*X(I):LY=V-YA-YE*Y(I)
1170 PRINT#1,"M",LX-2,V-LY+2:PRINT#1,"D",LX+2,V-LY-2:PRINT#1,"M",LX-2,V-LY-2:PRI
NT#1,"D",LX+2,V-LY+2:NEXTI
1180 PRINT#2,1:PRINT#1,"M",F*MM(0),V-F*NN(0):PRINT#1,"D",F*MM(1),V-F*NN(1)
1190 PRINT#1,"M",0,-50
1200 PRINT#2,1:X$="R="+K$
1210 PRINT#10,LEFT$(L$,20-LEN(X$)/2)X$
1220 PRINT#10,LEFT$(L$,20-LEN(Y$)/2)Y$
1230 PRINT"■■"LEFT$(D$,GG+3)TAB(27)"■P■"LEFT$(D$,Z+7)
1240 GRAPHIC0:GOTO510
```

Listing 3: Plotteil (fakultativ)

dem Bildschirm (Bild 2); bei «D» werden sie zusätzlich, zusammen mit der Eingabeliste, als Dokumentation ausgedruckt. Nun kann man wählen, welche Kombination man sich als Grafik anschauen will.

Nach Betrachtung der Grafik (Bild 3) kommt man mit einer beliebigen Taste in die Option «Plotten oder andere Grafik (P/G) ?», vorausgesetzt, man hat den Plotteil (Listing 3) mit eingetippt. Nach Ausführung der Option wird je nach Antwort ein G oder ein P in die schmale Spalte geschrieben, damit jederzeit sichtbar ist, welche Kombinationen schon gezeigt bzw. geplottet wurden. Nach und nach lassen sich so alle «guten» Korrelationen ausplotten (Bild 4).

Grafik- und Plotteil sind so programmiert, dass die Zahlenbereiche den Bildbereich voll ausfüllen. Das Bild enthält dabei nebst dem Punktfeld immer das Achsenkreuz. Auf diesem wird die Skalierung weggelassen, wenn die Koordinatenbereiche hundert Einheiten übersteigen. Die ausgleichende =Regressionsgerade verbindet ihre jeweils auf den Bildrand berechneten Endpunkte. Regressionsgleichung und Korrelationskoeffizient werden bei der Grafik je nach Vorzeichen von m links oder rechts unten ins Bild geschrieben, beim Plotter zentriert unter das Bild.

Literatur:

[1] W. Grossmann, Grundzüge der Ausgleichungsrechnung, Springer-Verlag, 1953

Vorgehen beim Abtippen

Das Programm besteht aus drei Teilen:

- Hauptteil: Bildschirm-I/O, mit optionaler Druckausgabe (Listing 1)
- Grafikteil (fakultativ, Listing 2)
- Plotteil (fakultativ, Listing 3)

Anwender, die einen Drucker, aber nicht den Plotter besitzen, lassen Listing 3 weg; Anwender ohne Drucker, aber mit Plotter, lassen die Zeilen 180, 210-250, 550 und 560 (=Druckausgabe) weg.

Wer Drucker und Plotter hat, schaltet nach dem Ausdrucken vom Drukker zum Plotter um. □

Skeptiker überzeugen wir besonders gern:

Gute Software muss nicht teuer sein! Aber anwenderfreundlich und technisch perfekt!

ES'EN'ES-Software ist in ihrem Bereich konkurrenzlos.
Einfach und logisch in der Anwendung – informativ und flexibel im Leistungsangebot. Verständlich abgefasste und übersichtlich gegliederte Einführungen machen unsere Programme problemlos einsetzbar.

Programm SOS-OFFICE

SFr. 40.–

SOS-OFFICE – der elektronische Schreibtisch: das ideale Arbeitsinstrument für den Büroalltag

- Taschenrechner
- Schreibmaschine
- Drucker-Einstellung
- Notizblock
- Agenda
- Dateidienst usw.

SOS-OFFICE, eines der erfolgreichsten Programme in den USA – erstmals in einer deutschsprachigen Adaption – bietet Ihnen jederzeit zuschaltbare Hilfe, um Ihren Computer besser zu nutzen.

Direkt über **SOS-OFFICE** aktivieren Sie die **verschiedenen Schriftarten des Druckers;** bestimmen Seitenlänge und unteren Rand. Diese Einstellungen können auch mit der im Programm simulierten **Schreibmaschine** weiterverwendet werden. Mit dem **Taschenrechner** können Werte aus dem Vordergrundprogramm übernommen, wie mit einem normalen Taschenrechner verarbeitet und an eine beliebige Stelle zurückgegeben werden. **Datum und Uhrzeit** sind stets abrufbereit und veränderbar. Eine **Notiz** ist zu vorbestimmter Uhrzeit auf den Bildschirm abrufbar. Mit dem **Kalender,** bis ins Jahr 2050 datiert, werden Tagesnotizen festgehalten. Die Speicherkapazität des **Notizblocks** richtet sich nach der verwendeten Diskette. Die Notizen können jederzeit abgerufen, geändert und ausgedruckt werden; gespeicherte **Dateien** aufgelistet, am Bildschirm überarbeitet und seitenorientiert ausgedruckt werden.

SOS-Office ist innerhalb jedes anderen Programmes abrufbereit und wird dadurch zu einer unverzichtbaren Hilfe für jeden Benutzer.

