M+K 88-3 DAS LESEN SIE IN DIESER NUMMER

Der Xerox 4020 Farb-Tintenstrahldrucker vermag mit Hilfe von 20 Düsen aus den sieben Farben Gelb, Cyan, Magenta, Violett, Grün, Rot und Schwarz mehr als 4'000 Farbnuancen herzustellen. Zudem verfügt er über fünf vollständige Zeichensätze, 14 Fremdsprachenvarianten und die Fähigkeit, Text in vier Stufen vertikal und horizontal zu komprimieren oder zu expandieren. Bei hoher Standard-Druckgeschwindigkeit erreicht der Xerox 4020 eine Auflösung von 120x120 Punkte/Zoll, für höchste Ansprüche lässt er sich sogar auf 240x120 Punkte/Zoll einstellen. Er benötigt nur gerade zwei Minuten für die Erstellung einer typischen Farbgrafik in Standardqualität und ist damit einer der leistungsfähigsten Drucker auf dem Markt. Der Xerox 4020 kann mit den verfügbaren Schnittstellen an nahezu alle PCs angeschlossen werden. Ausserdem wird er durch zahlreiche bekannte Anwender-Softwarepakete unterstützt. Dies ermöglicht die Produktion einer Vielzahl von CAD-Drucken, Geschäfts- und sogar Kunstgrafiken auf Papier oder auf Transparentfolien. Info: Micro Distribution & Trading Inc., Industriestrasse 404, 5242 Birr/Lupfig, Tel. 056/94'01'05. □

Ausgabe Juni 1988
Erscheint zweimonatlich
10. Jahrgang

COMPUTER aktuell

LEHRGÄNGE

GEWUSST WIE

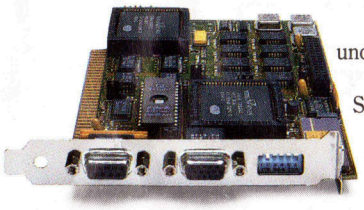
COMPUTER-BÖRSE

VORSCHAU

Bekommen sie bei der VEGA VGA höhere Auflösung und mehr Farben?

PC-PAINTBRUSH-SONDERAKTION!!!
Zum einmaligen Sonderpreis von nur Fr. 50.– erhalten Sie beim Kauf einer VEGA VGA-Karte das Programm: PC-PAINTBRUSH!!! Den in der Packung enthaltenen Coupon zusammen mit Fr. 50.– in bar oder Check an Computer 2000 einsenden, und Sie erhalten umgehend Ihr Programm!!!

Absolut.



Die VEGA VGA bietet hohe Auflösung und hohe Geschwindigkeit. Sie ist bis zu 400% schneller als eine Standard-EGA-Karte.

Es gibt keinen Grund, weshalb Sie sich mit weniger zufrieden geben sollten als der klarsten, feinsten Auflösung und der breitesten, vielfältigsten aller Farbpaletten.

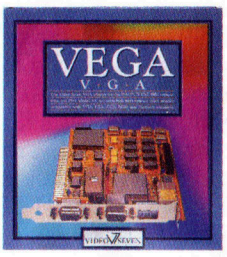
Die VEGA VGA wurde von einem Hersteller entwickelt, der schon mehr als 450.000 EGA-Karten verkauft hat. Sie bietet neben der höheren Auflösung und der größeren Farbauswahl noch viele andere Vorteile.

Es fängt damit an, daß die VEGA VGA es zuläßt, gleichzeitig 256 Farbtöne, auszuwählen—aus einer Gesamtpalette von 262.144 Tönen mit einer Auflösung von 320 x 200 Pixels bei einer erstaunlichen Bildqualität.

Der Grafik-Adapter ermöglicht außerdem eine Auflösung von 640 x 480 Pixels in 16 Farben für klare, scharfe Schrift und bis zu 132 Buchstaben pro Zeile auf dem Bildschirm.

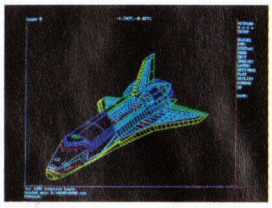
Aufgrund ihrer kurzen Bauart paßt die Karte in jeden IBM PC/XT/AT und alle kompatiblen Geräte sowie in den PS/2 Model 30.

Aber die VEGA VGA ist nicht nur auf dem Bios-Level kompatibel. Sie bietet absolute Software-Kompatibilität für alle VGA-Programme. Und sie ist abwärts-kompatibel, so daß sie auch zu jedem möglichen Software Programm für EGA, CGA, MDA und Hercules paßt—garantiert!



Die VEGA VGA kommt mit einer vollen 5-Jahres-Garantie.

Sie geht über den VGA-Standard hinaus. Die mitgelieferten Treiber ermöglichen eine Auflösung von 800 x 600 Pixels mit 16 Farben von 262.144.



Die Treiber sind ausgelegt für 800 x 600 Pixels.

Der besondere technische Aufbau der Karte macht sie bis zu 400% schneller als eine Standard-EGA-Karte.

Durch unsere neuartige Konstruktion ist sie sowohl für analoge als auch für digitale Monitore einsetzbar, so daß sie mit jedem möglichen Monitor funktioniert, den Sie momentan benutzen.

Und schließlich geben wir Garantie für volle 5 Jahre.

Jetzt wissen Sie, welche Vorteile die VEGA-VGA-Karte für Sie bereithält, aber kann sie all dies für einen Bruchteil der Kosten, den eine komplette Neu-Anlage verursachen würde?

Absolut.

Für weitere Informationen wenden Sie sich bitte an den Video Seven-Repräsentanten.

VIDEO SEVEN

COMPUTER 2000 AG Lettenstrasse 11, CH-6343 Rotkreuz, Telefon: 042/65 11 33, Telex: 862376 COAG CH

Voraussetzung für die Darstellung von 256 Farben aus 262.144 Tönen ist ein Monitor mit Analog-Eingang. Der Monitor muß eine entsprechend hohe Auflösung unterstützen. Es folgen eingetragene und nicht eingetragene Handelsmarken der aufgeführten Firmen: VEGA VGA, Video Seven, Video Seven Incorporated; Hercules, Hercules Computer Technology Incorporated; IBM, PS/2 Model 30, International Business Machines Corporation. Video Seven behält sich das Recht vor, technische Änderungen ohne Vorankündigung durchzuführen. Die Monitor-Abbildung erfolgte mit freundlicher Genehmigung der RIX Softworks, Inc.

Künstliche Intelligenz in der Praxis

Wer sich in das Spannungsfeld der «Künstlichen Intelligenz» begibt, muss mit Emotionen rechnen. Die heute beginnende Serie zu diesem Thema hat nicht das Ziel, diese Emotionen möglichst schnell abzubauen - manchmal wird sie wohl eher noch mehr Oel ins Feuer giessen. Sinn dieses Tuns soll die Anregung zum Nachdenken, zur Diskussion sein, und zur Umwandlung solcher Emotionen in eine eigene, fundierte Meinung. Nach einigen grundlegenden Betrachtungen, um die man nicht herumkommt, werden wir deshalb sehr schnell in die Praxis einsteigen. Eigenes Erforschen und Erfahren lässt die Möglichkeiten und Grenzen der KI am eindrücklichsten erkennen.

Beat und Fred Kipfer

Die Idee, menschliche Intelligenz künstlich nachzubilden, kann sensible, weniger technisch orientierte Menschen sehr erschrecken und in Angst versetzen. Dabei erschöpfen sich solche Aengste beileibe nicht vor dem bereits von Orwell beschriebenen «Big Brother», der uns allzeit und überall im Auge behält - da taucht, von Horror- und Utopiefilmen geschürt, viel Schlimmeres auf: Als wahrhaft fürchterliches Schreckgespenst gar Androiden, die sich durch selbständiges Denken und Handeln über den Menschen stellen. Unwesen, die eines Tages sogar die Welt erobern und uns zu ihren Sklaven degradieren - nicht zuletzt deshalb vielleicht, weil die Natur oder eine andere höhere Macht uns die Quittung für den begangenen Frevel überreicht.

Aengste also, die auch unsere Vorfahren schon hatten, wenn auch wir - eigentlich unberechtigterweise - über deren damalige Gründe heute milde lächeln... Das Phänomen als solches scheint sich in der Geschichte der Menschheit regelmässig zu wiederholen.

Zweifellos und glücklicherweise denken aber immer mehr Menschen zu realistisch und zu aufgeschlossen, um sich ernsthaft mit solchen Vorstellungen auseinanderzusetzen. Von der auch nur geringsten Wahrscheinlichkeit solcher Entwicklungen sind wir doch, aus überaus handfesten technischen Gründen, noch sehr weit entfernt.

Trotzdem bleiben da einige durchaus berechtigte Bedenken, die nicht mehr ohne weiteres in die Bereiche einer überbordenden Fantasie verwiesen werden dürfen: Könnte eine künstlich geschaffene Intelligenz den Menschen im Arbeitsprozess ersetzbar machen? Eine Frage, die auch ein sehr realitätsbezogener Mensch heute nicht mehr als abwegig bezeichnen darf.

Die Roboter sind schon da

Immerhin ist es doch so, dass sich fortschreitend von früher eher einfachen, immer kompliziertere manuelle Arbeiten sehr viel rationeller durch Roboter, als durch Menschen ausführen lassen. Durch Maschinen, die während 24 Stunden am Tag und sieben Tagen in der Woche mit hoher Zuverlässigkeit arbeiten. Sie verlangen weder Urlaub noch Sozialeinrichtungen, werden weder krank noch schwanger, treten keiner Gewerkschaft bei, murren nicht und demonstrieren nicht für höhere Löhne.

Zum Glück braucht es derzeit immer noch Menschen, die solche Maschinen programmieren, bedienen und überwachen. Was aber, wenn es gelingt, Industrieroboter - die immerhin tatsächlich schon imstande sind, sich selber herzustellen - auch noch intelligentes Handeln beizubringen? Könnten dann auch die letzten verbleibenden Arbeitsplätze, ja sogar diejenigen der Techniker und Ingenieure durch Rechner ersetzt werden? Welche Aufgaben verblieben noch für den Menschen - oder konkreter, für den bisher irgendwo in einem Produktions- oder Dienstleistungsprozess integrierten Angestellten?

Kommen diese Probleme wirklich auf uns zu? Wenn ja, muss, oder besser kann man sich dagegen wehren? Wahrscheinlich, und aus der Erfahrung gesprochen, doch eher nicht. Zahllose Erfindungen, durch welche die Menschheit ihren Untergang befürchtete, wie Rad, Schiesspulver, Kanone, Flugzeug, Atom- und Weltraumtechnik waren als Fortschritte und Weiterentwicklungen auch nicht abwendbar. Selbst diejenigen, die ihn nicht wollen, müssen mit dem Fortschritt leben lernen. Von einseitigen Verboten ziehen andere Nutzen, und weltweit gesehen ändert sich dadurch gar nichts.

Wer einen - auch nur möglichen - Feind bekämpfen will, muss ihn zunächst kennen. Es wird das Beste sein, wenn wir uns auch daran halten. Weil aber die Idee der Künstlichen Intelligenz keineswegs erst eine Ausgeburt des Mikrocomputer-Booms ist, müssen wir hier einen kurzen historischen Abstecher machen, der Sie vielleicht überraschen wird.

Die Geschichte der «Künstlichen Intelligenz»

«Von Homunculus zur Artificial Intelligence» oder «Der Wunsch des Menschen sich selber nachzubilden», liesse sich die Geschichte der «Künstlichen Intelligenz» (KI) wohl recht zutreffend betiteln.

In der Geschichte und der Literatur finden sich immer wieder Persönlichkeiten, die den Menschen ein zweites Mal erschaffen wollten. Homunculus, eine Erfindung des Paracelsus (1493-1541), ist eines der bekanntesten künstlich erschaffenen Wesen in der Geschichte der Entwicklung menschlicher Abbilder.

Das Rezept des Paracelsus, nach dem er seinen Homunculus erschaffen haben will, sei hier in Stichworten wiedergegeben: Man vergrabe Sperma für 40 Tage in Pferdedünger und magnetisiere es in der richtigen Weise. Ernährt man es nun mit dem Geheimnis des menschlichen Blutes, so wird es 40 Wochen später (!) leben und intelligent sein.

Nicht alle Menschenentwickler haben sich derart abstruser Hilfsmittel bedient. Bereits über 2000 Jahre vor Paracelsus beschreibt Homer in Ilias (ca. 850 v.Chr.) den Drang der Götter, sich Ebenbilder zu erschaffen. Dort jedoch geschieht dies viel konventio-

Glossary

Algorithmus

Nach einem bestimmten Schema ablaufender Rechenvorgang

Brainstorming

gemeinsames Bemühen, durch spntanes Aeussern von wertfreien Einfällen die Lösung eines Problems zu finden

heuristisch

erfinderisch, das Auffinden bezweckend

Intuition

Eingebung, ahnendes Erfassen, unmittelbare Erkenntnis

neller in der Schmiede des Hephaistos [3].

Die jüdische Sage berichtet von einem aus Ton geschaffenen «Golem», den der Prager Rabbi Löw solange als Synagogendiener beschäftigt hat, bis er ihm nicht mehr gewachsen war und ihn wieder zurückverwandeln musste.

Um den Reigen der historischen und literarischen Wesen abzuschliessen, sei noch auf Mary Shelley's Frankenstein (1818) verwiesen, der wohl zu den Berühmtesten seiner Art gehört.

Auffallenderweise zeigt sich auch in der Geschichte immer dieselbe, eingangs erwähnte Angst, dem selbst geschaffenen unterlegen zu sein und schliesslich von ihm dominiert zu werden. In Sagen und Geschichten mit Fantasiegebilden (Golem, Frankenstein, usw.) findet zudem die «Bestrafung für den Frevel» auch schön regelmässig tatsächlich statt, indem die Story ein böses Ende nimmt. Der Erfolg aus der Anmassung des gottesähnlichen Tuns rächt sich nach dem Schema des Zauberlehrlings (Goethe): «Die Geister, die ich rief, werd ich nun nicht los!» Objektiv betrachtet, dürften die Hintergründe dafür wohl weniger im mangelnden Selbstvertrauen der monsterschaffenden Autoren, als in religiösen Bedenken zu suchen sein - dort ist die Nachbildung des Menschen in jedem Falle verwerfliches Tun, das unangenehme Konsequenzen nach sich ziehen muss.

Wenn wir den Begriff der bösen Roboter und anderer Unwesen vergangener Jahrhunderte mit demjenigen der jetzt aktuellen «Künstlichen Intelligenz» vertauschen, so hat sich an der Situation nicht viel verändert. Vergleichbare Romane und Filme unserer Zeit, die sich hier als Diskussionsmaterial heranziehen liessen, gibt es mehrere. Insbesondere ist auch dargestellt worden, dass eine Welt von Robotern eben nicht auf menschliche Intelligenz verzichten kann und letztlich trotz ihrer Uebermacht durch eben diese menschliche Intelligenz zugrunde geht. Das deckt sich durchaus mit dem eben Gesagten.

Im wesentlichen blieb das alles auch so, als sich die Anstrengungen mehr darauf verlegten, «künstliche Menschen» nicht nur zu schaffen, um deren echte Vorbilder zu erstaunen, sondern um ihnen die tägliche Arbeit zu erleichtern.

Im 17. und 18. Jahrhundert waren es vor allem Mechaniker, welche sich mit dem Thema der Nachbildung des Menschen beschäftigten. Bei dieser «Kunstform» ging es weniger darum, den Menschen als solchen, als eher einige seiner manuellen Tätigkeiten zu simulieren.

Einige berühmte Automaten der Vergangenheit

Zwei der berühmtesten alten Automaten entstanden in der ersten Hälfte des 18. Jahrhunderts: Der Schweizer Uhrmacher Pierre Jaquet-Droz schuf 1772 den «Schriftsteller», ein sitzender Knabe an einem Schreibtisch, der eine Feder in Tinte taucht und den berühmten Satz des Philosophen René Descartes schreibt: «Ich denke, also bin ich». Der Automat lässt sich durch eine Scheibe am Rücken der Figur mit Texten bis zu 40 Buchstaben Länge «programmieren», so dass er sich unterschiedlichen Gelegenheiten anpassen kann. In den nächsten zwei Jahren erschaffte der Konstrukteur auch noch einen Bruder des Schreibers, «Le Dessinateur», der mit Hilfe auswechselbarer Nocken verschiedene Zeichnungen anfertigen konnte.

Etwa um die gleiche Zeit baute in Paris Jacques Vaucanson eine flügelschlagende, quakende, fressende und die im Innenleben zerstampfte Nahrung als «Exkremente» ausscheidende mechanische Ente (Bild 1).

Die Form eines behelmten Menschen hatte die 1893 in den USA von George Moore gebaute «Laufende Lokomotive». Eine Dampfmaschine im Innern bewegte die Beine. Ihr Gang soll etwa doppelt so schnell wie derjenige eines Menschen gewesen sein (Bild 2).

Eine weit zweifelhaftere Karriere hat eine der berühmtesten der frühen intelligenten Maschinen hinter sich, der schachspielende Türke von Kempelens. Dieser Automat glich äusserlich einem Tisch, an welchem ein Türke Schach spielt. Im Sockel des Tisches verbarg sich die «Mechanik» der genialen Konstruktion. Tatsächlich befand sich dort ein zwergwüchsiger (in anderen Ueberlieferungen wird von einem beinamputierten gesprochen) Mensch, welcher das eigentliche «Gehirn» der Maschine darstellte. Eindeutige Betrügerei, wenn auch unterhaltsame.

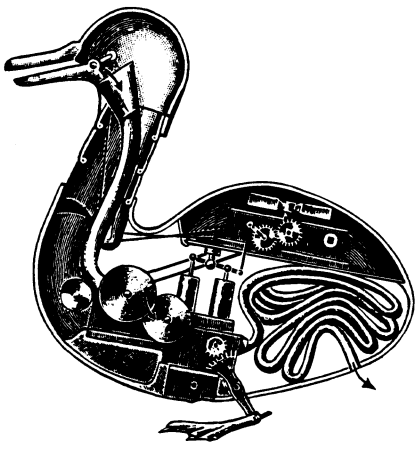


Bild 1: Die mechanische Ente von Jacques de Vaucansons 1738

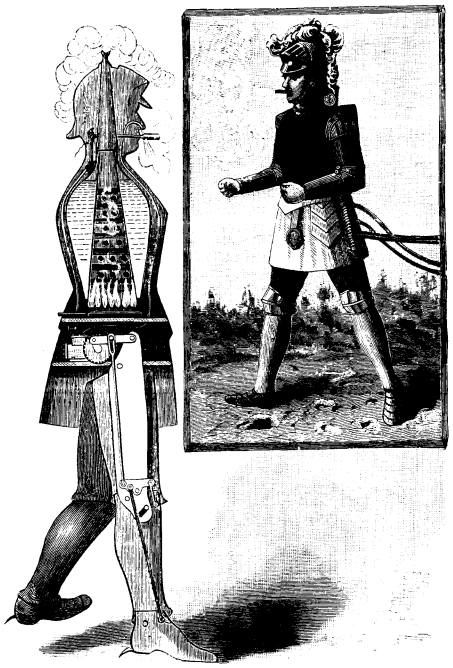


Bild 2: George Moores «Laufende Lokomotive» 1893

Auch aus der Gegenwart sind einige intelligente Maschinen bekannt, welche sich am Beispiel des schachspielenen Türken orientieren.

Menschliche Bewegungsmechanismen liessen sich also bereits sehr früh mehr oder weniger gut simulieren. Der Roboter als solcher ist demnach keine Erfindung unserer modernen Zeit. Bloss ist es früher nicht gelungen, den damaligen auch wirklich gebauten und nicht nur in der Fantasie existierenden Robotern sinnvoll nutzbare Funktionen beizubringen.

Logisch, dass die Spekulationen weiter gingen: Es müsste doch eigentlich irgendwie möglich sein, auch das Denken einem Automaten zu übertragen. Wäre es so nicht zu verwirklichen, in naher Zukunft einen Menschen nachzubilden...? Womit wir uns wieder näher bei Frankenstein befinden.

Streifen wir also noch kurz die historischen Aspekte der Nachbildung menschlichen Denkens. Die zahlreichen Versuche, dem Menschen die Denk-«Arbeit» zu erleichtern, entspringen für einmal nicht dem Geist der Alchimisten. Es waren vor allem

die Mathematiker, welche sich die Knochenarbeit des Rechnens erleichtern wollten. So baute Blaise Pascal 1642 die erste einfache digitale mechanische Rechenmaschine, und damit war auch der erste Schritt zur Entwicklung des Computers getan.

Auch wenn sie durchaus interessant und unterhaltend sein mögen, wollen wir dennoch darauf verzichten, hier historische Begebenheiten um die Geschichte des Computers aufzuzählen. Wer sich dafür interessiert, dem sei das Buch «The computer from Pascal to von Neumann» [1] empfohlen.

Natürliche Intelligenz

Wir sind der Ueberzeugung, dass es weit ausserhalb unserer Kompetenz liegt, sich darüber auszulassen, was menschliche Intelligenz ist und wie diese zustande kommt. Bereits etliche Generationen von Psychologen und Philosophen haben sich über diese Thema den Kopf zerbrochen.

Jedenfalls fällt es viel leichter zu erklären, welche Wirkung intelligentes Verhalten zeigt, als Intelligenz in ihrer Entstehungsform zu beschreiben. Das macht auch verständlich, weshalb sich die Definitionsversuche des Phänomens meist eher auf diese Wirkungen stützen. So Dr. Christopher Evans: «Intelligenz ist die Fähigkeit eines Systems, sich einer veränderlichen Welt anzupassen. Je ausgeprägter und vielseitiger diese Anpassungsfähigkeit ist, um so intelligenter ist das System».

Schlägt man denselben Begriff im Lexikon nach, so wird man nicht weniger überrascht: Intelligenz ist hier die «Fähigkeit eines Lebewesens(!), die Gehirnleistung in Anpassung an veränderte Umstände auf neue Forderungen einzustellen; die Fähigkeit, Bedeutungen und Beziehungen zu erfassen und sinnvoll darauf zu reagieren» oder im Duden: «besondere geistige Fähigkeit, Klugheit; Vernunftwesen». Na also, so einfach ist das.

Mit Bestimmtheit gehört es mit zu den wesentlichsten Aufgaben der «KI-Forschung», ihr Vorbild, die menschliche Intelligenz zu erklären und zu der Erforschung ihrer Funktionsweise beizutragen.

Künstliche Intelligenz

«Computer können nur Tatsachen verarbeiten, aber der Mensch - die Quelle der Tatsachen - ist selbst keine Tatsache oder eine Menge von Tatsachen, sondern ein Wesen, das im Verlauf des Lebens in der Welt sich selbst und die Welt der Tatsachen erschafft. Diese menschliche Welt mit ihren erkennbaren Gegenständen ist organisiert durch Menschen, die ihre verkörperten Fähigkeiten einsetzen, um ihre verkörperten Bedürfnisse zu befriedigen. Es gibt keinen Grund für die Annahme, dass eine Welt, die über diese fundamentalen menschlichen Fähigkeiten organisiert ist, durch irgendwelche anderen Mittel zugänglich sein sollte», schrieb Dreyfus 1985. [4]

Dieses Zitat, diese Absage an die künstlich nachgebildete Menschlichkeit, entstammt somit einem der bekanntesten Kritiker der KI. In seinem Buch «What computer can't do» beschreibt Hubert L. Dreyfus auf die ihm eigene, sehr subjektive Denkweise die Grenzen der künstlichen Intelligenz. Dreyfus Zitat steht für die Meinung mancher KI-Kritiker.

Vergleicht man die Fähigkeiten der Maschine eins zu eins mit denen des Menschen, so ist sehr leicht, zum Schluss zu gelangen, dass KI-Systeme eben doch nichts anderes sind, als dumme, wenn auch sehr schnelle «Rechenmaschinen«.

Wird aber nicht die menschliche Intelligenz in globo, sondern nur einzelne ihrer Fähigkeiten zum Vergleich beigezogen, so bewegen sich die Kritiker bereits auf wesentlich glatterem Boden. So behauptete Dreyfus, dass Computer niemals Schach spielen würden - etwas später wurde er von einer Maschine in eben dieser Disziplin geschlagen.

Daraufhin äusserte er sich in der Art, dass Computer wohl schachspielen, aber niemals Symphonien schreiben können. Wir werden sehen! Und wir sind gespannt, welche Fähigkeit dem Computer als Nächstes abgesprochen wird...

Wer behauptet, dass irgendetwas niemals sein wird, begibt sich immer in Gefahr, das Gesicht zu verlieren - das sollte uns die Geschichte der Entwicklung längst gelehrt haben. Und wer vergleicht, sollte die Relativität nicht ausser acht lassen - das ist gerade bei Betrachtungen zwischen Computern und Menschen besonders wichtig. Schlussendlich ist auch hier ausschlaggebend, mit welchen Ellen man misst. «Die Klugheit des Fuchses wird oft überschätzt, weil man ihm auch noch die Dummheit der Hühner als Verdienst anrechnet», sagte Hans Kasper.

Das maschinelle «Universalgenie» wird allerdings auch noch einige Zeit auf sich warten lassen, erst recht, wenn wir von ihm verlangen, dass es in jeder Hinsicht «menschelt». Die Vorstellung, dass ein Computer auf seinen «Arbeitskollegen» eifersüchtig ist, wenn dieser eine etwas bessere Arbeit erledigen darf, ist zumindest amüsant, wenn auch beängstigend.

Wird es dannzumal erlaubt sein, der Maschine bewusst den Stecker auszuziehen, oder grenzt das bereits an Mord?

Um dem Thema weiterhin im Ernst gerecht zu bleiben, sollten wir uns wieder zurück auf die Erde begeben. Betrachten wir also wiederum eine einzelne Aufgabe der KI. Alan Turing hat einen Test vorgeschlagen, bei welchem sich zwei Systeme über Fernschreiber unterhalten. Auf der einen Seite des «Drahtes» sitzt ein Mensch, auf der anderen eine Ma-

Wichtigste Stationen der KI

1642 Blaise Pascal konstruiert die erste digitale Rechenmaschine

1950 Allan Turing veröffentlich seine Schrift «Computing Machinery and intelligence». In dieser ist der bekannten Turing-Test verankert.

1956 John McCarthy ruft die Dartmouth-Konferenz ins Leben

1959 A.L. Samuel stellt ein Programm vor, welches meisterhaft Dame spielen kann.

1960 Der erste Lisp-Interpreter und das erste Lisp-Handbuch erscheinen

1963 Joseph Weizenbaum entwickelt ELIZA

1965 J. Feigenbaum beginnt mit der Entwicklung eines der ersten Experten-Systeme DENDRAL.

1970 Die ersten intelligenten Roboter werden entwikkelt

1972 A. Colmeraur, R. Kowalsky und P. Roussel entwikkeln an der Universität von Marseille die Programmiersprache Prolog

1980 KI-spezifische Hard- und Software kommt auf den Markt.

1981 Japan gibt seine Pläne bezüglich der 5. Computer-Generation bekannt.

1985 Durch die rassante technische Entwicklung werden die ersten kommerziellen Expertensysteme möglich.

schine. Ist es einem Beobachter nun nicht mehr möglich zu entscheiden, an welchem Ende wer sitzt, muss die Maschine denken können.

Eines der inzwischen bekanntesten «Dialog-Programme» wurde von Joseph Weizenbaum entwickelt. Er nannte es nach Eliza Doolittle aus »My Fair Lady» ELIZA. ELIZA simuliert das Gespräch zwischen einem Psychoanalytiker und seinem Patienten. Tatsächlich wird von anerkannten Wissenschaftlern berichtet, die sich ernsthaft mit ELIZA unterhalten und dabei vergessen haben, dass sie ja eigentlich an eine Wand geredet haben. Die Situation an sich ist überwältigend: Ein Mensch vertraut dem Gerät seine intimsten Geheimnisse an und wird dabei von der Maschine kein bisschen verstanden. Dieses «Insichkehren» mit Hilfe eines imaginären Gesprächspartners weist erschreckende Parallelen mit einem Gebet auf!? Wieder und wieder wird das Thema emotionell geladen.

Definieren wir Künstliche Intelligenz zum Abschluss unserer rückwirkenden Betrachtungen noch einmal mit einer sachlichen Aussage von Edward A. Feigenbaum: «Künstliche Intelligenz ist der Teil der Computerwissenschaft, welcher sich mit der Entwicklung von intelligenten Computersystemen beschäftigt. Das sind Systeme, bei welchen wir den Eindruck gewinnen, sie seien intelligent in der Art, wie es auch der Mensch ist. Dies bedeutet: Das Verständnis von natürlicher Sprache, Lernfähigkeit, Begründungen, Probleme lösen usw. Computer sind dann intelligent, wenn sie Probleme zu lösen vermögen, welche vom Menschen Intelligenz erfordern». [5]

Die Quintessenz dieser Aussage beruht darauf, dass es nicht das Ziel der KI sein kann, ein Abbild des Menschen zu schaffen. Deshalb ist es eigentlich völlig uninteressant, von der Maschine zu verlangen, dass sie den Menschen mit all seinen Makeln nachahmt.

Die ernsthafte KI-Forschung wird sich eher damit beschäftigen, den Computer auf speziellen Gebieten noch leistungsfähiger zu machen und so dem Menschen ein noch stärkeres, ihm vielleicht unglaublich ähnliches Werkzeug in die Hand zu geben.

Die faszinierende Thematik der Künstlichen Intelligenz verlangt nach Auffassung der Autoren bisweilen bewusst angriffige, manchmal für einige Leser vielleicht schockierende Aussagen, um sich der Angleichung des simulierten Verhaltens an die menschliche Realität bewusst zu werden.

Solche Aussagen müssen sich nicht unbedingt mit der persönlichen Einstellung der Autoren decken. Sich aufdrängende Assoziationen wollen aber beim Wort genannt sein, wenn sie zum Nachdenken anregen sollen. KI spielt im weitesten Sinne mit allen ihren Auswirkung so sehr in die ganz persönlichen Angelegenheiten, ja sogar in die Intimssphäre einzelner Menschen hinein, dass es nicht vertretbar wäre, Realitäten aus ethischer Rücksichtnahme totzuschweigen.

Ein wirkliches Bild kann nur gewinnen, wer sich objektiv und ohne Vorbehalte mit der Sache auseinandersetzt. Das wird sich auch immer wieder in den bevorstehenden praktischen Anwendungen bestätigen.

Nichts geht über die Praxis

Nichts ist so verheerend, wie jemand, der eine Meinung über etwas predigt, von dem er selber überhaupt nichts versteht - und kaum etwas ist so mühsam zu verstehen, wie Berge von grauer Theorie. Deshalb möchten wir uns auch nicht in die Reihe der Computerbücher einfügen, in denen man sich über zahllose Seiten und damit in unserem Falle mehrere Heft-Folgen zum ersten Tastendruck auf dem Computer «vorarbeiten» muss.

Aktive Mitarbeit, das sei hier nochmals betont, ist ausdrücklich erwünscht, deshalb wollen wir uns jetzt an den Computer wagen. Für KI-Anwendungen eignen sich am besten Aufgaben, für welche ein Algorithmus nicht ohne weiteres auf der Hand liegt oder für die überhaupt kein solcher existiert. Das nachfolgende Problem dürfte allgemein bekannt sein. Es handelt sich dabei um ein Spiel, bei welchem sich Zahlen in einer Matrix derart verschieben lassen, dass jeweils ein Element orthogonal (nach rechts, links, oben oder unten) in einen Freiraum gebracht wird.

4	7	2
6	5	8
	1	3

Skizze 1

1	2	3
4	5	6
	7	8

Skizze 2

Die triviale Lösung dieser Aufgabe besteht darin, mit Hilfe eines Zufallsgenerators die Elemente andauernd zu verschieben, bis sie sich an der gewollten Position befinden. Mit Hilfe der Kombinatorik lässt sich zeigen, dass für dieses Spiel theoretisch 9*8*7*6*5*4*3*2*1 = 9! oder 362'880 mögliche Spielpositionen ergeben können. Folglich ist leicht einzusehen, dass man selbst mit einem leistungsfähigen Computer die richtige Lösung nicht innert nützlicher Frist finden wird. Allerdings lässt sich bei dieser Art von Problemlösungs-Strategie bestimmt auch nicht von Intelligenz sprechen.

Ueberlegen wir uns, wie wir diese Situation verbessern können. Es wäre wünschenswert, wenn der Rechner ebenso wie der Mensch, unsinnige Züge vermeiden könnte. Das Problem scheint sehr einfach lösbar zu sein. Wenn wir dem Computer eine Möglichkeit zur Bewertung seiner Züge geben, so wird er diese schon wesentlich zielbewusster einsetzen und die gewünschte Position mit grösserer Wahrscheinlichkeit finden.

Computer-Programme, die so «berechnend» vorgehen, nennt man heuristisch. Im Gegensatz zur Lösungsfindung mit Hilfe des Zufalls besteht hier jedoch die Möglichkeit, dass sich der Computer in einer endlosen Schleife verfängt.

Jetzt wählt nämlich der Rechner nach einem strengen Bewertungskriterium den jeweils «besten» Zug aus einer Stellung aus. Daraus kann sich leicht ergeben, dass er nach einigen Zügen wiederum dieselbe Ausgangsposition erreicht. Logischerweise wird er daraufhin alle vorhergehenden Züge solange immer wiederholen, bis man ihn unterbricht.

Durch Unordnung zum Ziel

Ist er in eine solche Schleife geraten, sollte der Computer, wieder ebenso wie der Mensch es tun würde, nach einem anderen erfolgsversprecherenden Weg suchen. Bei uns geschieht dies meist durch Intuition, welche einen wesentlichen Bestandteil unserer Intelligenz bildet. Man gelangt sozusagen durch eine gewollte Unordnung vieler Denkanstösse zum Ziel - vergleichbar etwa mit einem Brainstorming.

Im vorliegenden, sehr einfachen BASIC-Programm helfen wir dem Rechner dadurch zur Ideenfindung, indem wir ihn nach einer gewissen Anzahl von Zügen veranlassen, nicht mehr gezielt, sondern für einige weitere Züge zufällig vorzugehen. Durch

```
 10 REM KI-DEMO IN BASIC NR.1
 20 DIM B(9)
 30 DIM M$(9)
 40 M=0
 50 Q=0
 60 Q1=0
 70 GOSUB 3000
 80 GOSUB 1000
 90 GOSUB 2000
100 GOSUB 4000
110 GOSUB 5000
120 IF S=0 THEN GOTO 190
125 IF (M+1) MOD 50 =0 THEN GOSUB 10000
130 PRINT
140 GOSUB 8500
150 K=J1:J=J1:P=P1:Q1=P1
160 GOSUB 9000
170 GOSUB 7000
180 P1=P:M=M+1:GOSUB 2000
185 GOTO 110
190 PRINT "Problem geloest in ";M;" Versuchen !"
200 PRINT
210 PRINT "Wollen Sie noch einmal probieren? (j/n) ";
220 INPUT A$
230 IF A$ ="n" THEN END
240 GOTO 40

250 REM MISCHEN
1000 FOR I=1 TO 9
1010 B(I)=I
1020 NEXT I
1030 P=9
1040 N=INT(RND(1)*10)+80
1050 FOR Z=1 TO N
1060 GOSUB 6000
1070 GOSUB 7000
1080 NEXT Z
1090 Q=0
1100 RETURN

1200 REM AUSGEBEN DER MATRIX
2000 PRINT
2010 FOR I=1 TO 9
2015 IF B(I)=9 THEN PRINT"   ";
2020 IF B(I)<>9 THEN PRINT B(I);"";
2030 IF I=INT(I/3)*3 THEN PRINT
2040 NEXT I
2050 RETURN

2060 REM INITIALISIERUNG
3000 M$(1)="24"
3010 M$(2)="135"
3020 M$(3)="26"
3030 M$(4)="157"
3040 M$(5)="2468"
3050 M$(6)="359"
3060 M$(7)="48"
3070 M$(8)="579"
3080 M$(9)="68"
3090 RETURN

3100 AUFFINDEN DER LEERPOSITION
4000 FOR I=1 TO 9
4010 IF B(I)=9 THEN P1=I
4020 NEXT I
4030 RETURN

4040 REM KONTROLLE
5000 S=0
5010 FOR I=1 TO 9
5020 IF I<>B(I) THEN S=1
5030 NEXT I
5040 RETURN
```

diese Strategie ergibt sich für den Computer eine neue Spielsituation (sofern die Zufallszüge auch tatsächlich zu einer relevanten Veränderung der Matrix führen).

Wenn wir Zufall und Heuristik auf diese Weise kombinieren, lässt sich mit Bestimmtheit voraussagen, dass das Programm immer eine Lösung des Problems finden wird. Diese wird zwar bestimmt nicht auf dem kürzesten Weg erreicht werden, aber immerhin gelangen wir zum Ziel.

Das Listing zeigt das nach diesen Prinzipien aufgebaute BASIC-Programm. Es entstammt einer Idee aus dem Buch «Artificial Intelligence in BASIC» von Mike James. [6]

Das Programm geht ungefähr wie folgt vor: Unterprogramm 1000 mischt zu Anfang das Spielbrett, 2000 gibt die aktuelle Situation aus, 3000 bezeichnet die erlaubten Züge, 4000 findet die Leerpositionen, 5000 überprüft die Gewinnposition, 6000 macht einen Zufallszug, 7000 schiebt das ausgewählte Element in die Leerposition, 8500 berechnet zwei Züge im voraus den bestmöglichen Zug, 9000 enthält die Heuristik, 10000 löst zehn zufällige Züge aus.

Wenn Sie den Rechner bei der Lösung seiner Aufgabe beobachten, werden Sie sich sehr oft fragen, warum er wohl derart ungeschickt vorgeht. Wie aus der Beschreibung der Unterprogramme ersichtlich ist, wird die Spielsituation nur auf die folgenden zwei Züge berechnet. Auch die eingesetzte Heuristik weist noch erhebliche Mängel auf.

Eine Verlangsamung des Programmablaufes lässt sich durch eine Warteschleife nach der Zeile 180 erreichen, z.B.:

```
182 REM WARTESCHLEIFE
183 FOR WS=1 TO 10000:NEXT WS
```

Die Zahl 10000 oder je nach Computer eine grössere oder kleinere Zahl ist verantwortlich für die Dauer der Wartezeit und damit für die Ablaufgeschwindigkeit.

Dieses Problembeispiel liesse sich durchaus auch mit Hilfe eines Algorithmus lösen. Vermutlich ist aber dann die Lösung wesentlich aufwendiger. Als Gedankenanstoss könnte man sich eine Lösung vorstellen, bei welcher für jedes Element der Matrix ein entsprechender Lösungsweg programmiert wird.

Falls sich unter den Lesern «Computer-Freaks» befinden, so möchten wir sie gerne dazu auffordern, sich zu dieser Gesamtproblematik weitere Gedanken zu machen und uns even-

tuelle Programm-Lösungen zuzusenden (sowohl algorithmische wie auch solche, wie sie hier beschrieben worden sind). Wir würden diese gerne in einer späteren Folge besprechen und eventuell abdrucken.

Mit diesem Beispiel ist es uns gelungen, selbst in der dafür nur bedingt geeigneten Hochsprache BASIC den Computer so etwas wie ein bisschen menschliches Verhalten beizubringen. Im nächsten Heft, wenn wir die Programmiersprachen der KI betrachten, werden wir uns darüber Gedanken machen, wie man solche Probleme mit Hilfe dieser mächtigen Werkzeuge elegant lösen kann.

Mit Prolog experimentieren

Wer schon jetzt experimentieren und sich auf das faszinierende Thema etwas vorbereiten möchte, dem bieten die vom Basler Ingenieurbüro Kibotrona zum Thema erarbeiteten Disketten eine attraktive Möglichkeit. Sie können diese Disketten mit der am Schluss des Heftes beigehefteten Karte «Disketten-Service» anfordern und bekommen dadurch sofort Kontakt zu «Prolog», einer Computer-Programmiersprache der fünften Generation, wie sie für KI-Programme verwendet wird, und die ab der nächsten Ausgabe intensiv genutzter Bestandteil unserer Betrachtungen sein wird.

Diese Disketten enthalten im Paket 1 Beispiele von Prolog-Programmen, ein kleines Beispiel eines Expertensystemes, PD-Prolog (ein Interpreter, der die Programme ohne zusätzliche Software auf MS-DOS-(IBM-Kompatiblen) Computern lauffähig macht), einen komfortablen, in Turbo-Prolog geschriebenen und selbständig auf DOS-Ebene lauffähigen Editor, mit dem sich Programme schreiben oder verändern lassen, ein ausführliches Handbuch in deutscher Sprache. Die Beispielprogramme zu den Erklärungen können direkt ab der Diskette geladen werden.

Wer Wert auf noch mehr Editier-Komfort legt, kann im Paket 2 ein professionelles, übersetztes PD-Textverarbeitungsprogramm mitbeziehen, mit dem sich auch sehr komfortabel Briefe und andere Texte schreiben lassen (mit zusätzlich erstelltem, deutschsprachigem Handbuch auf der Diskette).

Zum Verständnis unseres KI-Lehrganges sind diese Disketten nicht unbedingt notwendig. Sie sind aber so gestaltet, dass der Interessierte sofort damit zurechtkommen und sich mit der neuen Materie ein wenig vertraut machen kann.

```
5050 REM ZUFAELLIGES VERSCHIEBEN
6000 I=INT(RND(1)*LEN(M$(P)))+1
6010 J=VAL(MID$(M$(P),I,1))
6020 RETURN

6030 REM AUSGEWAEHLTE POSITION MIT LEERPOSITION AUSTAUSCHEN
7000 T=B(P)
7010 B(P)=B(J)
7020 B(J)=T
7030 Q=P
7040 P=J
7050 RETURN

7060 REM VERSCHIEBEN MIT HEURISTIK
8000 C=-5
8010 FOR I=1 TO LEN(M$(P))
8020 K=VAL(MID$(M$(P),I,1))
8030 IF K=Q THEN GOTO 8060
8040 GOSUB 9000
8050 IF E>C THEN J=K:C=E
8060 NEXT I
8070 RETURN
8500 C1=-5
8510 FOR Z=1 TO LEN(M$(P))
8520 J0=VAL(MID$(M$(P1),Z,1))
8530 IF Q1=J0 THEN GOTO 8630
8540 J=J0:P=P1:K=J
8550 GOSUB 9000
8560 CT=E
8570 GOSUB 7000
8580 GOSUB 8000
8590 CT=CT+C
8600 IF CT>C1 THEN J1=J0:C1=CT
8610 J=P1:P=J0
8620 GOSUB 7000
8630 NEXT Z
8640 RETURN
9000 E=ABS(P-B(K)-INT((P-1)/3)*3+INT((B(K)-1)/3)*3)
9010 E=E+ABS(INT((P-1)/3)-INT((B(K)-1)/3))
9020 F=ABS(K-B(K)-INT((K-1)/3)*3+INT((B(K)-1)/3)*3)
9030 F=F+ABS(INT((K-1)/3)-INT((B(K)-1)/3))
9040 E=F-E
9050 RETURN
9060 REM ZUFAELLIGES VERSCHIEBEN
10000 PRINT
10010 FOR II=1 TO 10
10020 M=M+1
10030 GOSUB 6000
10040 GOSUB 7000
10050 GOSUB 2000
10060 NEXT II
10070 GOSUB 4000
10080 RETURN
```

Gegenüber anderen Programmiersprachen gibt es hier soviel Neues, dass eine gewisse Vorbereitung und Einspielung von grossem Nutzen sein kann. Ausserdem wird in der Folge für die Besitzer der Disketten laufend auf erklärende oder ergänzende Beispiele und Sequenzen verwiesen.

Prolog eröffnet übrigens Möglichkeiten, die bisher im Bereich der Personal-Computer noch sehr wenig ausgeschöpft worden sind. Exklusives liegt hier noch nicht in den Sternen - wir wünschen Mut und Erfolg bei der Erforschung!

Literatur

[1] «The computer from Pascal to von Neumann» Princeton University Press NY

[2] Christopher Evans: «The might Micro» London

[3] Pamela McCorduck: «Machines who thinks»

[4] Hubert L.Dreyfuss: «Die Grenzen künstlicher Intelligenz; «Was Computer nicht können»

[5] Avron Barr & E.A. Feigenbaum: «The handbook of artificial intelligence»

[6] Mike James: «Artificial Intelligence in BASIC»

Ausblick auf die nächsten Folgen

Anhand der KI-Sprache «Prolog» werden wir uns schon in der nächsten Folge mit einfachen Beispielen von KI-Programmen auseinandersetzen und selber in diesem Rahmen beginnen zu beurteilen, wie weit es denn nun mit der damit dem Computer einverleibten «Intelligenz» her ist.

Dem besseren Verständnis zuliebe werden wir uns ein paar mathematische Hintergründe ansehen. Weil die KI sich überall breit macht, in jüngerer Zeit vielleicht vergleichbar mit dem Computer überhaupt und etwas früher mit der Schreibmaschine, werden wir uns später auch mit der Sprach- und Bilderkennung, und den damit sehr eng zusammenhängenden Anwendungen in der Robotik und in CIM, dem «Computer Integrated Manufacturing», oder auf Deutsch der vollautomatisierten Produktion, beschäftigen. «Intelligente» Computerspiele, darin aufsteigend Expertensysteme, weitere KI-Sprachen wie Lisp und der Ausblick auf die Zukunft der KI - alles mit einem immer aktuellen Diskettenservice - werden Sie in Spannung halten. □

COMPUTER-SPLITTER

IBM und UNIX

(572/fp) IBM unterstützt UNIX in den vor einem Jahr verkündeten SAA zwar nicht und neben SAA vorbei auch nur halbherzig. Doch an UNIX kommt nun niemand mehr vorbei. Deshalb kündigt IBM das auf UNIX basierende AIX (Advanced Interactive Executive) an. Es wird vorerst auf speziellen Engineering PC/RT als AIX/RT angeboten, später für PS/2, Modell 80 und dann für die Grossen aus den Familien 9370, 4381 und 3090. Wo AIX mit SAA kollidiert, werden die UNIX-Spezifikationen Vorzug haben.

Einführung in Turbo BASIC (2)

Turbo BASIC wurde bewusst aufwärts kompatibel zu BASICA der IBM-PCs und zu GWBASIC der IBM-Kompatiblen geschrieben. Man wollte damit erreichen, dass die Vielzahl von Programmen, die im interpretativen BASIC erstellt wurden, auch unter Turbo BASIC laufen. Wie man solche Programme im ASCII-Code speichert und anschliessend in Turbo BASIC abruft, compiliert, linkt und fährt, wurde im 1. Kapitel unserer Einführung in Turbo BASIC (M+K 88-2) genau erklärt.

Marcel Sutter

2. Kapitel: Die Sprachelemente von Turbo BASIC (1)

Die nachfolgende Uebersicht über alle reservierten Wörter von Turbo BASIC zeigt, dass nahezu alle Wörter von BASICA und GWBASIC als Teilmenge enthalten sind. Die Wörter wurden thematisch gegliedert, wobei ein * nach dem Wort ein erst in Turbo BASIC dazu gekommenes Sprachelement bezeichnet.