Technische Angaben:

- Unterstützt alle Computer, die IBM-kompatibel sind
- Unterstützt alle grafikfähigen Drucker, die IBM- oder EPSON-kompatibel sind
- Benötigt DOS-Version ab 2.0
- Speicherbedarf 256 KB (RAM)
- Wird resident geladen
- Ist in Assembler-Routinen geschrieben

Vertrauen gegen Vertrauen

Unser Versand erfolgt mit Rechnung und achttägigem Rückgaberecht. Die Disketten sind nicht kopiergeschützt. ES'EN'ES-Software ist in allen deutschsprachigen Ländern erhältlich.

- -

Ich bestelle bei ES'EN'ES:

	3.5 Zoll	5.25 Zoll
___ **Programm SOS-Office** SFr. 40.–	☐	☐
___ **Programm PRT-SC** SFr. 40.–	☐	☐
___ **Programm ERTRAG** SFr. 60.–	☐	☐

Mit jedem Programm erhalten Sie eine ausführliche Bedienungsanleitung.

Mein Computer heisst: _____

Mein Drucker heisst: _____

Name/Vorname: _____

Strasse: _____

Land / PLZ / Ort: _____

Senden an:
ES'EN'ES AG, Postfach 2750
CH-6002 Luzern

ES'EN'ES

Technik von morgen zu Preisen von heute

Programm PRT-SC

SFr. 40.–

Mehr Möglichkeiten mit der Hardcopy-Taste!

Sie können Texte in 80er- oder 40er-Breite und Grafiken in 640×200 oder 320×200 Punktauflösung ausdrucken. Verschiedene Druckformate sind wählbar.

Bildschirminhalte können gespeichert und später beliebig weiterverarbeitet werden. Zudem steht eine Schreibmaschine mit verschiedenen Schriftarten zur Verfügung.

PRT-SC wird resident geladen und ist dann auf Tastendruck abrufbereit. Unterstützt alle Computer, die IBM-kompatibel und alle grafikfähigen Drucker, die IBM- oder EPSON-kompatibel sind. Benötigt DOS-Version ab 2.0, Speicherbedarf 32 KB (plus Betriebssystem und Programm). Unterstützt alle Video-Modalitäten des Colorgrafik-Adapters.

Programm ERTRAG

SFr. 60.–

Das Programm ERTRAG ist so vielseitig wie Ihre Phantasie und so einfach wie das Einmaleins: Ertragsbuchhaltung, Budgetkontrolle, Projektüberwachung, Stundenstatistik, Kassenbuch und vieles mehr. Es ermöglicht eine einfache Überschussrechnung mit gleichzeitiger Kontenkontrolle. Die Ertragsrechnung kann in eine Finanzbuchhaltung überführt werden und umgekehrt. Buchungsjournal, Dateien, Kalkulation und Kursumrechnungen können professionell genutzt werden.

ERTRAG unterstützt alle Computer, die IBM-kompatibel und alle Drucker, die IBM- oder EPSON-kompatibel sind. Benötigt DOS-Version ab 2.0 und hat einen Speicherbedarf von 128 KB (RAM).

Berechnung der Sonnenlaufbahn

Ein kurzweiliges und interessantes Programm für alle Architekten und Sonnenanbeter ist die Berechnung des Sonnenstandes an einem beliebigen Tag und Zeitpunkt. Fassaden und Fenster können nach genauen Bedürfnissen geplant werden. Ein kleines Fensterloch kann beispielsweise so in die Fassade gesetzt werden, dass das Sonnenlicht am Geburtstag der Bauherrin erstmals am Morgen um 7.00 h ins Schlafzimmer dringt. Ist es genügend klein dimensioniert, wird dies nur einmal im Jahr der Fall sein.

Geschrieben wurde das Programm in BASIC für den Sharp PC-1500. Nachfolgend gebe ich die notwendigen Formeln wieder, damit sie auch für andere Systeme übernommen werden können.

Für die Berechnung sind die geografische Lage des Gebäudes, ein beliebiger Tag im Jahr und eine Tageszeit erforderlich.

Erwin Günther

Der Tag (d) errechnet sich näherungsweise aus der Formel

$d = D + 30 * (M - 1)$
wobei D = Tag, M = Monat

Nach dem 30. Mai gibt man die Tage genauer direkt ein. Mit der Anzahl Tage kann die Deklination der Sonne und die Zeitgleichung errechnet werden.

Deklination

$\delta = 3{,}68 * \sin d - 23{,}16 * \cos d$

Zeitgleichung

$g = -0{,}1236 * \sin d - 0{,}1571 * \sin 2d - 0{,}0507 * \cos 2d$

Im nächsten Schritt wird der Stundenwinkel (von Mittag gerechnet) ermittelt. In dieser Gleichung müssen die Lokalzeit (LZ), der Meridian des Ortes (λg) östlich von Greenwich und die zuvor berechnete Zeitgleichung (g) bekannt sein.

Stundenwinkel

$\omega = (LZ - 12 + g) * 15 + g - 15$

Für die Berechnung der Sonnenhöhe ist schliesslich die geografische Breite des Ortes (φg) und die zuvor berechnete Deklination (δ) und der Stundenwinkel (ω) erforderlich.