Ausgabe

PRINT		gibt Daten auf dem Bildschirm aus
PRINT USING		gibt formatierte Daten auf dem Bildschirm aus
PRINT SPC (n)		gibt n Leerzeichen auf dem Bildschirm aus
PRINT TAB(n)		springt direkt zur n. Spalte vom linken Schirmrand
WRITE		schreibt Zahlen durch Kommas getrennt und String in Anführungszeichen

Compiler-Befehle:

$COM	*	belegt Speicherplatz für die Puffer serieller Ports
$DYNAMIC	*	setzt den Array-Standardtyp auf «dynamisch»
$EVENT	*	kontrolliert die Erzeugung von Code für Ueberwachungen
$IF/$ELSE/ $ENDIF	*	bedingte Compilierung von Programmteilen
$INCLUDE	*	fügt Quelltext von anderen Dateien ein
$SEGMENT	*	legt Codesegment-Grenzen fest
$SOUND	*	belegt Speicherplatz für den Puffer für die Anweisung PLAY
$STACK	*	belegt Speicherplatz für den Stack des Prozessors
$STATIC	*	setzt den Array-Standardtyp auf «statisch»

Compiler-Daten

CLEAR		setzt statische Variablen und Arrays zurück, löscht dynamische Variablen und Arrays
DATA		markiert den Beginn einer DATA-Zeile mit Daten
DIM		dimensioniert ein Array
ERASE	*	setzt statische Arrays zurück, löscht dynamische Arrays
LOCAL	*	deklariert lokale Variablen einer Prozedur/Funktion
OPTION BASE		setzt Untergrenze für den Index eines Arrays
SHARED	*	deklariert globale Variablen innerhalb einer Prozedur/Funktion
STATIC	*	deklariert statische Variablen einer Prozedur/Funktion
READ		liest die Daten einer DATA-Zeile
RESTORE		setzt den DATA-Zeiger für READ

Dateien

BLOAD		lädt eine mit BSAVE gespeicherte Datei
BSAVE		schreibt den Inhalt eines Speicherbereichs als Datei
CLOSE		schliesst eine Datei oder ein Peripheriegerät
EOF		prüft, ob das Ende der Datei erreicht ist
FIELD		definiert Dateipuffer und Felder für Random-Datei
GET		liest einen Record einer Random-Datei
GET$	*	liest Daten einer BINARY-Datei in eine Stringvariable
INPUT		liest ein Element aus einer sequentiellen Datei
LINE INPUT		liest eine ganze Zeile aus einer sequentiellen Datei
LOC		liefert die Position innerhalb einer Datei zurück
LOF		liefert die Länge einer Datei zurück
LSET		schreibt Stringdaten linksbündig in den Random-Datei-Puffer
OPEN		eröffnet eine Datei oder ein Peripheriegerät
PRINT		schreibt Daten in eine sequentielle Datei
PRINT USING		schreibt formatiert in eine sequentielle Datei
PUT		schreibt einen Record in eine Random-Datei
PUT$	*	schreibt einen String in eine BINARY-Datei
RSET		schreibt Stringdaten rechtsbündig in den Random-Datei-Puffer
RESET		schreibt Dateipuffer und schliesst alle offenen Dateien
SEEK	*	setzt die Position in einer BINARY-Datei für GET$ und PUT$
WRITE		schreibt Daten in eine sequentielle Datei

DOS-Befehle

CHDIR	wechselt das Standard-Directory (bei der Harddisk)
FILES	gibt das Inhaltsverzeichnis einer Diskette aus
KILL	löscht eine Datei
MKDIR	erzeugt eine neue Subdirectory (bei der Harddisk)

NAME		gibt einer Datei einen neuen Namen
RMDIR		löscht ein leeres Subdirectory (bei der Harddisk)
SYSTEM		beendet ein Programm und steigt ins DOS ab

Drucker

LPOS		liefert die «Cursorpostion» im Drukkerpuffer zurück
LPRINT		schreibt Daten auf dem Drucker statt auf dem Schirm
LPRINT USING		schreibt die Daten formatiert auf dem Drucker
WIDTH		setzt die Zeilenbreite (40 oder 80 Spalten)

Eingabe

INKEY$		liest ein Einzelzeichen von der Tastatur
INPUT		verlangt vom Benutzer eine Eingabe über die Tastatur
INPUT$		liest eine bestimmte Anzahl Zeichen von der Tastatur
INSTAT	*	liefert den Status der Tastatur zurück (Wurde eine Taste gedrückt?)
KEY		definiert die Zuordnungen von Strings zu Tasten
LINE INPUT		liest eine ganze Zeile von der Tastatur

Fehlerbehandlung

ERADR	*	liefert die Adresse des jeweils letzten Fehlers
ERROR		simuliert Laufzeitfehler
ON ERROR GOTO		verzweigt zu einer Fehlerbehandlung
RESUME		setzt das Programm nach einem Fehler fort

Grafik

CIRCLE		zeichnet einen Kreis, Kreisbogen oder Sektor
CLS		löscht den Grafikbildschirm
COLOR		setzt die Zeichen- und Hintergrundfarbe
DRAW-Befehle		LOGO-ähnliche Befehle zum Zeichnen von Objekten
GET		liest einen Abschnitt des Grafikschirms in ein numerisches Array
LINE		zieht eine gerade Linie zwischen zwei Punkten
PAINT		malt einen umschlossenen Bereich in einer Farbe aus
PALETTE		bestimmt die Zuordnung der Nummern zu den Farben (nur EGA)
PEN-Funktion		gibt den Status des Light-Pens zurück
PEN-Befehl		aktiviert den angeschlossenen Light-Pen
PMAP		Umrechnung von mathematischen in Bildschirm-Koordinaten und umgekehrt
POINT		liefert die Farbnummer eines Pixels zurück
PRESET		löscht Pixel auf dem Grafikschirm
PSET		zeichnet Pixel auf den Grafikschirm
PUT		kopiert den Inhalt eines numerischen Array auf den Grafikschirm
SCREEN		setzt die Auflösung des Schirms (Videomodus)
VIEW		legt eine Zeichenfläche fest
WINDOW		definiert das grafische Koordinatensystem

Musik

BEEP n	*	erzeugt n Pieptöne
PLAY-Befehl		erzeugt Töne und Melodien
PLAY-Funktion		liefert die Notenzahl im PLAY-Puffer zurück
SOUND		erzeugt einen Einzelton bestimmter Frequenz und Dauer

Numerisches

ABS(X)		absoluter Betrag von X
ASC		liefert den ASCII-Code des 1. Zeichens eines Strings
ATN(X)		Arcustangens von X
CDBL		konvertiert Zahl in doppeltgenaue Realzahl
CEIL	*	liefert die nächstgrössere Integerzahl zurück
CINT		konvertiert Zahl in eine Integerzahl
CLNG	*	konvertiert Zahl in eine Langintegerzahl
COS(X)		Cosinus des Winkels X
CSNG		konvertiert Zahl in eine einfachgenaue Realzahl
CVI/CVL/ CVS/CVD		konvertieren Stringzahlen aus dem Random-Dateipuffer in numerische Zahlen
CVMD/CVMS	*	konvertieren Stringdaten einer Random-Datei vom Microsoft-Format in das IEEE-Format für Realzahlen
DECR J,5	*	erniedrigt die Variable J um 5
DEFINT/DEFLNG/ DEFSNG/DEFDBL		definieren den Typ von Variablen
EXP(X)		Exponentialfunktion von X
EXP10(X)	*	ergibt 10 hoch X
EXP2(X)	*	ergibt 2 hoch X
FIX		schneidet die Kommastellen ab
INCR J,3	*	erhöht die Variable J um 3
INT		rundet die Dezimalzahl auf die nächstkleinere ganze Zahl
LOG(X)		Logarithmus von X zur Basis e
LOG2(X)	*	Logarithmus von X zur Basis 2
LOG10(X)	*	Logarithmus von X zur Basis 10
MKI$/MKL$/ MKS$/MKD$		konvertieren numerische Daten in Stringdaten für Random-Dateien
MKMS$/MKMD$	*	konvertieren numerische Daten vom IEEE-Format in Stringdaten im Microsoft-Format für Random-Dateien
RANDOMIZE		setzt Startwert für den Zufallszahlengenerator
RND		liefert eine Zufallszahl zurück
SGN(X)		liefert das Signum der Zahl X
SIN(X)		Sinuswert des Winkels X
SQR(X)		Quadratwurzel aus X
TAN(X)		Tangens des Winkels X

Programmlauf

CALL	*	ruft eine Prozedur auf
CALL ABSOLUTE	*	ruft ein Maschinenprogramm auf
CALL INTERRUPT	*	führt einen System-Interrupt durch
DEF FN/END DEF	*	definiert eine mehrzeilige Funktion
DO/LOOP	*	Schleifenkonstrukt
END		beendet das Programm oder eine Blockdefinition
EXIT	*	verlässt eine Blockstruktur vor ihrem Ende
FOR/NEXT		Schleifenkonstrukt
GOSUB		ruft ein Unterprogramm (keine Prozedur) auf
GOTO LABEL	*	springt zur Zeile unmittelbar unter der Marke LABEL
IF/THEN/ELSE	*	einzeiliges Verzweigungskonstrukt
IF(Block)/END IF		mehrzeiliges Verzweigungskonstrukt mit Blockstruktur
ON (n) GOSUB		ruft Unterprogramme in Abhängigkeit von n auf
ON (n) GOTO		springt in Abhängigkeit von n zu einem Label
RETURN		Rückkehr aus einem Unterprogramm
SELECT CASE n	*	Auswahl in Abhängigkeit von n
STOP		bricht das Programm ab
SUB/END SUB	*	Beginn und Ende einer Prozedur
WHILE/WEND		Schleifenkonstrukt

Speicherverwaltung

DEF SEG		definiert das Segment für BSAVE und BLOAD
ENDMEM		liefert die Endadresse eines Speichers zurück
FRE		liefert den freien Speicherplatz zurück
MEMSET	*	reserviert Bereiche oberhalb des Programms
PEEK		liest den Inhalt einer Speicherzelle
POKE		schreibt einen Wert in eine Speicherzelle
VARPTR		liefert die Adresse einer Variablen zurück
VARPTR$		liefert einen Zeiger zu einer Variablen als String
VARSEG	*	liefert die Segmentadresse einer Variablen

Strings

BIN$	*	liefert die Binädarstellung einer Zahl als String
CHR$		liefert das Zeichen mit dem betreffenden ASCII-Code
DEFSTR		definiert Variablen als Stringvariablen
HEX$		liefert die Hexadezimaldarstellung einer Zahl als String
INSTR		sucht einen String nach einer bestimmten Zeichenfolge ab
LCASE$	*	verwandelt Grossbuchstaben in Kleinbuchstaben
LEFT$		liefert aus einem String die n linken Zeichen
LEN		liefert die Länge eines Strings
MID$		liefert aus einem String n Zeichen ab einer bestimmten Stelle k
OCT$		liefert die Oktaldarstellung einer Zahl als String
RIGHT$		liefert aus einem String die n rechten Zeichen
SPACE$		liefert einen String mit n Leerzeichen
STR$		wandelt eine Zahl in einen String um
STRING$		liefert einen String, der aus n Wiederholungen eines bestimmten Zeichens besteht
UCASE$	*	verwandelt Kleinbuchstaben in Grossbuchstaben
VAL		wandelt einen String in eine numerische Zahl um

Textbildschirm

CLS		löscht Textbildschirm
COLOR		bestimmt die Farbe zum Schreiben, den Hintergrund und den Rahmen
CSRLIN		liefert die Zeilennummer der Cursorpostion zurück
LOCATE		positioniert den Cursor auf dem Bildschirm
POS		liefert die Spaltennummer der Cursorpostion zurück

Ueberwachung von Zuständen

KEY(n), ON COM(n), ON KEY(n) ON PEN, ON PLAY, ON STRIG(n), ON TIMER
STICK, STRIG-Funktion und STRIG-Befehl

Die genaue Bedeutung dieser Spezialbefehle, die vor allem bei Computerspielen zum Einsatz kommen, entnehmen Sie bitte dem Handbuch zu Turbo BASIC, Band 2.

Verketten von Programmen

CHAIN	*	lädt eine .TBC-Datei und setzt das Programm mit ihr fort. COMMON-Variablen werden übergeben
COMMON	*	deklariert gemeinsame Variablen zweier Programme
RUN		startet ein Programm neu bzw. lädt eine .TBC-Datei ohne Uebergabe von COMMON-Variable

Verschiedenes

DELAY n	*	Zeitverzögerung um n Sekunden
INLINE	*	definiert Maschinencode innerhalb eines Programms
REG	*	setzt oder liest die Inhalte des Prozessor-Register-Puffers
REM		leitet einen Kommentar ein
SWAP		vertauscht die Inhalte zweier Variablen
TRON/TROFF		schaltet den Trace-Modus ein/aus

Die selten gebrauchten Befehle COM (n), INP, IOCTL, IOCTL$, OPEN COM, OUT, und WAIT im Umgang mit Peripheriegeräten lassen wir weg.

Wenn Sie die Tabelle aufmerksam gelesen haben, werden Sie feststellen, dass folgende Befehle aus dem interpretativen BASIC fehlen:

AUTO, CONT, DELETE, EDIT, LIST, LLIST, LOAD
MERGE, MOTOR, NEW, RENUM, USR

Einige Befehle sind überflüssig, da sie in dem Rollmenü von File aus dem Hauptmenü von Turbo BASIC integriert sind und über Tastendruck ausgelöst werden. Andere wieder entfallen, da der Texteditor bessere Möglichkeiten anbietet und da im Texteditor die Programme ohne Zeilennummern geschrieben werden.

Wir wollen im Folgenden jene neuen Sprachelemente von Turbo BASIC behandeln, die vorallem der besseren Strukturierung von BASIC-Programmen dienen und die das Konzept von Prozeduren mit lokalen Variablen ermöglichen. Auf Compiler-Befehle und andere exotische BASIC-Anweisungen treten wir in dieser Einführung nicht ein.

Programmstruktur

Jedes Turbo BASIC-Programm gliedert sich in ein Hauptprogramm und beliebig viele Prozeduren und benutzerdefinierte Funktionen.
Die Programme werden modularisiert und durch Einrückungen optisch gegliedert.
Die einzelnen Zeilen eines Programms müssen eine der drei nachfolgenden Formen haben:

1. Statement : Statement : .... : Statement 'Kommentar

2. Label:

3. $Compiler-Befehl

Merke: Ein Label (Einsprungmarke) muss auf einer Zeile allein stehen und unmittelbar nach dem Namen einen Doppelpunkt haben.
Ein Compilerbefehl muss immer allein auf einer Zeile stehen.
Schreiben Sie nur so viele Statements auf eine Zeile, dass die Bildschirmbreite von 80 Zeichen nicht überschritten wird. Falls Sie wirklich einmal eine lange Anweisung schreiben müssen, dann können Sie diese mit Hilfe des Unterstreiche-Zeichens (_) auf mehrere Zeilen verteilen.

Beispiel: (siehe Listing 6B aus dem 1. Kapitel)

```
z = a(j1,k1)+a(j1,k)+a(j1,k2)+_
    a(j,k1)+a(j,k2)+_
    a(j2,k1)+a(j2,k)+a(j2,k2)
```

Es zeugt von gutem Programmierstil, wenn man pro Zeile nur eine Anweisung schreibt. Die neuen Schleifenkonstrukte sowie das blockstrukturierte IF erleichtern diese in Pascal längst übliche Schreibweise ungemein.

8. Zahlen, Variablen, Arrays, Datentypen und ihre Vereinbarung

Turbo BASIC kennt im Gegensatz zum interpretativen BASIC vier verschiedene Typen von Zahlen:

Integer	Bereich von -32768 bis +32767. Zur Speicherung dieser ganzen Zahlen werden lediglich 2 Byte benötigt.
Langinteger	Bereich von -2'147'483'648 bis +2'147'483'647. Zur Speicherung dieser ebenfalls ganzen Zahlen werden 4 Byte benötigt.
Einfachgenaue Realzahl	Bereich von -10 hoch 38 bis +10 hoch 38. Zur Speicherung dieser mit einem Dezimalpunkt versehenen Zahlen sind 4 Byte nötig. Es werden nur 6 Ziffern der Mantisse gespeichert. Längere Zahlen werden auf 6 Stellen gerundet!
Doppeltgenaue Realzahl	Bereich von -10 hoch 308 bis +10 hoch 308. 8 Byte sind zur Speicherung nötig. Es werden von diesen mit einem Dezimalpunkt versehenen Zahlen 16 Ziffern der Mantisse gespeichert.

Wenn Sie keinen Coprozessor einsetzen, nimmt die Rechengeschwindigkeit von oben nach unten zu. Vorallem das Rechnen in Double Precision kostet immens Zeit. Wenn Sie aber einen Coprozessor einsetzen, dann wird die Geschwindigkeitslücke zwischen Rechnungen mit Integerzahlen und doppeltgenauen Realzahlen nahezu geschlossen! Das hängt damit zusammen, dass der Coprozessor im Gegensatz zum normalen Prozessor intern alle Rechenoperationen mit doppelter Genauigkeit ausführt. Wir empfehlen Ihnen, wenn Sie mit Turbo BASIC wirklich eine echte Verkürzung der Laufzeit Ihrer Programme anstreben, einen offiziellen Coprozessor zu kaufen. Kaufen Sie keinen Clone, da bei diesen meistens Schwierigkeiten auftreten.
In Turbo BASIC gibt es numerische Konstanten, Stringkonstanten und neu sogenannte benannte Konstanten.

Beispiele:

0, -35, 3.14159, -1.25E03, 123456789, usw.
sind numerische Konstanten.

«Computer Verlag», «Was gibt 12*23?», usw.
sind Stringkonstanten.

```
%altersgrenze = 65
for j=1 to %altersgrenze
    read x(j)
next j
```

Benannte Konstanten erkennt man daran, dass **vor** ihrem Namen das Prozentzeichen steht. Der Wert einer benannten Konstante ändert sich im Gegensatz etwa zum Wert einer Variablen nicht. Man verwendet benannte Konstanten, um mit einem Wort die Bedeutung einer Zahl zu erhellen.

&B1101	Binäre Konstante, stellt die Zahl 13 dar.
&O137	Oktale Konstante, stellt die Zahl 95 dar.
&H3F	Hexadezimale Konstante, stellt die Zahl 63 dar.

Turbo BASIC kennt fünf Typen von Variablen. Wenn Sie die Variablen weder explizit noch implizit deklarieren, ordnet Turbo BASIC wie auch das interpretative BASIC allen Variablen den Typ «real» zu. Wir empfehlen Ihnen dringend,

- entweder am Kopf des Programms die Variablen implizit zu definieren (sog. Vereinbarungsteil in Pascal)
- oder jede auftretende Variable im Programm explizit zu deklarieren.

Beispiel für implizite Deklaration:

DEFINT j,k,n	Variablen mit j,k,n sind vom Typ integer
DEFLNG p,q,u	Variablen mit p,q,r sind vom Typ langinteger
DEFSNG x-z	Variablen mit x,y,z sind vom Typ single precision
DEFDBL s,t	Variablen mit s,t sind vom Typ double precision
DEFSTR a-h	Variablen mit a,b,...,h sind Stringvariablen

Die Bedeutung ist die gleiche wie in BASICA resp. GWBASIC. Beachten Sie, dass in obigem Fall der Variablen a nur ein String und keine Zahl zugeordnet werden kann!

Beispiel für explizite Deklaration:

a% = 100	Integer-Variable
a& = 123456789	Langinteger-Variable
a! = 3.14159	Real-Variable mit single precision
a# = 2344.56778245	Real-Variable mit double precision
a$ = «Turbo BASIC»	String-Variable

Obwohl hier jedesmal der gleiche Name a verwendet wird, speichert der Computer wegen der expliziten Deklaration die Daten an fünf verschiedenen Speicherplätzen. Es ist also a% von a& usw. verschieden!

Die explizite Deklaration hat Vorrang vor der impliziten. Es ist deshalb gut, wenn Sie trotz impliziter Darstellung alle Stringvariablen mit einem nachfolgenden $-Zeichen explizit festlegen. Das sind Sie auch vom interpretativen BASIC her gewöhnt.

Ein Variablenname muss mit einem Buchstaben beginnen, auf den eine beliebige Anzahl Buchstaben und Ziffern folgen darf. Er darf theoretisch 255 Zeichen lang sein, wobei Turbo BASIC alle Zeichen als signifikant behandelt. Zwischen Gross- und Kleinbuchstaben wird nicht unterschieden. Die Umlaute ä,ö,ü sowie Leerstellen im Namen sind verboten. Da die Länge eines Variablennamens keinen Einfluss auf die Rechengeschwindigkeit hat, sollten Sie bei der Vergabe von Namen grosszügig sein. Der Name LAUFZEIT_ DES_ KAPITALS belegt genau so 4 Byte wie der Name N!

Auch in Turbo BASIC muss ein Array (Feld) Elemente vom gleichen Datentyp umfassen. Für **jedes Array** stehen maximal 64 KByte Speicherplatz unabhängig vom Programmcode zur Verfügung. Arrays können eindimensional und mehrdimensional sein, wobei bis zu acht Dimensionen möglich sind. Die maximale Anzahl Elemente pro Array beträgt

32768 bei Integerzahlen
16384 bei Langintegerzahlen
16384 bei Realzahlen mit single precision
8192 bei Realzahlen mit double precision
16384 bei Strings.

Arrays werden wie schon im interpretativen BASIC mit dem DIM-Befehl dimensioniert. Wird im DIM-Befehl der obere Index als Zahl eingeben, dann heisst das Array statisch. Der für das Array benötigte Speicherplatz bleibt während des Programmlaufs reserviert und kann nicht für andere Zwecke frei gegeben werden.

Beispiele:

DIM A%(6)	Integer-Array mit den 7 Elementen A%(0) bis A%(6)
DIM B&(20)	Langinteger-Array mit den 21 Elementen B&(0) bis B&(20)
DIM X!(100)	Array mit Realzahlen vom Typ single precision mit den 101 Elementen X!(0) bis X!(100)
DIM S# (32)	Array mit Realzahlen vom Typ double precision mit den 33 Elementen S# (0) bis S# (32)
DIM T$(5,4,3)	Dreidimensionales String-Array mit den 6*5*4 = 120 Elementen T$(0,0,0) bis T$(5,4,3). Jedes Stringelement kann maximal 255 Zeichen aufnehmen.

In Turbo BASIC ist es möglich, die Untergrenze des Array-Indexes beliebig festzusetzen.

Beispiele:

DIM A%(1980:1990)	im Speicher werden nur die 11 Plätze mit den Namen A%(1980), A%(1981), ..., A%(1990) reserviert.
DIM F(50:80, 20:40)	für dieses zweidimensionale Feld werden im Speicher nur 31*21 = 651 Plätze mit den Namen F(50,20), F(50,21),.....,F(50,40), F(51,20), F(51,21);.....,F(51,40), usw. F(80,20), F(80,21),.....,F(80,40) für Realzahlen mit single precision reserviert.

In Turbo BASIC kann man trotz der Kompilation dynamische Arrays erzeugen. Das entsprechende Feld wird erst während des Programmlaufs reserviert (sofern noch genügend Platz vorhanden ist). Das Feld kann im gleichen Programmlauf mit dem Befehl ERASE wieder frei gegeben werden.

Beispiel:

```
input«Wie viele Zahlen .....»;n
dim x(n)                    'dynamisch reservierter Array
for j=1 to n
   read x(j)
   .....
next j
```

9. Arithmetische Berechnungen und Werzuweisungen

Hier können wir uns kurz fassen, da die mathematischen Operatoren, die Hierarchie der Operationen und der Aufbau arithmetischer Ausdrücke sowie deren Wertzuweisung zu Variablen gleich sind wie im BASICA und GWBASIC.

Hierarchie der Operationen

1. Funktionsaufruf zuerst berechnen

2. Rechenoperationen in dieser Reihenfolge ausführen:

   2.1. Potenzieren
   2.2. Negation (unäres Minus)
   2.3. Multiplikation / Division
   2.4. Integerdivision (c=a\b)
   2.5. Modulo-Operation (c=a mod b)
   2.6. Addition / Subtraktion

3. Vergleichsoperationen ausführen

   a=b, a<>b, a<b, a<=b, a>b, a>=b

4. Logische Operationen in dieser Reihenfolge ausführen:

4.1.		not a	Verneinung
4.2.	a	and b	Und
4.3.	a	or b	inklusives Oder
4.4.	a	xor b	exklusives Oder
4.5.	a	eqv b	Aequivalenz
4.6.	a	imp b	Implikation

Die Vergleichs- und logischen Operationen ergeben entweder den Wahrheitswert «true» oder «false».

«true» entspricht &HFFFF = &B1111111111111111 = -1 (dezimal)
«false» entspricht &H0000 = &B0000000000000000 = 0 (dezimal)

	not a	ergibt «true», wenn a den Wert «false» hat.
a	and b	ergibt «true», wenn beide Operanden den Wert «true» haben.
a	or b	ergibt «true», wenn mindestens einer der Operanden den Wert «true» hat.
a	xor b	ergibt «true», wenn a und b verschiedene Werte haben.
a	eqv b	ergibt «true», wenn a und b gleiche Werte haben.
a	imp b	ergibt nur dann «false», wenn a «true» und b «false» ist.

Die Vergleichs- und logischen Operationen werden vorallem bei den Anweisungen IF...THEN...ELSE und SELECT CASE n benützt. Wir kommen sofort darauf zu sprechen.

Mit den logischen Operatoren sind aber auch Bit-Manipulationen möglich. Diese benützt man gerne beim Zeichnen von bewegter Grafik.

Wir haben darüber in M+K 86-6 (Programmieren mit dem IBM-PC) und im Buch «Erste Schritte mit dem PC» Band 2 ausführlich berichtet.

10. Anweisungen für Verzweigungen (einfache und mehrfache Auswahl)

Die bescheidenen Kontrollstrukturen für die einseitige, zweiseitige und mehrseitige Auswahl, die das sog. «klassische BASIC» anbietet, sind mit Recht ein Stein des Anstosses für jeden rechtschaffenen Programmierer.

Denken wir an die frühen 70-Jahre zurück!

Als 1964 BASIC von Kemeny und Kurtz vorgestellt wurde, gab es als einzige Konstollstruktur für Verzweigungen die Anweisungen:

```
IF Bedingung THEN GOTO Zeilennummer
IF Bedingung THEN eine einzige Anweisung
```

Da man damals nicht mehrere Anweisungen hintereinander schreiben konnte, musste man mit zusätzlichen GOTO's ständig Anweisungen überspringen, um einseitige, zweiseitige und mehrfache Verzweigungen simulieren zu können. Die vielen GOTO's haben die Lesbarkeit von BASIC-Programmen so drastisch verschlechtert, dass die Informatiker abfällig von «Spaghetticode» sprachen.

Die BASIC-Dialekte der frühen 80-Jahre wie BASICA, GWBASIC, Commodore-BASIC usw. boten eine Erweiterung der IF-Anweisung in folgenden Formen an:

```
IF Bedingung THEN Zeilennummer
ELSE Zeilennummer

IF Bedingung THEN mehrere Anweisungen
ELSE mehrere Anweisungen
```

Da aber der Interpreter nur 255 Zeichen (Zeilennummer eingeschlossen) auf einmal lesen und interpretieren kann, ist man mit der Anzahl der Anweisungen nach dem THEN und ELSE eingeschränkt. Oft ist die letzte Anweisung in einer solchen Folge das berüchtigte GOTO.

Trotzdem kann man mit obiger IF...THEN...ELSE-Anweisung kleine Programme recht gut strukturieren. Wir haben das in früheren M+K-Beiträgen (Programmieren mit dem IBM-PC) gezeigt.

Turbo BASIC kennt ebenfalls obige sog. zeilenorientierte IF-Anweisung. Man kann sie dank des Unterstreiche-Zeichens optisch besser gliedern:

```
IF Bedingung_
   THEN Anweisung(en)_
   ELSE Anweisung(en)
```

Aber Turbo BASIC bietet wesentlich bessere Kontrollstrukturen an. Sie ähneln denen von Pascal, Modula und anderen modernen Hochsprachen. Mit ihnen wollen wir uns intensiver beschäftigen.

Die einseitige Auswahl

Sie hat folgende allgemeine Form:

```
IF Bedingung THEN
   Anweisung(en)
   .....
   Anweisung(en)
END IF
```

Beachten Sie bitte die strenge Grammatik der Anweisung. Nach dem THEN muss unbedingt eine neue Zeile für den Block von Anweisungen begonnen werden. Das END IF muss allein auf einer Zeile stehen. Die Zahl der Zeilen im Anweisungsblock ist unbeschränkt! Hier können also Schleifen oder weitere IF-Anweisungen stehen.

Die zweiseitige Auswahl

Sie hat die folgende allgemeine Form:

```
IF Bedingung THEN
   Anweisung(en)
   .....
   Anweisung(en)
ELSE
   Anweisung(en)
   .....
   Anweisung(en)
END IF
```

Auch hier gilt wieder eine strenge Grammatik. Nach dem THEN muss eine neue Zeile begonnen werden und ELSE sowie END IF müssen für sich auf einer speziellen Zeile stehen. Turbo BASIC kennt eben nicht die Möglichkeit von Pascal, Blöcke von Anweisungen durch die Wörter «BEGIN» und «END» einzuschliessen.

Beispiel: Berechnung des Tages nach der Zeller-Formel

```
cls
deflng a-z
dim tag$(6)
for j=0 to 6 : read tag$(j) : next j

input«Tageszahl (1-31).....»;tage
input«Monatszahl (1-12).....»;monat
input«Jahreszahl (1901-1999)...»;jahr

if monat<=2 then
        jahr=jahr-1
        monat=monat+13
else
        monat=monat+1
end if

n=int(365.25*jahr)+int(30.6*monat)+tage-621049
rest=n-7*int(n/7)
print
print«Das ist/war ein »;tag$(rest)
end

data Sonntag, Montag, Dienstag, Mittwoch, Donnerstag,
Freitag, Samstag
```

Die mehrseite Auswahl

Sie hat die allgemeine Form:

```
IF Bedingung 1 THEN
   Anweisung(en)
ELSEIF Bedingung 2 THEN
   Anweisung(en)
ELSEIF Bedingung 3 THEN
   Anweisung(en)
   .....

ELSEIF Bedingung n THEN
   Anweisung(en)
ELSE
   Anweisung(en) für den Restfall
END IF
```

Sie können beliebig viele Bedingungen mit ELSEIF testen. Wenn Sie sicher sind, dass alle Fälle geprüft werden, dann können Sie den ELSE-Teil weglassen. Falls aber keine der Bedingungen den Wert «true» ergibt, dann entsteht ein Laufzeitfehler und das Programm wird abgebrochen!

Beispiel: Das bekannte Zahlenrate-Spiel HiLo

```
cls
randomize(timer)
defint a-z
zahl = int(128*rnd)+1
antwort = «falsch»

do while antwort = «falsch»
        input «Wie heisst die Zahl
        (1-128).....»;eingabe
        if eingabe > zahl then
                print«zu gross»
        elseif eingabe < zahl then
                print«zu klein»
        else
                print «Spitze, Du hast die Zahl
                gefunden»
                antwort$=«richtig»
        end if
loop
end
```

Für die mehrfache Verzweigung bietet Turbo BASIC eine weitere Anweisung an. Sie heisst SELECT CASE n und ist ebenfals dem Pascal entlehnt. Mit dieser modernen Anweisung wird die frühere Anweisung

ON n GOTO Zeile n1, Zeile n2, ..., Zeile nn

überflüssig. Dazu kommt noch, dass der Ausdruck n bei SELECT CASE n numerisch oder ein String sein kann.

Die SELECT CASE n hat folgende ausführliche Form:

```
SELECT CASE Ausdruck
        CASE Prüfung 1
           Anweisung(en)
        CASE Prüfung 2
           Anweisung(en)
        CASE Prüfung 3
           Anweisung(en)
           .....

        CASE Prüfung n
           Anweisung(en)
        CASE ELSE
           Anweisung(en)
END SELECT
```

Wiederum ist die strenge Grammatik bezüglich der Zeilendarstellung zu beachten.

Ausdruck ist eine numerische oder eine String-Grösse und wird bei den folgenden Prüfungen von oben nach unten getestet. Sobald eine Prüfung den Wert «true» hat, werden die Anweisungen des betreffenden Blocks durchgeführt. Danach verzweigt das Programm zur ersten Anweisung nach dem END SELECT. Wenn keine der Prüfungen den Wert «true» ergibt, entsteht ein Laufzeitfehler und das Programm wird abgebrochen.

Welcher Art sind die Prüfungen? Turbo BASIC erlaubt hier eine kompakte Kurzschreibweise:

```
SELECT CASE n
        CASE k          bedeutet n=k?
        CASE <k         bedeutet n<k?
        CASE >k         bedeutet n>k?
        CASE k1,k2      bedeutet n=k1 oder
                        n=k2?
        CASE k1 to k2   bedeutet k1<=n<=k2?
```

Statt langer Theorie möge ein Beispiel den Sachverhalt klären.

Beispiel:

```
cls
defint a-z
g1=100 : g2=50 :g3=10

input«Bitte eine Zahl (1-100).....»;zahl
select case zahl
        case 1 to 19
                print«Kein Gewinn»
        case 81 to 100
                print«Kein Gewinn»
        case 33,44,55,66,77
        if zahl = 77 then
                print«Gewinn Fr.»;g1
        else
                print«Gewinn Fr.»;g2
        end if
        case 20 to 80
                print«Gewinn Fr.»;g3
        case else
                print«falsche Eingabe»
end select
end
```

Nachtrag zu den Verzweigungs-Anweisungen

1. Mit dem Befehl EXIT IF resp. IF Bedingung THEN EXIT IF kann man vorzeitig aus einem Anweisungsblock herausspringen. Das Programm führt danach die erste Anweisung nach END IF durch.

2. Mit dem Befehl EXIT SELECT resp. IF Bedingung THEN EXIT SELECT kann man vorzeitig aus einem Anweisungsblock bei einem CASE-Fall herausspringen. Das Programm fährt dann bei der ersten Anweisung nach END SELECT fort.

3. Sowohl in den Blöcken bei der IF- als auch bei der SELECT-Anweisung können weitere IF- und SELECT-Anweisungen stehen. Wenn man solche Verzweigungsanweisungen verschachtelt, sollte man aufpassen, dass die Lesbarkeit des Programms nicht darunter leidet. Selbst N. Wirth, der Erfinder von Pascal und Modula, hat in diesen Fällen ein klärendes GOTO Sprungmarke empfohlen.

4. Mit den blockstrukturierten Verzweigungsanweisungen ist theoretisch das GOTO überflüssig geworden. Trotzdem kann es nach wie vor zur Vereinfachung der Konstrollstruktur gebraucht werden. Schreiben Sie dann aber bitte GOTO MARKE, wobei MARKE ein beliebiger Name für das Sprungziel ist. Dieses muss in der Form MARKE: dann allein auf einer Zeile stehen. Danach folgen jene Anweisungen, zu denen Sie mit GOTO MARKE springen wollen. Ein Beispiel haben wir Ihnen in Listing 1B (Tabelle der Primzahlen) im 1. Kapitel (M+K 88-2) dieser Einführung gegeben.

11. Anweisungen für Schleifen

Im «klassischen BASIC» gibt es für jede Form einer Schleife nur die bekannte FOR...NEXT-Anweisung. Sie hat die allgemeine Form:

```
FOR J = Anfangswert TO Endwert STEP Schrittweite
   Anweisung(en)
   .....
   [ IF Bedingung THEN GOTO Zeilennummer ]
   .....
   Anweisung(en)
NEXT J
```

Im Gegensatz zu Pascal kann die Schleifenvariable J (resp. ein anderer Name) nicht nur Integerzahlen sondern auch Realzahlen mit einfacher Genauigkeit durchlaufen. Ist die Schrittweite negativ, läuft die Schleife rückwärts.

Fehlt die IF-Anweisung, dann haben wir eine reine Zählerschleife. Ist die IF-Anweisung am Anfang resp. am Schluss des Anweisungsblockes, dann können wir die WHILE- resp. REPEAT-UNTIL-Schleife von Pascal simulieren.

In BASICA und GWBASIC steht noch das folgende Schleifenkonstrukt zur Verfügung:

```
WHILE Bedingung
    Anweisung(en)
    .....
    Anweisung(en)
WEND
```

Wenn die Bedingung nach WHILE den Wert «true» hat, werden die Anweisungen zwischen WHILE Bedingung und WEND durchlaufen. Ist die Bedingung aber «false», dann verzweigt das Programm zur 1. Zeile nach dem WEND und fährt mit den dort stehenden Anweisungen fort.

Turbo BASIC bietet insgesamt fünf verschiedene Schleifenkonstrukte an:

1. DO WHILE Bedingung Anweisung(en) LOOP	abweisende Schleife, wird eventl. nicht durchlaufen!
2. WHILE Bedingung Anweisung(en) WEND	abweisende Schleife, wird eventl. nicht durchlaufen!
3. DO Anweisung(en) LOOP UNTIL Bedingung	nicht abweisende Schleife, wird mindestens einmal durchlaufen!
4. DO Anweisung(en) IF Bedingung THEN EXIT LOOP Anweisung(en) LOOP	Schleife mit Abfrage für Ausstieg im Innern des Schleifenkörpers

```
5. FOR J=A TO B STEP C        Zählerschleife, eventl. mit
      Anweisung(en)           vorzeitigem Austieg
      [ IF Bedingung THEN EXIT FOR ]
      Anweisung(en)
   NEXT J
```

Die Wirkungsweise der fünf Schleifenkonstrukte ist wohl jedem Leser klar. Wir wollen aber mit drei kurzen Programmen die drei neuen DO-Schleifen vorstellen. Sie werden in Turbo BASIC-Programmen den eher veralteten FOR...NEXT-Schleifen vorgezogen.

1. Programm

```
cls
input«Geben Sie die Uhrzeit in der Form hh:mm:ss
ein...»;zeit$
time$=zeit$
locate 20,1:print«Tastendruck beendet das Einblenden
der Uhrzeit»

do while not instat
locate 12,15 : print«Die Uhrzeit ist »;time$
loop

cls
end
```

Solange Sie keine Taste antippen, wird fortlaufend am Bildschirm die Uhrzeit in digitaler Weise angezeigt, weil der logische Wert von NOT INSTAT (Taste nicht gedrückt?) den Wert «true» hat. Sobald Sie aber irgendeine Taste drücken, wird die DO-Schleife übersprungen und mit CLS und END das Programm abgebrochen.

2. Programm

```
cls
defdbl a-z
print«Die Berechnung von n!»
print:print
input«Geben Sie n ganzzahlig ein (1 < n < 171).....»;n
if n<>int(n) or n<2 or n>170 then
   print«Falsche Eingabe»
   stop
end if
fakultaet=1

do
   fakultaet = fakultaet*n
   decr n
loop until n<2

print«Die Fakultaet ist»;fakultaet
end
```

Um möglichst viele Stellen der Fakultät zu erhalten, benützen wir doppelt genaue Realzahlen. Mit ihnen kann man maximal 170! berechnen. Die Anweisung DECR N (genau eigentlich DECR N,1) ist eine in Turbo BASIC bequeme Kurzschreibweise für N = N-1. DECR N,K bedeutet somit N = N-K.

3. Beispiel

```
cls

do
   locate 12,10 : print«Willkommen bei TURBO BASIC»
   locate 20,1 : print«Bitte eine Taste drücken»
   taste$ = inkey$
   if taste$<>«» then exit loop
loop

cls
..... 'Fortsetzung des Programms
```

Dies ist eine andere Möglichkeit für einen sog. Bildstillstand. Wir benützen nicht die Anweisung DO WHILE NOT INSTAT sondern die Bedingung IF TASTE$<>«» THEN EXIT LOOP, um die DO-Schleife zu verlassen.

Sorgen Sie unbedingt dafür, dass solche «Endlos-Schleifen» auch wirklich verlassen werden können. In compilierten Programmen können Sie nicht mit der Break-Taste das Programm abbrechen!

12. Demo-Programme zum 2. Kapitel

Wir beenden dieses Kapitel mit einigen hübschen Demonstrationsprogrammen. Nochmals sollen die verschiedenen Schleifenkonstrukte inklusive dem vorzeitigen Ausstieg mit EXIT und die neuen Möglichkeiten für einfache und mehrfache Verzweigungen zum Zug kommen.

Es geht hier nicht darum, besonders ausgeklügelte Programme vorzustellen. Viele dieser Programme sind in anderen Programmiersprachen in diversen Büchern und Zeitschriften publiziert worden. Wir wollen die Turbo BASIC-Versionen dieser Programme bringen.

1. Programm: Nochmals die Liste der Primzahlen von m bis n

Listing 1B aus Kapitel 1 zeigt ein Programm, welches die Primzahlen von 10'000'000 bis 10'001'000 in 59,32 sec berechnet und auf dem Bildschirm hinschreibt.

Wenn man den Algorithmus verbessert, kann man die Zeit auf 16,81 sec verkürzen. Das ist eine rund dreifache Beschleunigung des Programms. Jetzt wird es sogar mög-

Listing 1

```
cls
print"Primzahlen von m bis n"
print"----------------------"

'Variablendeklaration
defdbl m,n
defint w,k,p,t,j

'Eingaben und Anfangswerte
print:print
input"Gib m ein ..............";m
input"Gib n ein ..............";n
m=6*int(m/6)+1
wurzel=int(sqr(n))+1:anzahl=int(wurzel/2)
dim p(anzahl)
p(1)=3 : p(2)=5 : p(3)=7 : p(4)=11 : k=4
print : print
t1!=timer
p=13 : s=4

'Vorlauf
do
    prim$="true"
    for t=3 to sqr(p)+1 step 2
        if p=t*int(p/t) then prim$="false" : exit for
    next t
    if prim$="true" then k=k+1 : p(k)=p
    p=p+s : s=6-s
loop until p>wurzel

'Hauptlauf
s=4
do
   primzahl$="ja"
   for j=1 to k
        if m=p(j)*int(m/p(j)) then
           primzahl$="nein"
           exit for
        end if
   next j
   if primzahl$="ja" then print m;
   m=m+s : s=6-s
loop until m>n
t2!=timer

print:print:print using"Rechenzeit in sec = ##.##";t2!-t1!
end
```

lich, mit einem BASIC-Programm Primzahlen oberhalb 1 Milliarde zu berechnen, ohne dass Sie lange am Bildschirm warten müssen. Das neue Programm enthält auch kein GOTO mehr.

Algorithmus

In einem Vorlauf berechnen wir die Primzahlen p(1)=3, p(2)=5, p(3)=7, p(4)=11 , p(5)=13, ..., p(j)=p. Dabei ist p jene grösste Primzahl, für die p*p gerade noch <= n ist. n ist dabei die obere Grenze des Bereiches, in dem wir Primzahlen suchen.

Im Hauptlauf prüfen wir alle ungeraden Zahlen von der Form 6*k+1 und 6*k+5, welche zwischen m und n liegen. Da 6*k+3 offensichtlich durch 3 teilbar ist, reduzieren wir unsere Primzahlkandidaten um einen Drittel.

Sei m = 6*k+1 die erste solche Zahl. Dann müssen wir m abwechselnd um 4, um 2, um 4, um 2, ... erhöhen, um die nächste Zahl zu erhalten, welche als Primzahl in Frage kommen kann. Wir erreichen dies mit den Anweisungen:

```
s = 4
.....
m=m+s : s=6-s
```

Bei jedem Durchlauf wechselt dann s von 4 auf 2, auf 4, auf 2,usw. Um zu prüfen, ob m eine Primzahl ist, teilen wir m der Reihe nach durch die gespeicherten Primzahlen p(1), p(2), ..., p(j)=p.

2. Programm: Numerische Integration nach Simpson

In der Nummer 87-6 der Zeitschrift Backup hat Werner Durandi in seinem Beitrag «Schnelle Real-Arithmetik mit Turbo Pascal» ein Pascal-Programm zur numerischen Integration einer Funktion mit Hilfe der sog. Trapezregel vorgestellt.

Um die Rechengeschwindigkeit in Turbo Pascal zu messen, wurde die Funktion y = (x + 1)/sqr(2*x*x*x*x*x + 1) von a=0 bis b=1 integriert, wobei die Schrittweite dx= (b-a)/10'000 gewählt wurde.

Uns interessieren die Zeiten, da wir sie mit den Laufzeiten eines ähnlichen Turbo BASIC-Programms vergleichen wollen.

IBM-PC-XT	(Pascal 3.01A)	ohne Coprozessor	184,7 sec
IBM-PC-XT	(Pascal 3.01A)	mit Coprozesor	20,3 sec
IBM-AT-03	(Pascal 3.01A)	ohne Coprozessor	55,9 sec
IBM-AT-03	(Pascal 3.01A)	mit Coprozessor	14,3 sec

Listing 2 zeigt ein Programm in Turbo BASIC, welches eine beliebige stetige Funktion y=f(x) im Abschnitt a <=x <= b mit Hilfe der sogenannten Simpson-Regel näherungsweise integriert.

Algorithmus

Teile das Intervall (a,b) in n Teile, wobei n eine gerade Zahl sein muss. h=(b-a)/n ist somit die Schrittweite.

Berechne die Funktionswerte an den Stellen
y(0)=f(a), y(1)=f(a+h),
y(2)=f(a+2*h), y(3)=f(a+3*h),.....,y(n-1)=f(a+(n-1)*h),
y(n)=f(b).

Dann gilt:

Das Integral ist angenähert h/3 * (S0 + 2*S1 + 4*S2), wobei

```
S0 = y(a) + y(b)
S1 = y(2) + y(4) + ..... + y(n-2)
S2 = y(1) + y(3) + ..... + y(n-1)
```

Das Turbo BASIC-Programm braucht für die numerische Integration der Testfunktion y = (x + 1)/ sqr(2*x*x*x*x*x + 1) von a=0 bis b=1 und n=10'000 auf dem Commodore PC 10-II und eingesetztem Coprozessor 8087 die Traumzeit von 9,23 sec. Dabei haben wir zur Erhöhung der Rechengenauigkeit erst noch mit doppeltgenauen Realzahlen gerechnet.

Es ist fast nicht zu glauben. Turbo BASIC auf einem 4,77 MHz getakteten PC schlägt Turbo Pascal (Version 3.x) auf dem IBM-AT-03!

Listing 2

```
cls
print"Numerische Integration nach Simpson"
print"-----------------------------------"
print:print:print

'Variablendeklaration
defdbl a,b,d,s
defint j,n
def fnf(x)=4/(1+x*x)

'Eingaben und Anfangswerte
input"Untere Grenze ......................";a
input"Obere  Grenze ......................";b
input"Anzahl Intervalle (n gerade).........";n
t1=timer
dx=(b-a)/n
s0=fnf(a)+fnf(b)  :s1=0 : s2=0

for j=2 to n-2 step 2
    s1=s1+fnf(a+j*dx)
next j

for j=1 to n-1 step 2
    s2=s2+fnf(a+j*dx)
next j

s=(s0+2*s1+4*s2)*dx/3

t2=timer
print:print
print"Integral ..........................";s
print:print
print using"Rechenzeit in sec = ###.##";t2-t1
end
```

Wenn Sie die Funktion y = 4/(1+x*x) von a=0 bis b=1 mit dem Simpson-Programm numerisch integrieren, erhalten Sie eine Näherung für die Zahl pi. In der Mathematik wird gezeigt, dass die Stammfunktion von f(x) = 4/(1+x*x) die Funktion F(x) = 4*arctan(x) ist. Folglich ist das bestimmte Integral über die Funktion f(x) von a=0 bis b=1 gleich 4*arctan(1) = 4*pi/4 = pi.

Mit z.B. n=5000 erhalten Sie in 2,64 sec den Wert **3.141592653**274536, wobei die fetten Ziffern richtig sind.

3. Programm: Spektrum verschiedener Funktionen z=f(x,y)

Wir haben in einem früheren Beitrag im COMPUTER-MARKT (Programmieren mit dem IBM-PC) und im Buch «Erste Schritte mit dem PC» Band 1 ausführlich erklärt, wie man von einer dreidimensionalen Funktion z=f(x,y) das sog. Funktionsspektrum erstellt. Als Beispiel wurde dort die Funktion z=exp(-x*x-y*y)/125 verwendet.