```
TAG      102.0000
ZEIT       6.0000
HOEHE      1.3216 G
AZIMUT  -100.4259 G

ZEIT       7.0000
HOEHE     11.4213 G
AZIMUT   -89.4914 G

ZEIT       8.0000
HOEHE     21.5134 G
AZIMUT   -78.3006 G

ZEIT       9.0000
HOEHE     31.3500 G
AZIMUT   -65.5522 G

ZEIT      10.0000
HOEHE     40.1749 G
AZIMUT   -51.0216 G

ZEIT      11.0000
HOEHE     47.0813 G
AZIMUT   -32.4432 G

ZEIT      12.0000
HOEHE     50.5649 G
AZIMUT   -10.4606 G

ZEIT      13.0000
HOEHE     50.4541 G
AZIMUT    12.5002 G

ZEIT      14.0000
HOEHE     46.3804 G
AZIMUT    34.3125 G

ZEIT      15.0000
HOEHE     39.3503 G
AZIMUT    52.2857 G

ZEIT      16.0000
HOEHE     30.4503 G
AZIMUT    67.0657 G

ZEIT      17.0000
HOEHE     20.5807 G
AZIMUT    79.3238 G
```

```
5000:"SON":USING
     "####.####"
5005:INPUT "GEOGR
     . LAGE NOERD
     L.";N,"OESTL
     .";E
5010:INPUT "TAG";
     A,"MONAT";M:
     D=A+30*(M-1)
5020:INPUT "TAGES
     ZEIT";Z:T=
     DEG Z
5021:LPRINT "TAG
       ";D
5025:FOR X=1TO 12
5026:Z=Z+1.00000:
     T=DEG Z
5027:LPRINT "ZEIT
       ";Z
5030:G=-0.1236*
     SIN D-.1571*
     SIN (2*D)-.0
     507*COS (2*D
     )
5040:C(1)=3.68*
     SIN D-23.16*
     COS D
5060:C(2)=(T-12+G
     )*15+E-15
5061:C(3)=((T+.00
     001)-12+G)*1
     5+E-15
5070:C=ASN (SIN N
     *SIN C(1)+
     COS N*COS C(
     1)*COS C(2))
5071:F=ASN (SIN N
     *SIN C(1)+
     COS N*COS C(
     1)*COS C(3))
5080:B=ASN (COS C
     (1)*SIN C(2)
     /COS C)
5081:H=ASN (COS C
     (1)*SIN C(3)
     /COS F)
5083:IF H<BTHEN
     LET B=90-ABS
     B+90
5090:LPRINT "HOEH
     E ";DMS C;"
     G"
5100:LPRINT "AZIM
     UT";DMS B;"
     G"
5200:LF 1:NEXT X
```

Sonnenhöhe

$\sin\varphi s = \sin\varphi g * \sin\delta + \cos\varphi g * \cos\delta * \cos\omega$

Aus der Sonnenhöhe über dem Horizont errechnet sich schliesslich das noch fehlende Azimut mit

Azimut

$\sin\alpha s = \cos\delta * \sin\omega / \cos\varphi s$

Geografische Lage

Das Beispiel gibt die Werte von 47,03 nördlicher Breite und 8,326 östlich von Greenwich, die nach der Landeskarte 1:50'000 ermittelte Lage meines Standortes in Luzern wieder.

Für einen beliebigen Standort können die Koordinaten nach der Landeskarte ermittelt werden. Für eine beliebige Lage in der Schweiz errechnen sich die genauen geografischen Lagen östlich von Greenwich und nördlich der Erdhalbkugel wie folgt:

Die Koordinate km 674'000 ist identisch mit dem Meridianwinkel 8°25'.

Eine Minute Differenz entspricht einer Strecke von 1.260 km. In der Gleichung bedeutet

XXX = Koordinate in km
NNN = Koordinate in m

Mit einem technisch wissenschaftlichen Rechner heisst die Gleichung

Der Meridian

$\lambda g = DEG\ 8.25 - DEG\ (674 - XXX.NNN) / 1.26 / 100)$

Mit der gleichen Formel kann die geografische Breite ermittelt werden, 224 km entsprechen dem Winkel 47°10' nördlicher Breite.

Eine Minute Differenz entspricht einer Strecke von 1.85 km.

Geografische Breite

$\varphi g = DEG\ 47.10 - DEG\ ((224 - YYY.NNN) / 1.85 / 100)$

Das Programm für den Sharp PC-1500

Die Programmzeilen 5061, 5071 und 5081, sowie die Zeile 5083 können weggelassen werden. Allerdings berechnet die vorgenannte Formel die Werte von Westen = 90° nach Norden > 90° nicht mit 91 - 180°, sondern mit 89 - 0°. Beim Uebergang von Westen nach Norden können aber bei Einzelberechnungen Falschinterpretationen der Richtung auftreten.

Aus diesem Grunde habe ich die Berechnung mit einer Zeitdifferenz von einer Sekunde ein zweites Mal berechnet. In Zeile 5083 merkt dann das Programm, ob die Werte zunehmend, also von Süden nach Westen, respektive von Süden nach Osten oder abnehmend von Westen nach Norden, respektive von Osten nach Norden sind.

Ein in sphärischer Trigonometrie Bewanderter kann diesen Mangel wahrscheinlich eleganter lösen. Für die Formel wäre ich dankbar.

Das Programmbeispiel gibt die Werte für den 12. April von Morgens um 6 h bis Abends 17 h wieder. Die Ausgabe der Sonnenhöhe sowie des Azimutes wird in Grad.Minuten und Sekunden ausgegeben.

Das Azimut wird in der Richtung von Süden nach Westen + und in der Richtung von Süden nach Osten - berechnet. □

Selektiver Bildschirmausdruck für beliebige Computer

Heutige Computer verfügen meist über die Möglichkeit, den Inhalt des Bildschirms auf den Drucker zu bringen. Damit lässt sich alles festhalten, was gerade angezeigt wird. Dies ist eine sehr nützliche Funktion, denn man kann damit ohne viel Mühe gewisse Zustände dokumentieren. Nun gibt es aber immer noch Rechner, bei denen diese Funktion nicht standardmässig vorhanden ist. Für diese Fälle ist das hier beschriebene einfache Programm nützlich, welches zusätzlich noch gestattet, einen gewissen Bereich zu markieren und nur diesen auszudrucken.

Ernst Pfenninger

Listing 1