Die Grundidee ist simpel: In einer Doppelschleife wird y von -12 bis +12 und x von -20 bis +20 variiert und jeweils der zugehörige z-Wert berechnet. Der ganzzahlige Teil von z modulo 16 bestimmt eine der 16 möglichen Zeichenfarben, in der an der Stelle P(Zeile/Spalte) ein farbiges ausgefülltes Quadrat als Pixel hingezeichnet wird.

Auf diese Weise entstehen in niederer Auflösung wunderschöne Farbmuster. Das Programm in Listing 3 gibt Ihnen eine Auswahl von 11 willkürlichen Funktionen. Sie können aber beliebig weitere Funktionen erfinden und die zugehörigen Spektren zeichnen lassen. Sicher werden Sie an den hübschen Mustern viel Freude haben.

Listing 3

```
cls:width 80
print"Spektrum von 11 Funktionen z = f(x,y)"
print"-----------------------------------"
print:print
defint k,j,n,f
defsng x,y,z

print"Was wollen Sie?" : print
print tab(10);" 1 ..... z=sin(8*x)+exp(y)"
print tab(10);" 2 ..... z=(exp(x)+exp(-x))/2+log(y+sqr(1+y*y))"
print tab(10);" 3 ..... z=3*(atn(x)+atn(y))"
print tab(10);" 4 ..... z=(x*x+y*y*y)/30"
print tab(10);" 5 ..... z=3*(x*x*x-y*y)*sin((x+y)/20)/(x*x+y*y+.3)"
print tab(10);" 6 ..... z=sin(x-y)+sqr(abs(x*y))"
print tab(10);" 7 ..... z=sqr(x*x+y*y)"
print tab(10);" 8 ..... z=exp(sqr(((x+9)/3^2/((y/5)^2+.03)))"
print tab(10);" 9 ..... z=16*atn(exp(x/9))-atn(exp(y))+y"
print tab(10);"10 ..... z=1600/exp(1/(1+abs(x)+abs(y)))/33"
```

```
print tab(10);"11 ..... z=x*y/16"
print:print:print

input"Geben Sie eine Zahl ein (1-11).....";n

cls : width 40

for j=1 to 25
    y=j-13
    for k=1 to 39
        x=k-20
        select case n
          case 1
             z=sin(8*x)+exp(y)
          case 2
             z=(exp(x)+exp(-x))/2+log(y+sqr(1+y*y))
          case 3
             z=3*(atn(x)+atn(y))
          case 4
             z=(x*x+y*y*y)/30
          case 5
             z=3*(x*x*x-y*y)*sin((x+y)/20)/(x*x+y*y+.3)
          case 6
             z=sin(x-y)+sqr(abs(x+y))
          case 7
             z=sqr(x*x+y*y)
          case 8
             z=exp(sqr(((x+9)/3*(x+9)/3/((y/5)*(y/5))+.03)))
          case 9
             z=16*atn(exp(x/9))-atn(exp(y))+y
          case 10
             z=1600/exp(1/(1+abs(x)+abs(y)))/33
          case 11
             z=x*y/16
          case else
               width 80:print"falsche Eingabe":stop
        end select
        z=int(z)
        farbe=z-16*int(z/16) : color farbe
        locate j,k : print chr$(219);
    next k
next j

beep : end
```

*4. Programm: Mandelbrot-Mengen von z(n+1) = z(n)*z(n) + c*

Ueber die berühmten «Apfelmännchen», die mit dem in Listing 4 abgedruckten Programm erzeugt werden, ist in Zeitschriften und Büchern schon so viel geschrieben worden, dass wir hier darauf verzichten wollen. Wer sich für die mathematische Theorie interessiert, dem sei das wunderschöne Buch von H.-O. Peitgen und P.H. Richter «The Beauty of Fractals», Springer-Verlag empfohlen.

Das Programm benützt den von Peitgen angegebenen Algorithmus. Wenn Sie auf Ihrem PC ebenfalls die in den

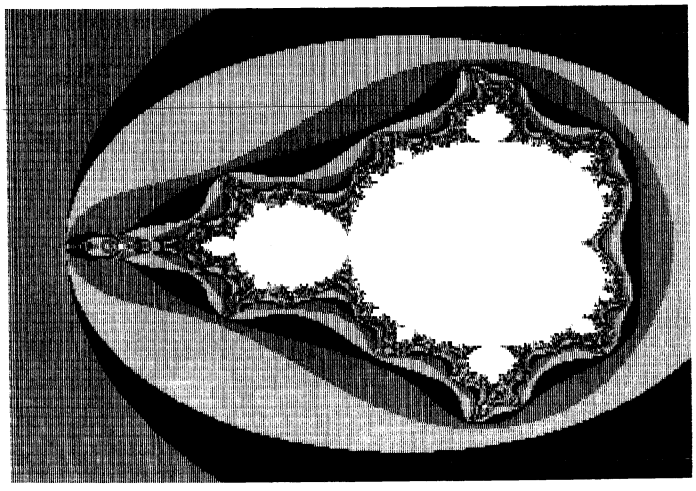


Abbildung 1

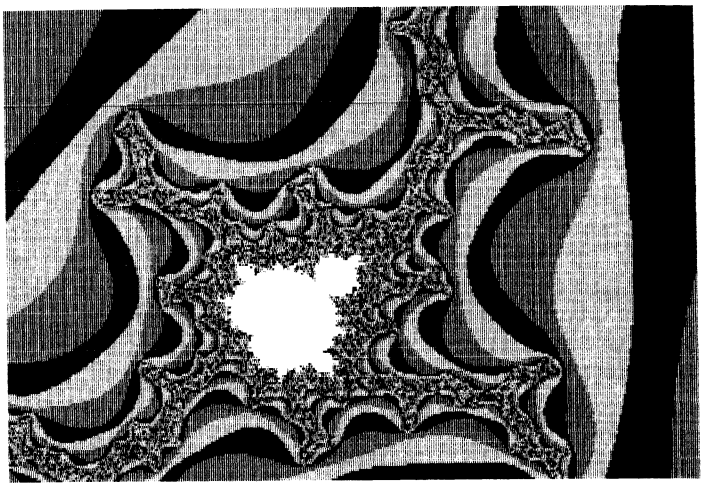


Abbildung 2

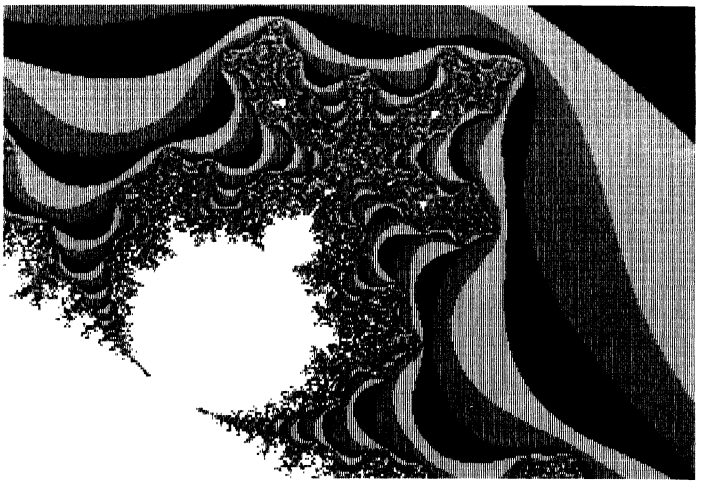


Abbildung 3

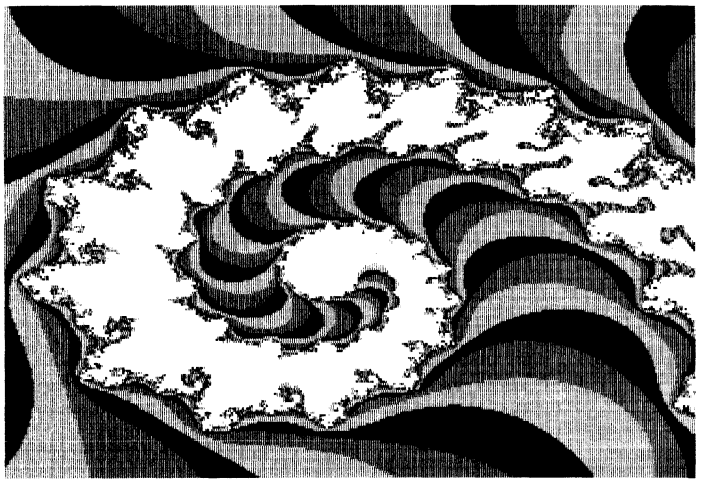


Abbildung 4

Abbildungen 1-7 gezeigten Figuren erstellen möchten, dann benützen Sie als Eingabewerte die Zahlen aus der folgenden Tabelle:

Abb.	R von cmin	I von cmin	R von cmax	I von cmax
1	-2.0	-1.25	0.5	1.25
2	-0.19920	1.01480	-0.12954	1.06707
3	-0.713	0.49216	-0.4082	0.71429
4	-0.74591	0.11196	-0.74448	0.11339
5	-1.254024	0.046252	-1.252861	0.047125
6	-0.745468	0.112979	-0.745385	0.113039
7	-0.7454356	0.1130037	-0.7454215	0.1130139

Dabei steht R für Realteil und I für Imaginärteil der komplexen Zahl c=(a+i*b). Bei allen Figuren wurde für die Maximalzahl der Iterationen die Zahl 100 eingegeben.

Abbildung 1 zeigt das berühmte Apfelmännchen. Der helle Bereich in der Mitte stellt die Mandelbrot-Menge dar, also jenen Bereich der komplexen c-Werte, für die z(n+1)=z(n)*z(n)+c für n gegen unendlich konvergiert. Die Abbildungen 2-7 sind Ausschnittsvergrösserungen vom fraktalen Rand der Mandelbrotmenge.

Der Autor hat seine ersten Apfelmännchen auf dem C-64 erstellt. Das war allerdings eine zeitraubende Angelegenheit. Nächtelang musste der Rechner eingeschaltet bleiben, da viele dieser Figuren erst nach sieben- bis zwölfstündigen Berechnungen fertig erstellt waren. Turbo BASIC mit dem Coprozessor schafft die Abbildung 1 in 24 Minuten und 21,35 Sekunden!

Inzwischen sind verschiedene mathematische Tricks ersonnen worden, um das Programm noch mehr zu beschleunigen. Es existieren auf dem Markt Programme, mit denen Apfelmännchen in 3-4 Minuten gezeichnet werden.

Wenn Sie das Programm genauer studieren, sehen Sie, dass wir eine «Abbruchmöglichkeit» eingebaut haben. Sobald Sie während des Zeichnens irgendeine Taste an-

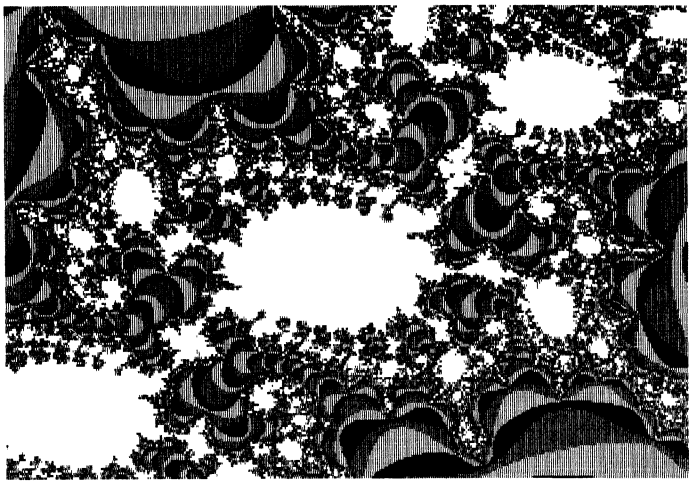


Abbildung 5

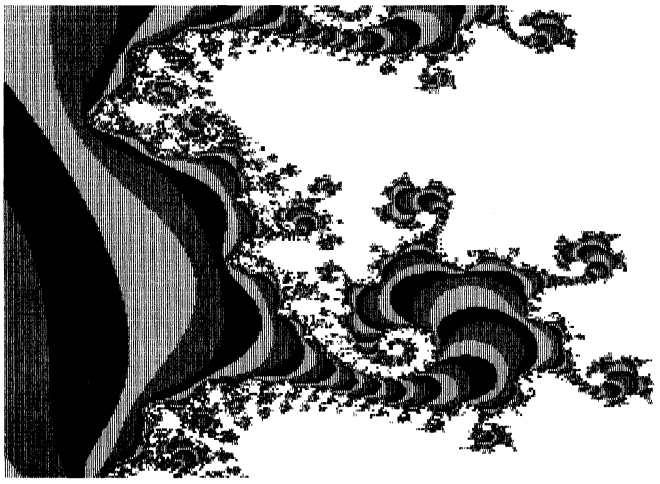


Abbildung 6

Abbildung 7

Listing 4

```
cls
print"Die Mandelbrot-Menge von z(n+1) = z(n)*z(n) + c"
print"-------------------------------------------"
print:print
defint m,n,s,z,f
defsng r,i,d,x,y,a,b

input"Realteil von c(min).....................";real.c.min
input"Imaginärteil von c(min).................";imag.c.min
print:print
input"Realteil von c(max).....................";real.c.max
input"Imaginärteil von c(max).................";imag.c.max
print:print
input"Maximalzahl der Iterationen.............";nmax

screen 1:color 0,0
dr=(real.c.max-real.c.min)/319
di=(imag.c.max-imag.c.min)/199

do
   for zeile=0 to 199
       b=imag.c.min+zeile*di
       for spalte=0 to 319
           a=real.c.min+spalte*dr
           r=0 : i=0 : d=0 : n=0
           do while d<4
              x=r : y=i
              r=x*x-y*y+a
              i=2*x*y+b
              d=r*r+i*i
              n=n+1
              if n=nmax then
                 exit loop
              end if
           loop
           if n<nmax then
              farbe=(n mod 3)+1
              pset(spalte,zeile),farbe
           end if
           if instat then exit loop
       next spalte
   next zeile
   beep 3 : stop
loop
end
```

tippen, wird die Berechnung abgebrochen und das Programm gestoppt. Sie brauchen also nicht 20-30 Minuten zu warten, falls Sie für ungeschickte Anfangswerte keine hübsche Figur erhalten. Die Abbruch-Kontrolle wird mit der Anweisung IF INSTAT THEN EXIT LOOP vorgenommen.

Im nächsten Beitrag werden wir uns mit dem Schreiben von Prozeduren und dem Einsatz der Rekursion in Turbo BASIC beschäftigen. Damit betreten wir echtes Neuland in BASIC, denn diese Domäne war bis jetzt nur Pascal, Modula und anderen modernen Hochsprachen vorbehalten. □

Auf der einen Seite begeistert der neue Laser-Drucker von Canon durch seine einmaligen Vorteile.

Auf der anderen Seite auch.

Canon LBP-8 II R
für doppelseitige Druckausgabe.
Die Weltneuheit vom Laser Printer-Hersteller Nr. 1.

Einmal mehr ist **Canon-Technologie** allen anderen einen Schritt voraus. Denn die Laser Qualität der Canon-Printer beeindruckt jetzt neuerdings auch auf den Rückseiten Ihrer Ausdrucke.

Aber nicht nur deshalb darf der Welt erster Duplex-Printer LBP-8 II R getrost als vielseitigster aller Drucker betrachtet werden. Sein doppeltes Kassettensystem erlaubt eine pausenlose Papierzufuhr von bis zu 400 Blatt A4 (=800 Vor- und Rückseiten). Ohne mühsames Wechseln der Kassetten wählen Sie per Tastendruck zwei verschiedene Formulare an. Zeit-, Papier- und Ablageplatzsparen gehören somit ebenfalls zum Repertoire des LBP-8 II R.

Zudem zeigt sich der Neue von Canon auch gegen aussen vielseitig. Als vollkompatibles Peripheriegerät, das sich mit allen Arten von Hard- und Softwaresystemen problemlos verbinden lässt. Kein Wunder, dass die unerreichte Canon-Technologie auch im Innern vieler weiterer Printer-Weltmarken die entscheidende Rolle spielt.

Der LBP-8 II R ist äusserst bedienungsfreundlich, nahezu wartungsfrei und verblüfft mit einem erstaunlichen Preis-/Leistungsverhältnis – für Canon-Kenner fast schon eine Selbstverständlichkeit.

Walter Rentsch bietet Ihnen neben dem LBP-8 II R eine Canon-Printer-Palette, die für jeden Bedarf die optimale Lösung bereithält. Spitzengeräte für den hochstehenden und effizienten Druck von Text und Grafik.

Walter Rentsch ist Ihr idealer Partner für Computer und Peripherie. Mit kompetenter Beratung und Betreuung vor, während und nach dem Kauf. Und einem ausgebauten Servicenetz, dank dem wir immer für Sie da sind.

Canon

Mehr Information von unserer Seite.

Gerne senden wir Ihnen gratis und unverbindlich unsere Unterlagen.

☐ Die Weltneuheit Canon LBP-8 II R interessiert mich. Bitte senden Sie mir Ihre ausführliche Dokumentation.

☐ Informieren Sie mich detailliert über die weiteren Spitzenprodukte aus der Canon-Printer-Palette.

Name:
Firma:
Strasse:
PLZ/Ort:
Telefon:

Einsenden an Walter Rentsch AG, Postfach, Industriestrasse 12, 8305 Dietlikon

Walter Rentsch
Zu Ihrer Information.

8305 Dietlikon, Postfach, Industriestrasse 12, Telefon 01/835 61 61
Aarau 064/25 44 22, Allschwil BL 061/38 31 16, Chur 081/22 79 86, Corcelles NE 038/31 53 69, Fribourg 037/24 24 76, Ittigen BE 031/58 81 81, Lausanne 021/33 31 41, Littau LU 041/57 02 33, Meyrin GE 022/82 08 00, Pregassona-Lugano 091/52 70 41, Sion 027/23 37 35, St. Gallen 071/27 77 27

Mathematik (2. Teil)

Wie wahrscheinlich ist die Wahrscheinlichkeit? Dieses oder ähnliches werden Sie sich wohl fragen, wenn Sie den ersten Beitrag zur Mathematik in M+K 87-6 gelesen haben. Denn obwohl Sie sich peinlich genau an die Anleitung gehalten haben, wurden Sie weder Lottomillionär, geschweige denn Sieger beim Würfelspiel. Für diese Misere bieten sich zwei Gründe an: entweder hat der Autor dieses Artikels keine Ahnung von Mathematik oder aber die Wahrscheinlichkeitsrechnung hat einige Tücken, auf die bisher noch nicht eingegangen wurde. Nehmen wir einmal an, die zweite Behauptung sei wahr und gehen der Sache auf den Grund.

Michael Schlingmann

Bleiben wir gleich beim Würfel und untersuchen eine Anzahl von Experimenten. Es ergebe sich das folgende Bild:

Augenzahl	Häufigkeit
1	387
2	356
3	377
4	391
5	402
6	398

Der Mittelwert dieser 2'311 Würfe beträgt ungefähr 3.54 und nicht 3.5, wie die Theorie es verlangt. Somit erhebt sich die Frage, ob die Abweichung vom Erwartungswert 3.5 rein zufällig ist, oder ob es sich um eine durch Unsymmetrie des Würfels verursachte signifikante Abweichung handelt. Oder anders formuliert: mit welcher Wahrscheinlichkeit erhält man einen Mittelwert von grösser oder gleich 3.54 bei 2'311 Würfen, wenn man annimmt, dass der Würfel in Ordnung ist?

Mit dieser und ähnlichen Fragen beschäftigt sich die *Beurteilende Statistik*, die in der Praxis wesentlich häufiger gebraucht wird als die Wahrscheinlichkeitsrechnung. Denn in der Wirtschaft kann man nicht von idealen Bedingungen ausgehen und die erhaltenen Daten in eine der früher ermittelten Wahrscheinlichkeitsverteilungen einsetzen. Vielmehr muss man das Experiment der Theorie annähern und sich anschliessend fragen, wie gross der Fehler war, den man dadurch begangen hat. Es ist leicht einzusehen, dass diese Fragestellungen wesentlich komplizierter sind als die Ermittlung des Erwartungswerts eines idealen Würfels.

Aus diesem Grund soll in das Thema der Beurteilenden Statistik nicht allzu tief eingedrungen werden. Diese Beschäftigung wollen wir den Wirtschaftswissenschaftlern überlassen. Trotzdem will ich hier einen kleinen Ueberblick geben.

Wie die Wahrscheinlichkeitsrechner haben auch die Statistiker eine eigene Nomenklatur. Damit diese beim Studium von Literatur etwas verständlicher wird, seien hier einige Definitionen vorangestellt: Eine *Grundgesamtheit* ist eine (unendliche) Anzahl von Versuchen unter gleichen Bedingungen.

Einen einzelnen Versuch bezeichnet man als *Element* der Grundgesamtheit.

In der Statistik kann man aus Zeitgründen immer nur eine endliche Teilmenge von Elementen aus der Grundgesamtheit betrachten. Man nennt dies eine *Stichprobe vom Umfang n*, wenn n die Anzahl der in der Stichprobe enthaltenen Elemente ist.

Da Rückschlüsse auf die Grundgesamtheit umso besser gezogen werden können, je grösser der Umfang der Stichprobe ist, muss dieser in den entsprechenden Formeln berücksichtigt werden. Dies steht im Gegensatz zur Wahrscheinlichkeitsrechnung, in der die Anzahl der Versuche keine Rolle spielt, da sowieso immer nur der Mittelwert betrachtet wird.

Für den Mittelwert und die Varianz einer Stichprobe gelten aber die schon früher verwendeten Formeln

$$\bar{x} = x(i)$$
$$s^2 = 1/(n-1)$$
$$(\bar{x} - x(i))^2$$

Bei kleineren Stichproben ist hierbei entscheidend wichtig, wie die *Klasseneinteilung* vorgenommen wurde. Die Klasseneinteilung ist bildhaft gesehen die Anzahl der Fächer, in die man verschiedene Versuchsergebnisse einordnen kann. Dazu ein Beispiel: Es werde die Anzahl der Autos pro Minute auf einer Hauptstrasse in der Züricher Innenstadt gezählt. Die Stichprobe habe den Umfang n=80.

Ohne Klasseneinteilung

Anzahl der Autos	Häufigkeit
33	1
34	0
35	0
36	2
37	1
38	7
39	6
40	7
41	11
42	14
43	12
44	4
45	13
46	0
47	2

Mit Klasseneinteilung

Anzahl der Autos	Häufigkeit
33-34	1
35-36	2
37-38	8
39-40	13
41-42	25
43-44	16
45-46	13
47-48	2

Werden beide Häufigkeitsverteilungen grafisch aufgetragen, so wird man im zweiten Fall erkennen können, dass es sich um die glockenförmige Gaussverteilung handeln könnte. Im ersten Fall (ohne Klasseneinteilung) ist dies schwieriger.

Wenn man nun die Mittelwerte und die Varianzen bildet (bei der Klasseneinteilung ist für die Messergebnisse x(i) der arithmetische Mittelwert der jeweiligen Klassengrenzen einzusetzen), so erhält man durchaus verschiedene Ergebnisse:

Ohne Klasseneinteilung
$\bar{x}=41.65$ $s^2 = 7.19$

Mit Klasseneinteilung
$\bar{x}=42.20$ $s^2 = 8.36$

Unterschiedliche Rechenmethoden bedingen also in der Regel auch verschiedene Ergebnisse, was den Meinungsforschern wohl recht gelegen kommt. Die obigen Ergebnisse gleichen sich natürlich an, wenn man den Umfang der Stichprobe gegen Unendlich gehen lässt.

Die Fragestellung in der Statistik läuft in der Regel darauf hinaus, festzustellen, mit welcher Wahrscheinlichkeit man sich bei einer Aussage irren kann, die man aufgrund der angefallenen Daten getätigt hat. Man

berechnet also eine Irrtumswahrscheinlichkeit.

Um zu prüfen, ob die Abweichung eines Ergebnisses zufälliger oder systematischer Natur ist, setzt man die sogenannte *Nullhypothese* an. Diese besteht darin, dass man sagt, es handele sich um eine zufällige Abweichung der Messwerte. Bei Annahme der Nullhypothese kann man zwei Fehler begehen: Ein Fehler 1. Art tritt auf, wenn die Nullhypothese nach der Prüfung verworfen wird, obwohl sie richtig ist.

Einen Fehler 2. Art macht man, wenn man die Nullhypothese akzeptiert, obwohl sie falsch ist.

Dazu ein Beispiel: Wird bei einem Vergleich festgestellt, dass ein neues Medikament besser ist als das alte (obwohl das nicht der Fall ist), so handelt es sich um einen Fehler 1. Art. Stellt sich bei dem Vergleich hingegen heraus, dass beide Medikamente gleichwertig sind (obwohl das neue besser ist), so wurde ein Fehler 2. Art begangen.

Zum Prüfen der Nullhypothese verwendet man entweder die Gauss-'sche Normalverteilung oder aber solche Verteilungen, die eine normalverteilte Grundgesamtheit voraussetzen, der Realität aber etwas näherkommen als Gauss.

Die Normalverteilung wendet man in der Regel dann an, wenn von der Grundgesamtheit der Mittelwert und die Varianz hinreichend genau bekannt sind. Die Irrtumswahrscheinlichkeit α erhält eine anschauliche Bedeutung, wenn man das Bild 1 betrachtet. Sie ist die für bestimmtes x1 und x2 unter der Glockenkurve liegende Restfläche. x1 und x2 seien dabei vorgegebene Abweichungen vom Mittelwert. Ein α von 0.05 (also 5% Irrtumswahrscheinlichkeit) bedeutet, das 5% der Fläche der Glockenkurve als Restfläche gelten. Alles, was zwischen den beiden kleinen Flächen ist, erfüllt die Nullhypothese.

vWenn x wieder der Mittelwert und $s = \sqrt{s^2}$ die Standardabweichung sind, so ergibt sich, wenn man die Gaussformel anwendet eine statistische Sicherheit von 68.3% für die Annahme, dass x im Intervall x - s bis x + s liegt. Dementsprechend erhält man eine statistische Sicherheit von 95.4% (99.7%) für x + 2s (x + 3s). Für das obige Beispiel mit einer Irrtumswahrscheinlichkeit von 5% muss man also etwa das Intervall x + 2s zulassen. Der Mittelwert liegt dann mit 95.7% Wahrscheinlichkeit innerhalb dieses Intervalls, um es noch einmal zu sagen.

Eine Möglichkeit zur Berechnung der Gaussverteilung wurde schon in der ersten Folge dieser Reihe angegeben (M+K 87-6). Natürlich gilt sie auch im Bereich der Beurteilenden Statistik.

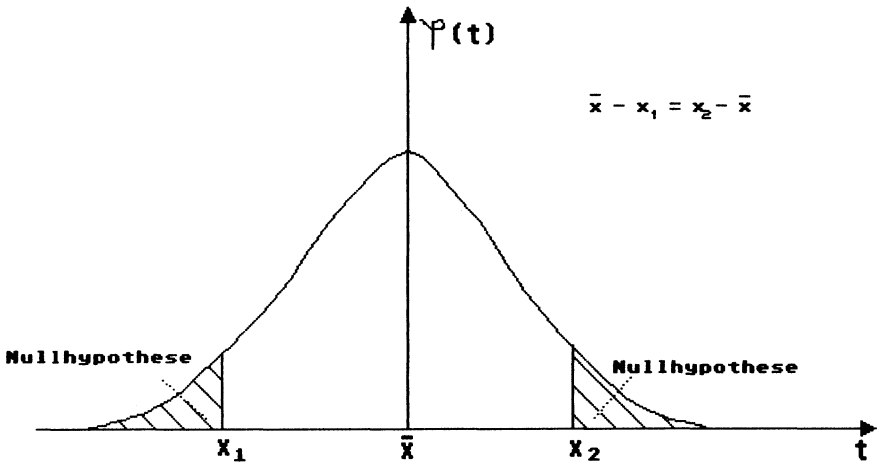


Bild 1: Gaussverteilung und Nullhypothese

Anstatt der Normalverteilung wird häufig auch die Student'sche t-Verteilung verwendet. Bei dieser Formel wird der Umfang n der Stichprobe berücksichtigt. Für n gegen Unendlich geht die Studentverteilung in die Normalverteilung über. Auch diese Verteilung ist in Nachschlagewerken tabelliert. Stattdessen kann man sie natürlich wie die Gaussverteilung näherungsweise iterieren. Hierauf wollen wir aber nicht eingehen. Zur Form der t-Verteilung siehe Bild 1.

Natürlich gibt es noch andere statistische Verteilungen, die den speziellen Problemen besser angepasst sind. Zum Beispiel wäre es nicht sinnvoll, zur Prüfung einer poissonverteilten Grundgesamtheit eine Gaussverteilung einzusetzen. Die verschiedenen Möglichkeiten werden in fast jedem Buch über Beurteilende Statistik erläutert.

Fehlerrechnung

Spätestens jetzt müssten Sie sich die Frage stellen, wie man eigentlich eine Messung anstellt. Denn was hilft die beste statistische Methode, wenn sich schon beim Zusammentragen des zu beurteilenden Materials Fehler eingeschlichen haben?

Als Beispiel diene eine Radarfalle der Polizei. Hierbei wird mittels eines Radarstrahls die Geschwindigkeit eines vorbeifahrenden Autos ermittelt. Der Fahrer wird natürlich behaupten, dass er sich an die Geschwindigkeitsbegrenzung gehalten hat und dass lediglich der Tachometer seines Autos falsche Tempi anzeigt. Um sich gegen solche Behauptungen von vornherein verwahren zu können, zählen die Gesetzteshüter zur erlaubten Höchstgeschwinigkeit noch 8% hinzu, so dass man statt 50 km/h auch 54 km/h fahren könnte, ohne bestraft zu werden. Vorausgesetzt, der Tacho funktioniert einwandfrei. Mit diesem Beispiel begegnet uns eine sehr einfache Form der Fehlerrechnung. Man sagt, die Messung sei nicht richtig und versucht sie zu korrigieren, indem man zu jedem Messwert einen gleichen Prozentsatz hinzuaddiert.

Aehnlichen Problemen sehen sich die Wissenschaftler tagtäglich gegenüber. Bevor sie aufwendige statistische Analysen über ein Thema anfertigen können, müssen sie zuerst die Messwerte so aufbereiten, dass sie möglichst wenig fehlerbehaftet sind oder aber dass der Fehler, denn man im ungünstigsten Fall begeht, bekannt ist.

Wie man sich leicht klarmachen kann, gibt es systematische und zufällige Fehler: systematische Fehler entstehen z.B. dann, wenn die Messapparatur nicht in Ordnung ist (mit einem Meterstab, der in Wirklichkeit nur 90 cm lang ist, wird man in der Regel keine allzu genauen Ergebnisse erzielen). Man versucht die systematischen Fehler so klein zu machen, dass ihr Einfluss auf das Messergebnis vernachlässigbar ist.

Sind die systematischen Fehler weitestgehend unterdrückt, dann bleiben noch die zufälligen. Diese Fehler sind nicht reproduzierbar und können nur mit mathematischen Methoden berücksichtigt werden.

In den meisten Fällen werden die zufälligen Fehler einer Gaussverteilung gehorchen. Die Gausskurve wird dabei symmetrisch um den wahrscheinlichsten Messwert eingezeichnet. Die Wahrscheinlichkeiten für eine bestimmte Grösse der Fehlers können direkt aus der Kurve abgelesen werden. Wesentlich dabei ist, dass die falsche Messwerte mit glei-

cher Wahrscheinlichkeit grösser oder kleiner als der «Mittelwert» liegen. Ist dies nicht der Fall, so liegt zusätzlich ein systematischer Fehler vor, den man in der Regel leicht berücksichtigen kann.

Die Frage ist, wie man den wahrscheinlichsten Messwert x aus den Messungen x(i) berechnen kann. Es hat sich gezeigt, dass es am zweckmässigsten ist, diesen Wert durch die Bedingung

$$\Sigma (x(i) - \bar{x})^2 = \text{Minimum}$$

darzustellen. Berechnet man dieses Minimum mit Hilfe der Differentialrechnung (auf sie wird in einem der folgenden Beiträge eingegangen), so erhält man das Ergebnis

$$\bar{x} = 1/n \; \Sigma x(i)$$

Der wahrscheinlichste Wert ist also das arithmetische Mittel, ein Ergebnis, das wir schon oft benutzt haben.

Der mittlere Fehler m einer Einzelmessung ist definiert durch

$$m = \sqrt{\Sigma[(x-x(i))^2/(n-1)]}$$

Je mehr Messwerte man erhält, um so genauer lässt sich der «wahrscheinlichste» Messwert, nämlich das arithmetische Mittel, bestimmen. Sei n die Anzahl der Messungen, so ist

$$\sqrt{\Sigma[(\bar{x}\,1-x(i))^2/(n(n-1))]} = m/\sqrt{n}$$

der mittlere Fehler des arithmetischen Mittels. Je mehr Messungen also gemacht worden sind, umso kleiner wird der mittlere Fehler.

Die verschiedenen Fehler kann man mit Hilfe des schon bekannten Programms über Mittelwerte und Varianzen leicht mit dem Computer ermitteln. Wenn man Versuche anstellt, bei denen mehrere Messwerte in einer Formel vorkommen, wird die Sache natürlich etwas komplizierter, da alle vorkommenden Grössen fehlerbehaftet sein können. Will man beispielsweise eine elektrische Leistung (in Watt) messen, so benötigt man dazu eine Strom- und eine Spannungsmessung, die beide nur mit einer bestimmten Genauigkeit durchgeführt werden können.

Der dabei entstandene Fehler kann durch eine sogenannte Taylorentwicklung angenähert werden. Anschaulich betrachtet handelt es sich hierbei um den Fehler, den man macht, wenn man von der wahren Messung (die Kurve in Bild 2) in einer bestimmten Richtung abweicht. Jede Dimension in der Raumkurve charakterisiert dabei eine Komponente der Messung. Der Trick ist nun der, dass die Kurve nicht in einer Ebene liegt, sondern in einer Art Gebirge, so dass eine Abweichung um denselben Wert in verschiedenen Richtungen durchaus verschiedene Fehler verursachen kann (einmal bleibt man in etwa in der Höhe der Kurve, beim anderen Mal findet man sich in einem Tal wieder, fernab von der Kurve).

Sei D(P) der Gesamtfehler der Messung, wenn P eine Funktion von z.B. zwei Messgrössen U (im Beispiel die Spannung) und I (Strom) ist. Dann gilt

$$D(P) = (3|dP/dU3|\delta U + 3|dP/dI3|\delta I)$$

Man nennt diese Formel das «Fehlerfortpflanzungsgesetz».

Die senkrechten Balken sind sogenannte «Beträge». Sie implizieren die Vorschrift, dass das Ergebnis zwischen den Balken mit positivem Vorzeichen zu versehen ist. Ansonsten kann es nämlich sein, dass ein Fehler in der einen Richtung (zwei Volt zuviel gemessen) den in der anderen Richtung (ein Ampere zuviel) aufhebt. Die Faktoren dP/dU und dP/dI ergeben sich aus der Differentialrechnung und sind praktisch der Multiplikator für die Fehler δU und δI. Die Delta sind vom Experimentator selber abzuschätzen. Steht zum Beispiel auf dem Spannungsmessgerät, dass die Anzeige eine Genauigkeit von 1% hat, so ist $\delta I = 0.01$ zu setzen. Sie sind also praktisch der Fehler, den man sich zutraut, begangen zu haben.

Was können wir mit dem obigen Ergebnis für D(P) eigentlich anfangen? Wenn sich D(P) = 0.1 ergab, heisst das dann, dass unsere Messung grundsätzlich mit einem Fehler von 10% behaftet ist? Diese Frage ist mit Nein zu beantworten, was man schon daran sieht, dass man bei verschiedenen Messungen meistens auch unterschiedliche Ergebnisse erhält. D(P) ist lediglich eine Abschätzung für den grössten Fehler, den man unter den ungünstigsten Umständen begehen kann. Der Fehler, den man wirklich begangen hat, wird also meistens kleiner als D(P) sein.

Um die Formel für das Fehlerfortpflanzungsgesetz in ein Programm fassen zu können, fehlt uns leider noch der Zugang zur Differentialrechnung, der notwendig ist, um die Quotienten dP/dU und dP/dI berechnen zu können. Aus diesem Grund wird das Problem auf einen späteren Teil dieser Beitragsreihe vertagt.

Stattdessen kommen wir jetzt wieder zu einem Thema, das man auch im Alltag einsetzen kann, ohne ein abgeschlossenes Mathematikstudium hinter sich gebracht zu haben.

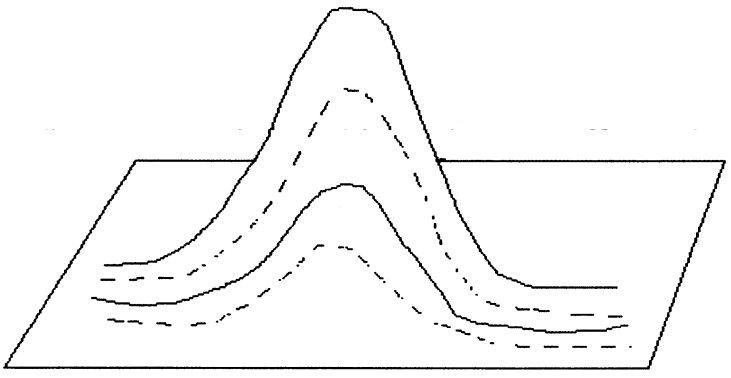


Bild 2: Taylorentwicklung anschaulich
Obwohl der Abstand zwischen den beiden gestrichelten Linien nicht gross ist, wie man in der Ebene sieht, wächst er bei Steigungen stark an. Geringe Abweichungen in einer Richtung können also völlig unterschiedliche Kurven ergeben.

Lineare und nichtlineare Regression

Hinter diesem komplizierten Ausdruck steckt schlicht die Absicht, durch eine Anzahl von Messpunkten eine Gerade zu legen, auf dass die mit ihr interpolierten Messwerte der Wahrheit möglichst nahe kommen.

Man steht öfters vor dem Problem, dass man eine grössere Zahl von Messungen vor sich hat, die eigentlich in einem linearen Zusammenhang miteinander stehen sollten (Beispiel: Messung des zurückgelegten Weges eines Autos mit konstanter Geschwindigkeit zu verschiedenen Zeitpunkten). Aufgrund von zufälligen Messfehlern wird man aber selten ein Ergebnis erhalten, bei dem alle Messpunkte auf einer Geraden liegen. Die Werte werden mehr oder weniger streuen.

Die Aufgabe der linearen Regression ist es nun, eine Gerade zu finden, deren Steigung den Zusammenhang der Messwerte optimal widerspiegelt. Die Grundidee ist einfach: Die Abweichung der Summe der Quadrate der Messwerte von der Geraden soll minimal werden. Dieses Problem wurde zuerst von Gauss mit Hilfe der Differentialrechnung gelöst. Mit seiner Formel kann man die Regressionsgerade berechnen. Dabei ist voranzustellen, dass die Gleichung einer Gerade die Form

$$Y = m\,X + b$$

hat. m ist die Steigung der Gerade, die so definiert ist: die Gerade hat die Steigung m = 3, wenn man 3 cm in Y-Richtung gehen muss, um 1 cm in X-Richtung zu kommen (siehe Steigungdreieck in Bild 3). b ist der sogenannte Achsenabschnitt. Er gibt den Y-Wert an der Stelle X=0 an.

Seien x(i) und y(i) die gemessenen Wertepaare, so erhält man folgendes Ergebnis:

$$m = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n (x_i - \bar{x})^2} \qquad b = \bar{y} - m\bar{x}$$

$\bar{y}$, $\bar{x}$ **sind die Mittelwerte,**
x_i, y_i **die Messwerte**

Diese Formeln kann man in die obige Geradengleichung einsetzen und erhält so eine Linie, die den wahrscheinlichsten Zusammenhang der Messwerte vermittelt. Bei sehr stark streuenden Messwerten stellt sich dabei die Frage, wie gut denn nun die berechnete Gerade in Wirklichkeit ist. Schliesslich kann es ja sein, dass die Verteilung der Messwerte keinem linearen, sondern z.B. einem exponentiellen Gesetz folgt. Dies hätte zur Folge, dass man zwar auch wieder eine Gerade berechnen kann, diese aber vollkommen falsche Werte ergeben würde. Aus diesem Grunde hier noch die Formeln für den sogenannten «Korrelationskoeffizienten». Er gibt quasi die Wahrscheinlichkeit an, dass die erhaltene Regressionsgerade den richtigen Zusammenhang der Messwerte darstellt:

$$r = \frac{\sum_i^n x_i\,y_i \;-\; \frac{1}{n}\left[\sum_i^n x_i \sum_i^n y_i\right]}{\sqrt{\sum_i^n x_i^2 - \frac{1}{n}\left(\sum_i^n x_i\right)^2}\;\sqrt{\sum_i^n y_i^2 - \frac{1}{n}\left(\sum_i^n y_i\right)^2}}$$

Ist r = +/- 1 so liegen spätere Messwerte mit Sicherheit auf der Regressionsgeraden. Für r=0 lässt sich keine Aussage über den Zusammenhang machen. Hierbei ist allerdings zu beachten, dass als Grundgesamtheit die Menge der Messwerte zugrunde gelegt ist. Misst man z.B. das Volumen V eines Gases bei konstantem Druck p in Abhängigkeit von der Temperatur T, so gilt für «ideale Gase» die Formel pV = RT

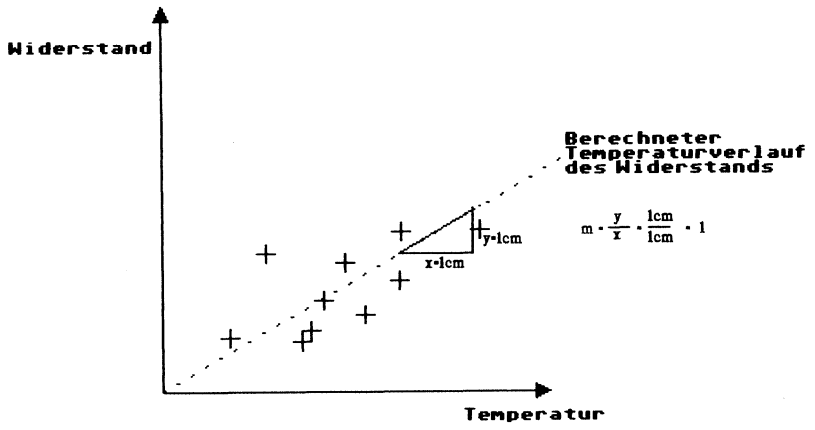


Bild 3: Ein Beispiel für lineare Regression

Ermittelt man dazu die Regressionsgerade, so werden alle Messwerte darauf liegen. Werden nun später zusätzliche Messungen gemacht, so gilt die Regressionsgerade natürlich nur noch in dem Fall, dass sich der Druck des Gases inzwischen nicht geändert hat, was z.B. beim Aufziehen eines Gewitters der Fall sein kann. Regression hat also nur dann einen Sinn, wenn sich die Versuchsbedingungen nicht ändern.

Was kann man nun mit der linearen Regression anfangen? Es ist möglich, Aussagen über das durchschnittliche Gewicht von Kindern in Abhängigkeit von ihrem Alter zu erhalten. Man kann für seine Firma den voraussichtlichen Umsatz fürs nächste Jahr berechnen, indem man den entsprechenden Umsatzwert auf der Regessionsgerade zugrunde legt. In derselben Weise ist es möglich, eine Antwort darauf zu erhalten, wann der Umsatz einen bestimmten Wert übersteigt. Hierbei ist anstatt des Regressionswertes bei einem bestimmten Jahr einfach der Regressionswert bei einem bestimmten Umsatz zu nehmen.

Generell gesagt wird man die lineare Regression immer dann verwenden, wenn man eine Gesetzmässigkeit in Form einer Geraden in ein Diagramm einzeichnen kann. Wenn alle Messwerte auf der Geraden liegen, so macht die Regression eine exakte Aussage über die interpolierten Werte, ansonsten gibt sie eben den wahrscheinlichsten Wert aus.

Das abgedruckte Listing ist ein solches Regressionsprogramm und gibt den Messwerteverlauf im Rahmen der Rechengenauigkeit wieder.

Es kommt sicher recht selten vor, dass eine Messreihe in Form einer Geraden aufgenommen werden kann. In der Regel wird man einen exponentiellen oder logarithmischen Verlauf vor sich haben. Was ist also zu unternehmen, wenn Ihnen am Stammtisch jemand weismachen will, dass die zeitliche Entwicklung des Schaums im Bierglas linear von der Zeit abhängt (das heisst, wenn nach zwei Minuten nur noch die Hälfte des ursprünglichen Schaums vorhanden ist, müsste er nach vier Minuten ganz verschwunden sein)? Auch dieses Problem ist mit unserem Programm zu meistern! Der Trick besteht darin, die Messwerte zu linearisieren, also quasi aus einer «krummen» Kurve eine Gerade zu machen. Sie werden dann feststellen, dass der Bierschaum exponentiell abnimmt.

Die Technik der Linearisierung ist recht simpel: Hat man zum Beispiel eine Abhängigkeit der Form Y = exp(m*X+b), dann setzt man den natürlichen Logarithmus darauf an und erhält so eine Abhängigkeit LN(Y) = m*X + b. Definiert man jetzt LN(Y) := LN(Z) und dann Y := Y, so kommt man wieder auf die bekannte Geradengleichung Y = m*X + b.

In ähnlicher Weise wird vorgegangen, wenn man eine Logarithmusfunktion linearisieren will: man lässt eine Exponentialfunktion darauf wirken. Den Rest ersehen Sie bitte aus dem Listing.

Natürlich ist es etwas ermüdend, wenn man hundertmal den Logarithmus aus einer Formel zieht und immer noch keine Gerade erhält, da man die falsche Basis des Logarithmus gewählt hat. Diese Arbeit kann auch der Computer übernehmen.

Im Programm wird das so gemacht, indem die Y-Werte solange verändert werden, bis sie am besten mit den

X-Werten übereinstimmen. Da sich negative und positive Abweichungen herauskürzen könnten, vergleicht man am besten anhand der Varianzen, die ja in jedem Fall positiv sind. Das Programm merkt sich die Steigung m, bei der die Uebereinstimmung am besten war und gibt sie aus, sobald die Abweichungen grösser werden (die beste Varianz wird dazu in der Variablen KR gespeichert, die Steigung in der Variablen E1). Anschliessend wird mit den wahrscheinlichsten Y-Werten noch einmal durchgerechnet und die wichtigsten Informationen wie Steigung und Korrelationskoeffizient ausgegeben. Die y-Mittelwerte und Standardabweichungen, die ausgedruckt werden, gelten für die linearisierten Werte.

Das Programm macht darauf aufmerksam, dass bei normaler Regression falsche Werte herauskommen können. Statt der richtigen Funktion Y=2^(3*X+1) ergibt sich z.B. die Form 8.1^X. Das liegt daran, dass der Computer alles, was in der Klammer steht, für die Variable X hält. Die Meisterung dieses Problems ist nicht trivial, siehe die zweite Anmerkung am Ende dieses Textes. Wir können aber auf relativ einfache Weise doch einen Ueberblick erhalten, inwieweit die gefundene Funktion der Realität nahe kommt, indem wir eine «selbstdefinierte Funktion» einsetzen. Hierbei geben wir die Steigung und den Achsenabschnitt schon vor, im obigen Beispiel also m=3 und b=1. Die Basis der Exponentialfunktion sucht der Rechner dann wieder selber.

Es ist darauf hinzuweisen, dass im Programm Formeln vorkommen, bei denen Logarithmen im Nenner stehen. Da diese zuweilen gleich Null werden, stürzt das Programm ab. Eine Abhilfe besteht meist darin, dass man das Intervall, in welchem die richtige Abhängigkeit gesucht wird, enger eingrenzt. Besondere Schwierigkeiten entstehen im Intervall zwischen Null und Eins. Versuchen Sie, einen Algorithmus zu finden, der den Absturz umgeht, indem er z.B. auf die Gefahr hinweist!