```
program printScreen(input,output);


const
              {es folgen Tastencodes fuer die Bedienung}
  ctrlK=^K;
  ctrlB=^B;
  ctrlP=^P;
  esc=#27;
              {es folgen Tastencodes fuer Cursor-Steuerung}
  links=^S;
  rechts=^D;
  auf=^E;
  ab=^X;

type
  invNorm=(invers,normal);

var
  eingabe, zeichen: char;
  zeile,spalte, i, beginn, altBeginn, ende, altEnde: integer;
  bildschirm: array[81..1920] of char;



procedure getChar(var zeichen:char);
{--------------------------------}
{Prozedur zum Lesen des Zeichens auf dem Bildschirm an Cursor-Posi-
 tion, Cursor wird um eine Stelle nach rechts weiterbewegt}
{Version fuer ITT 3030}
begin
  inline ($3E/$04/              {  MVI   A,H          }
          $CD/$9A/$FE/          {  CALL  $FE9A        }
          $2A/zeichen/          {  LD    HL,(zeichen) }
          $73);                 {  LD    (HL),E       }
end; {procedure getChar}



procedure umsetzen(beginn,ende:integer;art:invNorm);
{--------------------------------------------------}
{Prozedur zum Invers-setzen und Normal-setzen von Teilen des Bild-
 schirms. Als "Invers" wird dabei die Darstellung bezeichnet, wel-
 che durch die Prozedur "LowVideo" in Turbo-Pascal erreicht wird.}
var i, zeile, spalte:integer;
    zeichen: char;
begin
  if art=invers then lowVideo;
  zeile:=trunc(beginn/80); spalte:=beginn-zeile*80;
  if spalte=0 then begin spalte:=80; zeile:=pred(zeile); end;
  gotoXY(spalte,zeile);
  for i:=beginn to ende do
  begin
    write(bildschirm[i]); spalte:=succ(spalte);
    if spalte=81 then spalte:=1;
  end;
  normVideo;
end; {procedure umsetzen}
```

Das Programm wurde in Turbo-Pascal geschrieben und kann für beliebige Rechner angepasst werden. Dazu ist nur eine einzige Prozedur anzupassen, welche rechnerspezifisch ist. Im vorliegenden Listing ist das Programm für den ITT 3030 geeignet. Die Anpassung für andere Rechner betrifft die Prozedur «get Char». Diese liest das an der Position des Cursor auf dem Bildschirm stehende Zeichen und gibt es in der Variablen «zeichen» zurück. Gleichzeitig wird der Cursor um eine Stelle nach rechts gerückt. Es sollte keine Schwierigkeiten machen, diese Prozedur auch für andere Rechner zu schreiben. Entweder kann dazu eine BIOS-Funktion verwendet werden (z.B. bei IBM), oder der Bildschirmspeicher kann direkt gelesen werden. Für den IBM-PC ist eine mögliche Lösung angegeben, für andere Rechner müssen gegebenenfalls die Unterlagen des Herstellers beigezogen werden.

Für IBM ist die Prozedur «getChar» in Listing 2 angegeben. Es wird eine BIOS-Funktion verwendet, welche durch einen Software-Interrupt gestartet wird. Da Turbo-Pascal das Generieren von Software-Interrupts gestattet, ist nicht einmal die Verwendung von Maschinensprache-Befehlen notwendig. Es wird Interrupt Nummer 16 ausgelöst, nachdem vorher die Register auf den richtigen Wert gesetzt wurden. Dies betrifft Register BX, welches auf 0 gesetzt wird (Bildschirmseite 0) und AX, welches auf 2048 gesetzt wird (bedeutet «Lesen eines Zeichens an der Cursor-Position»). Zusätzlich muss der Cursor um eine Stelle nach rechts geschoben werden. Dazu wird die Position des Cursors gelesen, um 1 erhöht und der Cursor an die neue Position gesetzt.

Das vorgestellte Programm wird mit dem Compiler von Turbo-Pascal in eine .COM-Datei verwandelt, welche direkt von DOS aus abgerufen werden kann. Das Programm PRSCR (PrintScreen) wird gestartet, nachdem die interessierenden Daten auf dem Bildschirm angezeigt werden. Automatisch wird der gesamte Bildschirm auf LowVideo geschaltet, was bedeutet, dass der gesamte Bildinhalt für den Ausdruck gewählt wird. Dies ermöglicht es, einen vollständigen Ausdruck des Bildschirmes ohne Markierung von Anfang und Ende vorzunehmen. Jetzt kann der Cursor

mit Hilfe der Tasten CtrlE (auf), CtrlX (ab), CtrlS (links) und CtrlD (rechts) beliebig bewegt werden. Die Tasten für die Cursorsteuerung wurden dem Editor von Turbo-Pascal nachempfunden und können natürlich beliebig geändert werden.

Mit CtrlB (Beginn) und CtrlK (Ende) kann Beginn und Ende des Bildschirmausschnittes bezeichnet werden. Mit CtrlP (Print) wird der Druck gestartet und mit ESC (Abbruch) kann das Programm ohne Ausdruck verlassen werden.

Das Programm hat sich im täglichen Gebrauch schon oft als nützlich erwiesen. Es macht manche manuelle Abschreibearbeit überflüssig, spart dabei Zeit und elimeniert Fehlerquellen. Dabei wird dank der eingebauten Funktion des Bildschirmausschnittes nur das Notwendige und Wesentliche gedruckt, was die Uebersichtlichkeit erhöht. □

```
                        {Beginn des Hauptprogramms}
                        {-------------------------}
  begin
  gotoXY(1,24); clrEol;
  gotoXY(1,24);
  write('Beginn=CtrlB, Ende=CtrlK, Druck=CtrlP, Abbruch=ESC');
  for zeile:=1 to 23 do
  begin
    gotoXY(1,zeile);
    for spalte:=1 to 80 do
    begin
      getChar(zeichen); bildschirm[zeile*80+spalte]:=zeichen;
    end;
  end;
  zeile:=1; spalte:=1;
  beginn:=zeile*80+spalte; altBeginn:=beginn;
  ende:=23*80+80; altEnde:=ende;
  umsetzen(beginn,ende,invers);
  repeat
    gotoXY(spalte,zeile);
    repeat
      read(KBD,eingabe);
    until eingabe in [ctrlB,ctrlK,ctrlP,links,rechts,auf,ab,esc];
    case eingabe of
      ctrlB: begin
               beginn:=zeile*80+spalte;
               if beginn>ende then beginn:=altBeginn;
             end;
      ctrlK: begin
               ende:=zeile*80+spalte;
               if ende<beginn then ende:=altEnde;
             end;
      links: if spalte>1 then spalte:=pred(spalte);
      rechts: if spalte<80 then spalte:=succ(spalte);
      auf:    if zeile>1 then zeile:=pred(zeile);
      ab:     if zeile<23 then zeile:=succ(zeile);
    end; {case}
    umsetzen(altEnde,ende,invers);
    umsetzen(beginn,altBeginn,invers);
    umsetzen(succ(ende),altEnde,normal);
    umsetzen(altBeginn,pred(beginn),normal);
    altBeginn:=beginn; altEnde:=ende;
  until eingabe in [ctrlP, esc];
  if eingabe=ctrlP then
  begin                    {Beginn des Ausdruckens}
    zeile:=trunc(beginn/80); spalte:=beginn-zeile*80;
    if spalte=0 then begin spalte:=80; zeile:=pred(zeile); end;
    {Hier kann bei Bedarf ein Befehl zu Initialisierung des
     Druckers eingefuegt werden.}
    for i:=1 to pred(spalte) do write(LST,' ');
    for i:=beginn to ende do
    begin
      write(LST,bildschirm[i]);
      spalte:=succ(spalte);
      if spalte=81 then begin spalte:=1; writeln(LST); end;
    end;
    writeln(LST);
    {Hier kann bei Bedarf ein Befehl fuer Seitenvorschub des
     Druckers eingefuegt werden.}
  end;
end.
```

Computer-Splitter

Floppy-Disk-Controller

Für Entwickler von IBM-AT-kompatiblen Computern gibt es eine Single-Chip-Lösung zum Aufbau des Floppy-Controllers. Der FDC 9268 von SMC ist eine Version des bekannten Industriestandards 765A, erweitert um Funktionen, die IBM im AT mit externen Schaltungen realisiert. Info: Datacomp AG, Zürcherstrasse 20, 8952 Schlieren, Tel. 01/730'21'65. □

RISC-Coprozessor

Einen weiteren Meilenstein in der Einführung superschneller Prozessoren setzt Novix mit dem NB4100. Diese Karte wird direkt in den PC-Bus gesteckt und läuft parallel zur Host-CPU. Der mit 6 MHz getaktete NC4016 ermöglicht acht Millionen Hochsprachbefehle pro Sekunde. Der Datenaustausch mit dem PC-Bus erfolgt über einen gemeinsamen Speicherbereich von 2 KByte. Das NB4100 wird komplett mit Forth-83 und dem Interface zum PC-Massenspeicher geliefert. Zusammen mit 128 KByte lokalem RAM können Echtzeitaufgaben gelöst werden, die bisher ausser der Reichweite von PC's lagen. Info: OmniRay AG, Industriestrasse 31, 8305 Dietlikon, Tel. 01/835'21'11. □

Listing 2