Eine Anmerkung für Nicht-IBM-Programmierer: die Funktion LOG ist in IBM-BASIC der Natürliche Logarithmus, nicht der Zehnerlogarithmus!

Eine zweite Anmerkung für Mathematiker unter den Lesern: wie Sie sicher schon bemerkt haben, handelt es sich hier nur um ein einfaches Hilfsmittel, um einen Ueberblick über nichtlineare Zusammenhänge zu gewinnen. Mit «richtiger» nichtlinearer Regression hat das Ganze natürlich nichts zu tun! Die Erklärung dieser Technik übersteigt aber die Anforderungen dieses Kurses.

Zurück zu den Anwendungen. Die Regression bringt dann einen Nutzen, wenn man einen gesetzmässigen Zusammenhang der Messwerte untereinander voraussetzt. Beim Bierproblem werden Sie bei einer Auftragung der Werte auf logarithmischem Papier feststellen, dass sie recht stark streuen können. Dies ist vor allem bei Leichtbieren der Fall, bei denen die Schaumblasen ziemlich gross werden können. Falls Sie einen Augenblick zu spät ablesen, ist die Blase zerplatzt und die Schaumhöhe unter Umständen nur noch halb so gross wie vorher. Bei Eingabe der Werte ins Regressionsprogramm ergibt sich aber doch näherungsweise eine Exponentialfunktion.

Ein Paradebeispiel für den falschen Einsatz der Regression ist das Geschehen an den Wertpapierbörsen. Etwa die letzten drei Jahre verzeichnete man einen fast linearen Anstieg der Aktien. Stellen Sie sich vor, Sie hätten am 18.10.87 das Regressionsprogramm gestartet, den Knopf für lineare Regression gedrückt und den Wert einer bestimmten Aktie am 20.06.88 berechnen lassen (Sie können natürlich kein Datum regressieren, stattdessen aber die Differenz in Tagen zwischen zwei Messwerten). Mit den dabei erhaltenen Ergebnissen müssten Sie sich ernsthafte Gedanken darüber machen, was mit dem Gewinn anzufangen ist. Der Haken an der Sache ist der, dass sich Börsenkurse nicht vorhersagen lassen, da sie keinen (erkennbaren) Gesetzmässigkeiten unterliegen: am 19.10.87 sanken die Kurse durchschnittlich um etwa 15%. □

COMPUTER-SPLITTER

Telefax ab DEC-Rechner

Mit dem Kencom-Telefax-Interface ist es möglich, Text direkt ab jedem beliebigen DEC-Rechner (PDP/VAX), oder auch anderen Rechnern mit RS232C-Schnittstelle zu übermitteln. Die Kosten einer mit Fax übermittelten A4-Seite betragen ca. 10 % einer mit Telex übermittelten Seite, was eine sehr starke Kosteneinsparung ermöglicht.

Als Fax wurde ein Telefax von Sanyo gewählt, der zusätzlich mit dem intelligenten Interface der Kencom ausgerüstet wurde, welches die gesammte Ueberwachung des Wählvorgangs sowie der Datenübermittlung übernimmt (weitere Geräte in Vorbereitung). Zur Gestaltung des Textes stehen die bekannten Möglichkeiten wie Double Height Double Width, Single Height Double Width, Double Height Single Width und NRC Charaktersets mit den original DEC Esc-Sequenzen zur Verfügung.

Bei einem Unterbruch oder einer schlechten Verbindung macht das Interface automatisch ein Redial und ein Reprint der letzten Seite oder des gesammten Fax.

Die Ansteuerung ist befehlskompatibel zum HTU Telex Interface, es können somit bereits bestehende Softwarepakete verwendet werden. Als Option kann ein Nadeldrucker angeschlossen werden, auf dem der Datentransfer inklusive angewählte Nummer und Text, oder für die Softwareentwicklung die gesammten Steuerzeichen ausgedruckt werden.

Der jeweilige Zustand oder auch Fehlermeldungen werden über eine LED-Display auf der Front der Fax-Interfacebox angezeigt. Alle Funktionen des Interface sind mit Hilfe eines internen Monitors mit einem Help und Setup veränderbar und können nichtflüchtig gespeichert werden. Info: Kencom AG, Würzgrabenstr. 6, 8048 Zürich, Tel. 01/432'23'44. □

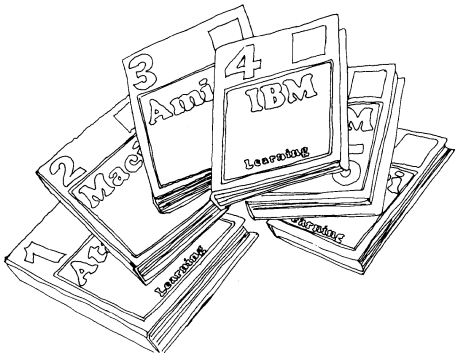
«In the very near future»

(570/fp) An einer Medientagung in Zürich wurde Dr. Toshi Doi kürzlich über den Stand der Entwicklung bei der löschbaren CD befragt. Als Miterfinder der CD und Chef der entsprechenden Entwicklungsgruppe bei Sony ist Doi einer, der es eigentlich wissen müsste. Nun, nach einem kurzen verbalen Rundgang durch seine Laboratorien meinte er zur löschbaren CD in einer für die Branche schon fast erstaunlichen Verbindlichkeit, seine Firma könne entsprechende Ankündigungen «in the very near future» machen. □

Lotus 1-2-3 Vers. 3.0 verspätet

(469/eh) Die von Lotus angekündigte neue Version 3.0 des Tabellenkalkulationsprogrammes Lotus 1-2-3 wird erst im dritten und nicht schon im zweiten Quartal 1988 auf den Markt gebracht werden. Der Hauptgrund für die Verzögerung scheint das Bemühen der Entwickler zu sein, den Code des Programmes zu kürzen und zu optimieren. Gelingt es, den Code zu verkürzen, so steht entsprechend mehr Speicherplatz für die Kalkulationstabellen zur Verfügung; die Optimierung kann zu einer noch rascheren Programmabarbeitung führen. □

Der Leichtere Weg zum Erfolg



Wir bieten Ihnen ein leichtverständliches und ausgewogenes, auf Sie zugeschnittenes Lehrmaterial an.

EDV-Fachwissen gehört morgen schon zu Allgemeinbildung. Jeder weiss heute schon wie wichtig es ist, dass ein solches Fachwissen einem im Berufsleben weiterbringt und dass die EDV im heutigen Alltag immer wichtiger wird.
Zur Zeit stehen Ihnen 35 Lehrgänge zur Verfügung - für Hobby und Beruf. Diese bringen Sie sicher und ohne Lohnausfall ans Ziel. Denn in der heutigen Zeit gehört ein fundamentiertes EDV-Grundwissen zum Berufsalltag, da immer mehr danach gefragt wird. Unsere Kurse sind praxisnah und von qualifizierten Fachleuten aufgebaut.

Profitieren Sie von den Vorteilen eines guten Fernunterrichtes.

Sie lernen zu Hause in den eigenen vier Wänden.
Keinen umständlichen Schulweg, keine fixe Schulzeit.
Sie arbeiten selbständig und unabhängig.
Kein Verdienstausfall, keine Berufsunterbrechung.
Anspornende Unterrichtsbetreuung.
Frei wählbares Studientempo.

Zu Ihrem Studien-Erfolg trägt das leicht verständliche Lehrmaterial bei, welches optimal und sehr aufwendig gestaltet ist.

PC Grundkurs I 6 Lektionen à je Fr. 75.-
MS-DOS Kurs I 12 Lektionen à je Fr. 45.-
Desktop Publishing 12 Lektionen à je Fr. 45.--
Page Maker MacIntosch 12 Lektionen à je Fr. 75.-
Textverarbeitung Intensiv 6 Lektionen à je Fr. 125.-
Lotus 1-2-3 24 Lektionen à je Fr. 85.-
Amiga und C 12 Lektionen à je Fr. 65.-
GFA Grundkurs Atari 6 Lektionen à je Fr. 45.-

Gewünschtes bitte ankreuzen

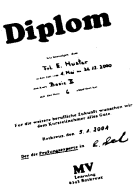
TESTSTUDIUM ↓ **Definitive Bestellung** ↓

- ❒ ❒ **PC Grundkurs I**
- ❒ ❒ **MS-DOS Kurs I**
- ❒ ❒ **Desktop Publishing**
- ❒ ❒ **Page Maker MacIntosh**
- ❒ ❒ **Textverarbeitung Intensiv**
- ❒ ❒ **Lotus 1-2-3**
- ❒ ❒ **Amiga und C**
- ❒ ❒ **GFA Grundkurs Atari**

Ich möchte weiterkommen und wähle zwischen einem Teststudium oder einer definitiven Bestellung.
Beim Teststudium kann ich den ausgewählten Kursus während drei Wochen unverbindlich testen. Sagt er mir zu, behalte ich den Kursus und meine Studienzeit beginnt - oder aber ich sende die erhaltenen Unterlagen in der festgesetzten Frist zurück und es entstehen mir keinerlei Kosten.

❒ **Bitte senden sie mir weitere Kursunterlagen**

Jeder Kursabsolvent erhält das Abschlussdiplom.



Absender:

NAME _____

VORNAME _____

STRASSE

PLZ/ORT _____

DATUM _____

UNTERSCHRIFT _____

Coupon ausschneiden und einsenden an: MV-Learning, Grundstrasse 20, 6343 Rotkreuz

Vom Umgang mit dBase III PLUS (Nachtrag)

Schon zu Beginn unseres Lehrganges haben wir zum Ausdruck gebracht, dass dBase eigentlich eine strukturierte Programmiersprache ist. Im Verlaufe der Fortsetzungsreihe wurde aber diesem Umstand recht wenig Beachtung geschenkt. Dieses Versäumnis wollen wir nun als Nachtrag zu dieser Serie nachholen.

Heinz Kastien

Warum, so werden sich sicherlich einige unserer Leser fragen, wurde die Strukturierung der dBase-Programme so schmählich vernachlässigt? Die Antwort ist relativ einfach. Gut zwei Drittel der M+K-Leser bevorzugen noch immer BASIC für Ihre Programmierungen, so wissen wir aus unseren Umfragen, und es galt, diesem Leserkreis eine neue Programmiersprache so zu vermitteln, wie er es vom BASIC her gewohnt ist. Zwangsläufig kamen dadurch einige Stärken des dBase etwas zu kurz.

Für alle, die bisher in Pascal, dBase oder C programmiert haben, wollen wir uns nun speziell der strukturierten Programmierung des dBase zuwenden. Um keine unnötige Verwirrung zu stiften, haben wir aus den bisher veröffentlichten Programmen das Mutations- und das Listenprogramm ausgewählt und zeigen an diesen Beispielen die Feinheiten des dBase. Aber nicht nur die Strukturierung sondern auch andere Finessen sollen hier diskutiert werden. Durch die Verwendung bereits bekannter Programmstrukturen ist es so auch den weniger versierten Lesern möglich, Vergleiche zu ziehen.

Was versteht man aber nun unter der Strukturierung eines Programms und welche Aufgabe erfüllt sie? dBase kennt, wie Turbo-BASIC, Pascal oder Logo, keine Zeilennummern, sondern die Programmteile und Subroutinen sind durch Label gekennzeichnet. Dadurch ist es bei dBase relativ schwierig zusammenhängende DO WHILE ...ENDDO-Schleifen, IF...ENDIF oder DO CASE...CASE ...ENDCASE-Strukturen zu erkennen, durch Einrücken der einzelnen Befehlszeilen wird jedoch die Zusammenhörigkeit eindeutig erkennbar und erleichtert die Programmierung und die Fehlersuche wesentlich. Inzwischen sind auch Hilfsprogramme im Handel, die ein bestehendes Programm automatisch strukturieren.

Einem weiteren Punkt muss aber beim dBase noch Beachtung geschenkt werden. Da in dieser Programmiersprache unbedingte Sprungbefehle, wie GOTO <Zeilennummer> oder bedingte Sprünge wie IF...THEN <Zeilennummer> Befehl, des BASIC nicht bekannt sind, ist es unerlässlich, ein Flussdiagramm der Programmstruktur zu erstellen. Dadurch wird zwar der Programmweg eindeutig erkannt, die verschiedenen DO...WHILE-Schleifen kommen jedoch nicht klar zum Ausdruck. Besser geeignet dafür ist ein Struktogramm.

Alle Programme auf Diskette

Die komplette Programmsammlung kann 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. □

```
** M+K Vereinsadressmutationsprogramm by H. Kastien 19.08.1987 Vers-01 / HR **

USE Mitglied INDEX NamInd              && Dateien laden
RESTORE FROM Daten

DO WHILE .T.

   CLEAR                               && Masken-Aufbau
   sName = SPACE(40)
   Antwort = 'M'

     SET COLOR TO /W
   @ 2,7 SAY 'M+K'                     && Kopf-Zeile
     SET COLOR TO B/
   @ 2,30 SAY 'Vereinsverwaltung'
   @ 2,69 SAY 'H.Kastien'
     SET COLOR TO +W/

   @ 5,7 SAY 'Mitgliedname'
     SET COLOR TO W/
   @  5,26 SAY ':'
   @  7,7  SAY 'Vorname             :'
   @  9,7  SAY 'Strasse             :'
   @ 11,7  SAY 'PLZ :          Ort :'
   @ 13,7  SAY 'Geburtsdatum        :              Eintrittsdatum :'
   @ 15,7  SAY 'Telefonnummer       :'
   @ 17,7  SAY 'Mitgliederart       :'
   @ 19,7  SAY 'Mitgliederbeitrag   :                Fr.  Zahlung :'
   @ 21,7  SAY 'Bemerkung           :'
   @  1,0 TO 24,79 DOUBLE              && Doppelter Rahmen
   @  3,1 TO 3,78                      && Horizontale Linie
   @  5,30 GET sName
   READ
```

```
FIND '&sName'                          && Satz MIT 'sName' suchen
IF EOF()

   § 21,30 SAY 'Der Name ' + TRIM(sName) + ' ist NICHT in der Datei !'
   Antwort = ' '
   DO WHILE .T.
      Antwort = 'N'
      § 23,30 SAY 'Weiter mit N (Neubeginn) oder M zum Menu !'GET Antwort
      READ
      IF UPPER(Antwort) = 'N' .OR. UPPER(Antwort) = 'M'
         EXIT
      ENDIF
   ENDDO

   IF UPPER(Antwort) = 'M'
      RETURN
   ELSE
      LOOP
   ENDIF

ELSE

   DO WHILE .NOT. EOF() .AND. Name = sName
      §  5,30 SAY Name
      §  7,30 SAY VorN
      § 09,30 SAY Stra
      § 11,14 SAY PoLZ
      § 11,30 SAY OrtB              &&  Satz-Daten editieren
      § 13,30 SAY gDat
      § 13,59 SAY eDat
      § 15,30 SAY TelN
      § 19,30 SAY Mita
      § 19,58 SAY Zahl
      § 21,30 SAY Beme

      Antwort = 'J'

      DO WHILE .T.
         § 23,30 SAY 'Ist dies die richtige Adresse ? (J/N) 'GET Antwort
         READ
         IF UPPER(Antwort) = 'J' .OR. UPPER(Antwort) = 'N'
            EXIT
         ENDIF
      ENDDO

      IF UPPER(Antwort) = 'J'
         EXIT
      ENDIF

      SKIP                           &&  Zum nächsten Satz Gehen
   ENDDO
ENDIF

IF Name <> '&sName'
   LOOP                              &&  Wenn der nächste Satz NICHT
ENDIF                                &&  den RICHTIGEN Namen enthält

Antwort = 'N'

DO WHILE .T.
   § 23,30 SAY 'Wollen Sie diese Adresse löschen ? ( J/N ) ' GET Antwort
   READ
   IF UPPER(Antwort) = 'J' .OR. UPPER(Antwort) = 'N'
      EXIT
   ENDIF
ENDDO

IF UPPER(Antwort) = 'J'
   DELETE                             &&  Satz löschen
   PACK
   LOOP
ENDIF

eName = Name
eVorN = VorN
ePoLZ = PoLZ
eOrtB = OrtB
eStra = Stra
egDat = gDat
EeDat = eDat
eTelN = TelN
eMita = Mita
eBeit = Beit
eZahl = Zahl
eBeme = Beme
```

```
            @ 23,30 SAY SPACE(48)
            @ 22,21 SAY SPACE(40)
            @  5,30 GET eName
            @  7,30 GET eVorN
            @  9,30 GET eStra
            @ 11,14 GET ePoLZ PICTURE '######' RANGE PoLZMin, PoLZMax
            @ 11,30 GET eOrtB
            @ 13,30 GET EgDat
            @ 13,59 GET EeDat
            @ 15,30 GET eTelN
            @ 17,30 GET eMita
            @ 19,30 GET eBeit RANGE BeitMin, BeitMax
            @ 19,58 GET eZahl
            @ 21,30 GET eBeme
            READ

         REPLACE Name WITH eName, VorN WITH eVorn, eDat WITH EeDat
         REPLACE Stra WITH eStra, PoLZ WITH ePoLZ, TelN WITH eTelN
         REPLACE OrtB WITH eOrtB, gDat WITH EgDat, Beit WITH eBeit
         REPLACE Mita WITH eMita, Zahl WITH eZahl, Beme WITH eBeme

         DO WHILE .T.
            @ 23,30 SAY SPACE(48)
            @ 23,30 SAY 'Wollen Sie weitere Mutationen ? ( J/N ) ' GET Antwort
            READ
            IF UPPER(Antwort) = 'J' .OR. UPPER(Antwort) = 'N'
               EXIT
            ENDIF
         ENDDO

         IF UPPER(Antwort) = 'N'
            CLOSE ALL
            RETURN                          && Programm-Ende
         ENDIF

      ENDDO
```

Aber aufgepasst: Fehlerhafte Programme mit einem Struktogramm oder durch eine Strukturierung laufen keineswegs besser. Die Strukturierung ist lediglich ein Weg, ein Programm zu optimieren. Der Vergleich mit den berühmtberüchtigten Spagettiprogrammen des BASIC scheint daher nicht ganz unangebracht. Ein solches BASIC-Programm mit unzähligen GOTO, GOSUB und IF...THEN läuft zwar auch, ein durchdacht gestaltetes BASIC-Programm ist jedoch übersichtlicher und schneller.

Listenprogramm

Das überarbeitete und strukturierte Listenprogramm unterscheidet sich von der bereits veröffentlichten Version nur unwesentlich, dies nicht zuletzt durch die Tatsache, dass es nur aus einem Block besteht. Man erkennt aber an den eingerückten Zeilen deutlich die verschiedenen DO...WHILE-Schleifen. In diesem überarbeiteten Programm wurden die Adressen nicht sortiert, das Sortieren der Datensätze kann dann entfallen, wenn die Hauptdatei nach einem alphanumerischen Feld indiziert wird. Die wesentlichste Aenderung ist der Ausdruck der Adressen, dessen DO.. WHILE-Schleife nicht nur nach dem EOF() fragt, sondern auch nach der .AND. verknüpten Variablen «anzahl». Diese wird nach jeder ausgedruckten Adresse um 1 erhöht. Wird diese Variable grösser als eine ganzzahliges Vielfaches von 50 (maxzahl), verlässt das Programm die Schleife, der Seitenzähler Seite wird um 1 und die Variable «maxzahl» um 50 erhöht, sowie durch den Befehl EJECT ein Seitenvorschub erzwungen. Neben diesen eigentlichen Programmänderungen wurden einige, bisher nicht veröffentlichte Befehle verwendet.

SET COLOR TO
ermöglicht bei Farbmonitoren die Auswahl von Farben bzw. die Wahl der Farbintensität und bei monochromen Monitoren die Auswahl von Bildschirmattributen und der Helligkeit der Anzeige. Der komplette Befehl lautet:

SET COLOR TO <Standard>,
<Hervorhebung>,Bildschirmrand

SET COLOR TO 6/1,7,4,6
oder
SET COLOR TO GR/B,W/R,GR
stellt die Zeichen auf dem Bildschirm im Standardmode Gelb auf blauem Hintergrund dar, hervorgehobenen Zeichen erscheinen Weiss auf rotem Hintergrund, der Rahmen ist in jedem Fall Gelb. Mit der Option + werden die Zeichen intensiv hervorgehoben, mit der Option + blinkt das Zeichen.

SET COLOR TO /W
schaltet im monochromen Modus auf inverse Darstellung und mit

SET COLOR TO W
wird wieder auf normale Darstellung zurückgeschaltet.

SET DEVICE TO
legt fest, ob die Ausgabe auf dem Bildschirm oder auf einem Drucker erfolgen soll.

SET DEVICE TO SCREEN
ist die Grundeinstellung, alle Ereignisse werden auf dem Bildschirm ausgegeben.

SET DEVICE TO PRINTER
leitet alle Ereignisse auf einen Drucker um, GET Befehle werden mit diesem Modus ignoriert.

```
*****************************************************************************
*  M+K Vereins-Listen-Programm by H. Kastien 19.08.1987 Vers-02 01.01.88 HR  *
*****************************************************************************
*  Erstellen von Adress-Listen ab Datei 'Mitglied'

CLEAR                              && Bildschirm löschen

§ 2,7 SAY 'M+K'                    && Programm-Kopf
§ 2,31 SAY 'Vereinsverwaltung'
§ 2,64 SAY 'H.Kastien'
  SET COLOR TO W+                  && Fett-Darstellung
§ 5,15 SAY 'Die Adressliste wird gedruckt  /  WARTEN !'

SET DEVICE TO PRINT                && Ausgabe an Drucker leiten
SET MARG TO 4                      && Drucker-Rand anpassen
USE Mitglied INDEX NamInd          && Datei laden mit Namen-Index
STORE 1 TO seite                   && Variablen definieren
anzahl = 1
maxzahl = 51
§ PROW(),0 SAY CHR(27) + "C" + CHR(72)          &&   72 Zeilen / Seite

DO WHILE .NOT. EOF()                             &&  Seiten - Schleife
   § PROW(),0  SAY CHR(27) + CHR(64)             &&  Drucker normieren
   § PROW(),7  SAY 'Datum : ' + DTOC(DATE())     &&  Aktuelles Datum
   § PROW(),75 SAY 'Seite : ' + STR(Seite,2)     &&  Seiten-Nummer
   § PROW(),0  SAY CHR(27) + "!" + CHR(32)       &&  Breit-Schrift
   § PROW() + 3,5 SAY 'M+K  Vereins-Adress-Liste'&&  Programm-Titel
   § PROW() + 1,5 SAY REPLICATE('_',25)          &&  Titel unterstreichen
   § PROW() + 0,0 SAY CHR(27) + "!" + CHR(4)     &&  Schmal-Schrift
   § PROW() + 3,4 SAY 'Nr.  Mitgliedername'      &&  Ueberschrift-Zeile
   § PROW() , 46 SAY 'Vorname'
   § PROW() , 66 SAY 'POLZ'
   § PROW() , 72 SAY 'Ortsbezeichnung'
   § PROW() , 109 SAY 'Strasse und Nr.'
   § PROW() + 1,3 SAY REPLICATE('_',135)    &&  Ueberschrift unterstreichen
   § PROW() + 1,0 SAY ' '                   &&  Abstand zu 1. Adresse

   DO WHILE .NOT. EOF() .AND. anzahl < maxzahl          &&  Zeile ausgeben
      § PROW() + 1,3 SAY STR(anzahl,3) + ".  " + name + '   ' + vorn
      § PROW(),PCOL() SAY STR(POLZ,4) + '  ' + OrtB + '   ' + Stra
      SKIP                                  &&  nächster Datei-Satz
      anzahl = anzahl + 1
   ENDDO (Ende Zeile ausgeben)

   maxzahl = maxzahl + 50                   &&  Zähler um 50 Zeilen erhöhen
   seite = seite + 1                        &&  Seiten-Nummer um 1 erhöhen
   EJECT                                    &&  Seiten-Vorschub

ENDDO  (Ende Seiten-Schleife)

§ PROW(),0 SAY CHR(27) + "§"                &&  Drucker normieren

SET DEVICE TO SCREEN                        &&  Ausgabe an Bildschirm leiten
§ 5,0 CLEAR
§ 5,15 SAY 'Liste dem Drucker entnehmen ! ' &&  Ende Programm
  SET COLOR TO W                            &&  Normale Schrift wählen

RETURN
```

SET MARGIN TO
justiert für alle Ausdrucke auf dem Drucker den linken Rand.