```
procedure getChar(var zeichen:char);
{---------------------------------}
{Prozedur zum Lesen des Zeichens auf dem Bildschirm an Cursor-Posi-
 tion, Cursor wird um eine Stelle nach rechts weiterbewegt}
{Version fuer IBM-PC}
var
  result: record
            ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
          end;
begin
  result.bx:=0;                    {Bildschirmseite 0}
  result.ax:=2048;                 {Funktion "Lesen eines Zeichens"}
  intr(16,result);                 {Interr. 16 fuer Bildschirmfkt."}
  zeichen:=char((result.ax) and 255);
  result.bx:=0;                    {Bildschirmseite 0}
  result.ax:=768;                  {Funktion "Cursorposition lesen"}
  intr(16,result);                 {Interr. 16 fuer Bildschirmfkt."}
  result.dx:=succ(result.dx);      {Cursorposition um 1 erhoehen}
  result.bx:=0;                    {Bildschirmseite 0}
  result.ax:=512;                  {Funktion "Cursorposition setzen"}
  intr(16,result);                 {Interr. 16 fuer Bildschirmfkt."}
end; {procedure getChar}
```

COMPUTER MARKT

DIE AKTUELLE COMPUTERINFORMATION 6/87

Kompakte Leistung im Epson-Standard

Die PC-Palette von Epson ist um eine attraktive Neuheit reicher. Voll kompatibel zum anerkannten Industriestandard, schnell, bedienungsfreundlich, flexibel und leistungsfähig, so lauten einige Merkmale des neuen AT-kompatiblen PC AX2 von Epson. Von der Systemarchitektur her identisch mit dem bewährten AX, entspricht der Neue in seinen Ausmassen jedoch der 19-Zoll-Industrienorm.

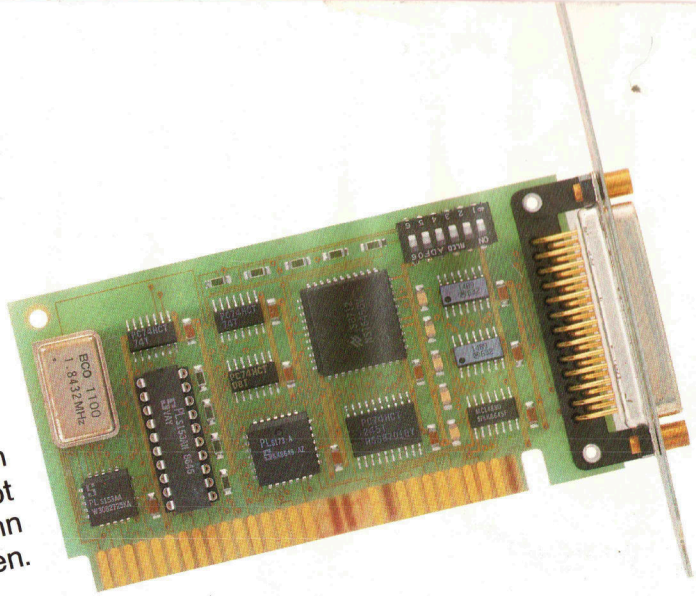
Für Anwendungen in der Industrie lässt sich der PC AX2 in 19-Zoll-Racks einbauen. Seine Konstruktion bietet enorme Ausbaumöglichkeiten, so dass fast unbegrenzte Einsatzmöglichkeiten offen stehen. Ob nun für Prozesssteuerungsanlagen, ob für komplexe Maschinenstrassen, überall kann man sich auf die bewährt

Epson-Qualität verlassen
nur dank seinem gefäll
steht dem PC AX
Bürowelt offen
in einem Ne

Die t
eind

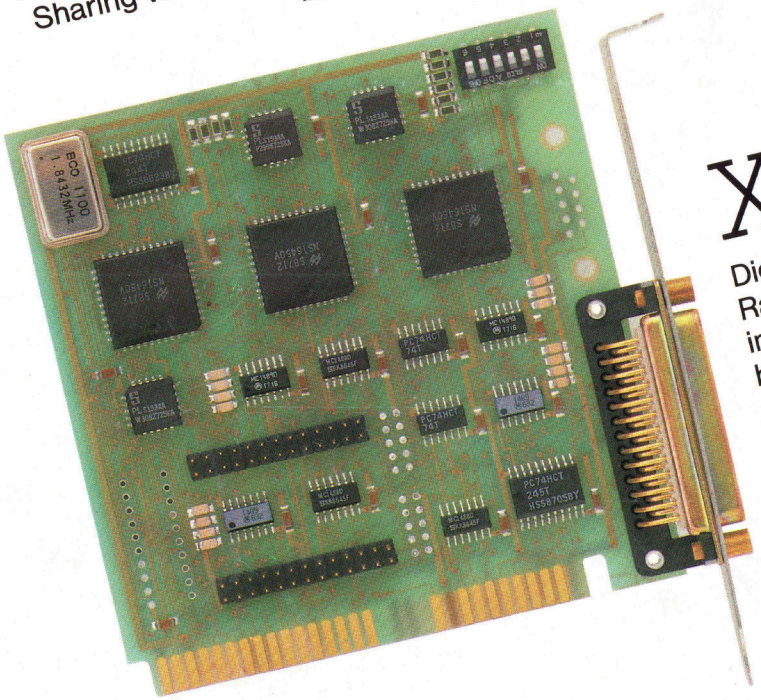
Infosys

XPort 1



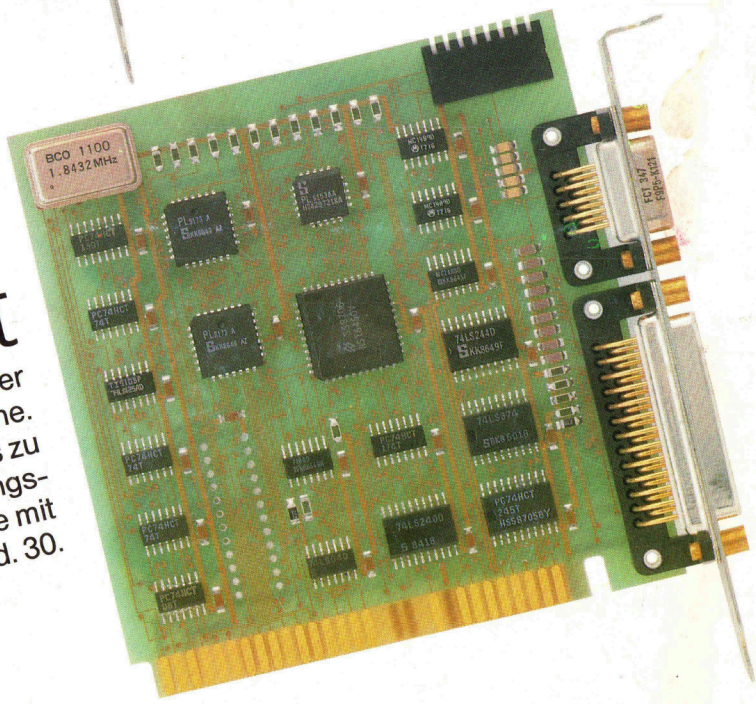
Die richtige RS 232 (V24) Schnittstelle für PCs, XTs, PS/2 Mod. 30 und alle ATs. Selbstverständlich unterstützt XPort 1 das von IBM definierte Interrupt Sharing Verfahren. Die mitgelieferte Software kann bis zu 4 COM: Ports verwalten.

XPort 3



Die dreifache Leistung der XPort 1 auf kleinstem Raum benötigt nur einen Steckplatz. Beim Einsatz im IBM PS/2 Mod. 30 steht eine extra Rückwand-buchsenbefestigung zur Verfügung.

Combi Port



Die richtige Verbindung einer seriellen mit einer parallelen Schnittstelle auf einer high tech Platine. Interrupt Sharing und Unterstützung von bis zu 4 seriellen Schnittstellen gehören zum Leistungs-umfang. Entwickelt für alle Rechnersysteme mit PC-Bus, incl. IBM PS/2 Mod. 30.

Computer Elektronik Infosys GmbH, Am Kümmerling 2, 6501 Bodenheim, Telefon 061 35/30 81
Computer Elektronik Infosys AG, Im Rötel 10a, CH-6301 Zug, Telefon 042/21 78 57
Lager und Vertrieb Schweiz: Oberdorfstrasse 11, CH-8953 Dietikon, Telefon 01/741 30 41