SET MARGIN TO 10
Alle Ausdrucke werden um 10 Zeichen nach rechts versetzt. Der Defaultwert ist Null.

EJECT
erzeugt bei einem Drucker einen Seitenvorschub, dieser Befehle ist identisch mit der Befehlsfolge

SAY CHR(27)+CHR(12)

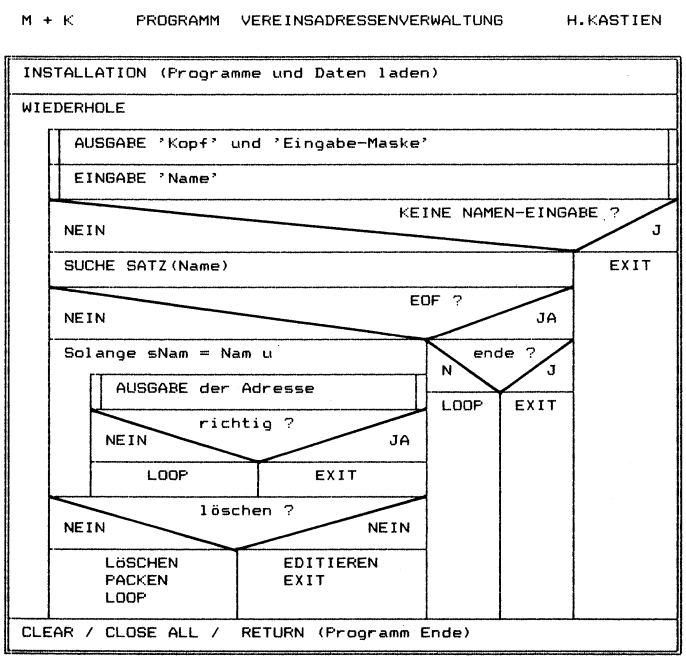
REPLICATE
wiederholt ein Zeichen. Der Befehl

SAY REPLICATE («*»,25)
druckt 25 mal den Stern aus.

Mutationsprogramm

Das Mutationsprogramm aus M+K 87-6 ist komplexer aufgebaut, als das oben beschriebene Listenprogramm, daher kann an diesem Beispiel die Strukturierung besser demonstriert werden. Je komplexer ein Programm ist, desto mehr Lösungswege ergeben sich. Wir möchten zwei verschiedene Varianten zeigen. Im ersten Beispiel ist das Originalprogramm übernommen worden, jedoch in der dBase-üblichen Schreibweise. Diese Version läuft jedoch nicht unter dBase III, da der Befehl «DOUBLE» für Doppellinien nur im dBase III PLUS implementiert ist.

Im zweiten Beispiel des Mutationsprogramms wurde ebenfalls eine Strukturierung vorgenommen, gleichzeitig aber einige Routinen leicht verändert, so muss das «Menü» nicht mittels des Buchstabens «M» aufgerufen werden, sondern das Mutationsprogramm bricht automatisch ab, sobald bei der Namensabfrage dieser als Nullstring mit der RETURN-Taste quittiert wird. Weiterhin wur-



de die eigentliche Mutationsroutine stark vereinfacht, indem die Variablen mittels GET direkt abgefragt werden, ohne eine vorherige Umbenennung. Da bei beiden Programmen am Flussdiagramm keine Veränderungen vorgenommen wurden, kann für beide Programme das gleiche Struktogramm verwendet werden.

Es gibt noch andere Möglichkeiten, dieses Problem zu lösen, z.B. indem immer wiederkehrende Programmteile, wie das Listen auf dem Bildschirm oder die Masken, in einer separaten Prozedur abgespeichert sind, die dann von der Hauptprozedur aufgerufen werden. Jedoch würde eine Besprechung aller Möglichkeiten zu weit führen.

Wir beenden mit diesem Nachtrag unsere Reihe über dBase III PLUS und hoffen, dass wir sowohl den Neulingen dieser interessanten Programmiersprache als auch den Kennern etwas bieten konnten, den Neulingen einen Einstieg und den Profis vielleicht diesen oder jenen Tip. Wir haben keinesfalls beabsichtigt, mit der Vereinsadressverwaltung ein professionelles Programm anzubieten. Vielmehr sollte dieses Programm Mittel zum Zweck sein, nämlich an einem praktischen Beispiel möglichst umfänglich dBase zu erklären und Anregungen für ähnlich Probleme zu geben.

```
*****************************************************************************
*  M+K  VerAdressMutaProgr by H. Kastien 19.08.1987 Vers-02  02.12.87 /  HR  *
*****************************************************************************

USE Mitglied INDEX NamInd              && Dateien laden
RESTORE FROM Daten

DO WHILE .T.
   CLEAR                               && Masken-Aufbau
   sName = SPACE(40)
     SET COLOR TO /W
   § 1,0 SAY SPACE(80)                 && Kopf-Zeile
   § 1,5 SAY 'M + K'
   § 1,30 SAY 'Vereinsverwaltung'
   § 1,69 SAY 'H.Kastien'
     SET COLOR TO W+
   §  4,5 SAY 'Mitgliedname'           &&  Maske
      SET COLOR TO W
   §  4,26 SAY ':'
   §  6,5  SAY 'Vorname              :'
   §  8,5  SAY 'Strasse              :'
   § 10,5  SAY 'PLZ :            Ort :'
   § 12,5  SAY 'Geburtsdatum         :              Eintrittsdatum :'
   § 14,5  SAY 'Telefonnummer        :'
   § 16,5  SAY 'Mitgliederart        :'
   § 18,5  SAY 'Mitgliederbeitrag    :                Fr.  Zahlung :'
   § 20,5  SAY 'Bemerkung            :'
   §  4,30 GET sName
   § 23,0 TO 23,79                     &&  Trenn-Strich / Status-Zeile
   § 24,30 SAY 'Name eingeben, oder ENTER für Ende'
   READ
   IF LEN(TRIM(sName)) = 0             && OHNE Eingab : Schleife verlassen
      CLOSE ALL
      EXIT
   ENDIF
   § 24,30
   FIND '&sName'                       &&  Satz MIT 'sName' suchen
   IF EOF()                            &&  Wenn Satz NICHT vorhanden
      § 22,30 SAY 'Der Name ' + TRIM(sName) + ' ist NICHT in der Datei !'
      Antwort = ' '
      DO WHILE .T.
         Antwort = 'N'
         § 24,30 SAY 'Programm abbrechen ?  ( J/N )  'GET Antwort
         READ
         IF UPPER(Antwort) = 'N' .OR. UPPER(Antwort) = 'J'
            EXIT
         ENDIF
      ENDDO ( Ende der Auswahl-Schleife )
```

```
   IF UPPER(Antwort) = 'M'
      EXIT
   ELSE
      LOOP
   ENDIF
ELSE                                  &&  Wenn Satz vorhanden
   DO WHILE .NOT. EOF() .AND. Name = sName
      $  4,30 SAY Name
      $  6,30 SAY VorN
      $ 08,30 SAY Stra
      $ 10,14 SAY PoLZ
      $ 10,30 SAY OrtB                &&  Satz-Daten editieren
      $ 12,30 SAY gDat
      $ 12,59 SAY eDat
      $ 14,30 SAY TelN
      $ 16,30 SAY Mita
      $ 18,30 SAY Beit
      $ 18,58 SAY Zahl
      $ 20,30 SAY Beme


      Antwort = 'J'
      DO WHILE .T.
         $ 24,30 SAY 'Ist dies die richtige Adresse ? (J/N) 'GET Antwort
         READ
         IF UPPER(Antwort) = 'J' .OR. UPPER(Antwort) = 'N'
            EXIT                      &&  Auswahl-Schleife verlassen
         ENDIF
      ENDDO ( Ende der Auswahl-Schleife )
      IF UPPER(Antwort) = 'J'
         EXIT                         &&  Editier-Schleife verlassen
      ENDIF
      SKIP                            &&  Zum nächsten Satz Gehen
   ENDDO ( Ende der Editier-Schleife )
ENDIF
IF Name <> '&sName'
   LOOP                               &&  Wenn der nächste Satz NICHT
ENDIF                                 &&  den RICHTIGEN Namen enthält
Antwort = 'N'
DO WHILE .T.
   $ 24,30 SAY 'Wollen Sie diese Adresse löschen ? ( J/N ) ' GET Antwort
   READ
   $ 24,30
   IF UPPER(Antwort) = 'J' .OR. UPPER(Antwort) = 'N'
      EXIT
   ENDIF
ENDDO ( Ende Auswah-Schleife )



   IF UPPER(Antwort) = 'J'
      DELETE                          &&  Satz löschen
      PACK
      LOOP                            &&  Zum Schlaufen-Beginn
   ENDIF
   $ 22,21
   $  4,30 GET Name
   $  6,30 GET VorN
   $  8,30 GET Stra
   $ 10,14 GET PoLZ PICTURE '######' RANGE PoLZMin, PoLZMax
   $ 10,30 GET OrtB
   $ 12,30 GET gDat
   $ 14,59 GET eDat
   $ 14,30 GET TelN
   $ 16,30 GET Mita
   $ 18,30 GET Beit RANGE BeitMin, BeitMax
   $ 18,58 GET Zahl
   $ 20,30 GET Beme
   $ 24,30 SAY 'Aendern, oder  < ESC > wenn alles richtig ist.'
   READ
   DO WHILE .T.
      Antwort = 'J'
      $ 24,30
      $ 24,30 SAY 'Wollen Sie weitere Mutationen ? ( J/N ) ' GET Antwort
      READ
      IF UPPER(Antwort) = 'J' .OR. UPPER(Antwort) = 'N'
         EXIT
      ENDIF
   ENDDO ( Ende der Auswahlschleife )
   IF UPPER(Antwort) = 'N'
      EXIT                            &&  Schleife verlassen
   ENDIF
ENDDO ( Ende Hauptschleife )
CLOSE ALL                             &&  Alle Dateien schliessen
RETURN                                &&  Rückkehr ins Hauptprogramm
```

Testprogramme für VGA

Es ist kein Geheimnis mehr: der neue Farbgrafikstandard heist VGA. Nach dem CGA (Color Graphics Adapter) mit müder 8x8-Punktmatrix und 8/16 Farben, wobei nur zwei bei der höchsten Grafikauflösung von 640x200 Punkten einsetzbar waren, folgte der wesentlich bessere EGA (Enhanced Graphics Adapter) mit 640x350 Punkten, wobei bereits 16 aus 64 Farben wählbar sind und jedes Zeichen in einer 8x14-Punkte-Matrix dargestellt wird. Mit dem PS/2-System von IBM setzt sich nun der VGA (Video Graphics Array) durch, der dem EGA deutlich überlegen ist. Die Farbzahl wurde auf 262'144 erhöht, wobei bei einer Auflösung von 640x480 Punkten (viele Firmen bieten Extras wie etwa 800x600 Punkte) sechzehn Farben gleichzeitig greifbar sind. Bei der Minimalauflösung von 320x200 Punkten sind es immerhin 256 Farben aus der umfangreichen Farbpalette.

Leopold Asböck

Da einige Firmen (Paradise, Cirrus Logic) Grossschaltkreise bereits in Serienproduktion haben, darf in diesem Jahr mit einem Grossangriff der VGA-Platinen gerechnet werden. Dabei werden MDA, CGA, EGA, Hercules, VGA in einen oder zwei «Riesenarrays» gepresst, sodass die Hardwarespezialisten geringe Entwicklungszeiten und -kosten kalkulieren müssen. Ein Boom an VGA-Karten ist aus Taiwan zu erwarten. Anfangs ist noch ein spürbares Zuwarten merkbar, da die Kompatibiltätsfrage noch nicht restlos geklärt ist. IBM gewährt keine Einblicke, das Resultat ist zwar die erwünschte Zeitverzögerung, aber auch ein höheres Leistungsniveau als das der Original-VGA-Karte.

Was mich zu zwei Testprogrammen veranlasst hat, ist die Tatsache, dass in verschiedenen Zeitschriften die VGA hochgelobt wird, nahezu kein Wort aber über den Zugriff auf die 262'144 Farben verloren wird. Manche Testberichte begnügen sich mit Prospekt- und Manualreproduktionen. Es ist sehr erstaunlich, dass selbst Produzenten von leistungsfähigen VGA-Karten in ihren Manuals *kein* Wörtchen über die Programmierung der CLUT (Color Look Up Table) verlieren. Dies gilt auch für die Video Seven VEGA VGA, eine sauber gebaute Karte, die im Manual ihre 262'144 Farben einfach «vergessen» hat.

Die beiden vorgestellten Programme VGATEST1.BAS und VGATEST2.BAS sollen die Möglichkeit bieten, den wundervollen Farbenzauber dieser (oder anderer Karten) hervorzuholen. Dies geht bis zur Darstellung aller Farben der Farbpalette bei einer Auflösung von 800x600 Punkten.

Voraussetzung für die beiden Programme sind eine VGA-Karte (Video Seven VEGA VGA oder andere) und selbstverständlich ein Mehrfrequenzmonitor mit Analogeingang (EIZO 8060S Flexscan, NEC, Mitsubishi, usw.).

Als Programmiersprache wurde Turbo BASIC gewählt, die Bildschirmgrafik bis 640x480 Punkte unterstützt. Für höhere Auflösung lässt sich über BASIC die VGA direkt programmieren. Dabei wurden zwei Methoden verwendet: entweder man «pokt» die Bytes in den Speicher der VGA (vier parallele Speicherebenen) oder man schiebt (als INLINE-Code) eine Maschinensprachroutine ein. Die letztere Methode läuft einige hundert Male schneller ab.

VGATEST1.BAS

Das erste Testprogramm ist in zwei Module unterteilt: der erste zeichnet auf den Bildschirm 16 Rechtecke sowie 16 Kreissektoren, die verschieden gefärbt sind. Zudem werden durch Pixelmischung in einem Rechteck ausser den 16 Grundfarben noch 120 Mischfarben dargestellt.

Jede der 262'144 wählbaren Farben setzt sich aus 64 möglichen Farbwerten für Rot/Grün/Blau zusammen. Die Werte werden angezeigt und können beliebig variiert werden. Gleichzeitig findet die Farbänderung auf dem Bildschirm statt.

```
                        ;       ****************************************
                        ;       INLINE-Programm für 256 Farben (Mode 13)
                        ;       ****************************************
                        ;       Diese Routine schreibt 256 verschieden-
                        ;       farbige Rechtecke auf den Bildschirm

0000                    CSEG    SEGMENT PARA PUBLIC 'CODE'
                                ASSUME CS:CSEG

; --------- Anfang des INLINE-Codes --------------

0000  50                START:  push    ax          ; Registerwerte
0001  51                        push    cx          ; zwischenspeichern
0002  1E                        push    ds
0003  57                        push    di

0004  B8 0013                   mov     ax, 0013h ; Mode 13 aufrufen
0007  CD 10                     int     10h

0009  B8 A000                   mov     ax, 0A000h; Zeiger auf A000:0000
000C  8E D8                     mov     ds, ax
000E  BF 0000                   mov     di, 0000h
0011  B0 00                     mov     al, 00h

0013  B9 0004                   mov     cx, 4       ; Anzahl der Streifen
0016  51                SCH1:   push    cx
0017  B9 0018                   mov     cx, 24      ; Höhe der Rechtecke
001A  51                SCH2:   push    cx
001B  B9 0040                   mov     cx, 64      ; Anzahl der Rechtecke
001E  51                SCH3:   push    cx
001F  B9 0005                   mov     cx, 5       ; Breite eines Rechtecks
0022  88 05             SCH4:   mov     [ds:di], al
0024  47                        inc     di          ; nächstes Pixel
0025  E2 FB                     loop    SCH4
0027  FE C0                     inc     al          ; nächster Pixelwert
0029  59                        pop     cx
002A  E2 F2                     loop    SCH3
002C  2C 40                     sub     al,64       ; Pixelwert - 64
002E  59                        pop     cx
002F  E2 E9                     loop    SCH2
0031  81 C7 0500                add     di, 320*4 ; 4 Leerzeilen addieren
0035  04 40                     add     al, 64
0037  59                        pop     cx
0038  E2 DC                     loop    SCH1

003A  5F                        pop     di          ; Registerwerte
003B  1F                        pop     ds          ; zurückholen
003C  59                        pop     cx
003D  58                        pop     ax

; --------- Ende des INLINE-Codes ----------------

003E  B8 4C00                   mov     ax,4c00h
0041  CD 21                     int     21h

0043                    CSEG    ENDS
```

Listing 1

Mit der SPACE-Taste lassen sich die Farbwerte 1 bis 14 anwählen, ein Balken unterstreicht das Farbwerttripel, das aktiviert ist. Die Werte 0 und 15 (Schwarz 0/0/0 und Weiss 63/63/63) wurden «eingefroren», um die Farben des Hintergrundes und der Schrift nicht zu ändern.

Mit den Tasten 1,2,3,4,5,6 können die Farbenuancen von Rot/Grün/Blau im Bereich von 0 bis 63 inkrementiert oder dekrementiert werden. Mit der Plus-/Minus-Taste lassen sich die Farbwerte gleichschalten, sodass sich 64 Graustufen ergeben.

Besondere Beachtung sollte man den Tasten S und I schenken. S bewirkt folgendes: für die Farben 1 und 14 werden per Zufall je eine Farbe aus den 262'144 möglichen ausgewählt. Die Farbwerte für Farbe 2 bis 13 werden nun interpoliert, sodass sich Farbübergänge zwischen den beiden Randfarben ergeben. Diese werden natürlich auf dem Bildschirm dargestellt.

Mit der Taste I kann man einen Interpolationsvorgang wählen: man stellt die Farbe 1 und die Farbe 14 mit SPACE,1,2,3,4,5,6 nach Wunsch ein, bei Druck auf I werden die Farben 2 bis 13 zwischen den beiden vorgewählten Farben interpoliert.

Jederzeit können die Farbpalettenwerte, bestehend aus sechzehn mal drei Farbzahlen mit D ausgedruckt werden.

Im zweiten Modul des ersten Testprogramms werden 256 Farben aus der vollständigen Palette demonstriert. Dazu wird mit Hilfe von zwei INLINE-Routinen ein Muster aus 256 verschiedenen Werten in den VGA-Speicher geschrieben, anschliessend wird die Palette variiert. Auf Tastendruck (SPACE-Taste) werden fliessende Uebergänge aus 64 Grauwerten, 64 Farbwerten oder 192 Farbwerten gezeigt.

Eine INLINE-Routine (Listing 1) erzeugt Rechtecke mit 256 Farben (4 Zeilen zu 64 Rechtecken), die INLINE-Routine in Listing 2 erzeugt eine Folge von 256 Farbpunkten, die zeilenweise verschoben sind. Durch Palettenwertänderungen ergeben sich optisch ansprechende Farbänderungen und -bewegungen.

VGATEST2.BAS

Das zweite Testprogramm zeigt 262'144 Farben an Hand von Grafikdarstellungen in Auflösungen von 640x350, 640x480 und 800x600 Punkten. Dazu wird bei einer Auflösung von 640x350 Punkten eine Grafik in ei-

```
                        ;       ****************************************
                        ;       INLINE-Programm für 256 Farben (Mode 13)
                        ;       ****************************************
                        ;       schreibt Pixelwerte 0-255 in den VGA-Speicher
                        ;       pro Zeile um ein Pixel versetzt

0000                    CSEG        SEGMENT PARA PUBLIC 'CODE'
                                    ASSUME CS:CSEG

; ----------- Anfang des INLINE-Codes ----------

0000  50                START:      push    ax          ; Registerwerte
0001  51                            push    cx          ; zwischenspeichern
0002  1E                            push    ds
0003  57                            push    di

0004  B8 0013                       mov     ax, 0013h ; Mode 13 aufrufen
0007  CD 10                         int     10h

0009  B8 A000                       mov     ax, 0A000h; Zeiger auf A000:0000
000C  8E D8                         mov     ds, ax
000E  BF 0000                       mov     di, 0000h
0011  B0 00                         mov     al, 00h

0013  B9 00B4                       mov     cx, 180   ; 180 Zeilen
0016  51                SCH1:       push    cx
0017  B9 013F                       mov     cx, 319   ; 319 Pixel schreiben
001A  88 05             SCH2:       mov     [ds:di], al
001C  FE C0                         inc     al          ; Pixelwert erhöhen
001E  47                            inc     di          ; Pixeladresse erhöhen
001F  E2 F9                         loop    SCH2
0021  2C 3F                         sub     al,319-256; Pixelwert modulo 256
0023  59                            pop     cx
0024  E2 F0                         loop    SCH1

0026  5F                            pop     di        ; Registerwerte
0027  1F                            pop     ds        ; zurückholen
0028  59                            pop     cx
0029  58                            pop     ax

; ----------- Ende des INLINE-Codes ------------

002A  B8 4C00                       mov     ax,4c00h
002D  CD 21                         int     21h

002F                    CSEG        ENDS

                                    END
```

Listing 2

Programm 1

```
' ---------------------------------------
'  ***** Testprogramm für VGA-Karte *****
'  640x480 -  16 Farben aus 262144 Farben
'  320x200 - 256 Farben aus 262144 Farben
' ---------------------------------------
'  Leopold Asböck               15.2.1988
' ---------------------------------------
'  geschrieben in Turbo-BASIC
'  getestet auf Video Seven VEGA VGA und EIZO 8060S Flexscan

Start:
screen 9: print: print
print "     V G A  -  T e s t    16/256 Farben aus 262144"
print "     ----------------------------------------------"
print "     Dieses Programm bedingt eine  V G A  mit CLUT"
print "     (Color Look Up Table mit 262144 Farben)"
print "     und einen  A n a l o g m o n i t o r  !"
print "     ----------------------------------------------"
print
print "        E ... Ende"
print
print "        1 ... Mode 12 (640x480 Punkte,  16 Farben aus 262144)"
print
print "        2 ... Mode 13 (320x200 Punkte, 256 Farben aus 262144)";

AX=1: BX=2: CX=3: DX=4: VIDEO=&H10
dim Nummer(255)
randomize timer

Auswahl:
  t$=ucase$(inkey$): if t$="" goto Auswahl
  if t$="E" then goto Fertig
  if t$="1" then goto Mode12
  if t$="2" then goto Mode13
  goto Auswahl

' *********
   Mode12:
' *********

  ersteZeile = 6: zweiteZeile = 13
  breit = 70: hoch = 60
  a = 3: b = 10: d = b+112
```

```
  restore Grundfarben
  for i=0 to 15
    read Nummer(i)
  next i
  Grundfarben:
  data 0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63

  screen 12

  Zeichnen von 16 Rechtecken / Kreissektoren
  ------------------------------------------
  Pi=3.14159: Sektor = 2*Pi/16
  r = 90: rr = r-10: mx = 100: my = 380
  line (100,380) - step (90*cos(0),90*sin(0)),15

  for i=0 to 15
    if i=8 then b=d
    if i=0 then line (a+80*(i and 7),b) - step (breit,hoch), 15, b else_
                line (a+80*(i and 7),b) - step (breit,hoch), i, bf
    circle (mx,my), r, 15, -i*Sektor, -(i+1)*Sektor
    paint (mx+rr*cos((i+0.5)*Sektor), my-rr*sin((i+0.5)*Sektor)),i,15
    next i


' 120 Mischfarben
' ---------------
  locate 15,30: print "Geduld ...";

  xr=240: yr=300: hr=160
  line (xr-1,yr-1)-step(16*24+2,hr+2),15,b

  for Farbe=0 to 15
    line (xr+2*12*Farbe,yr)-step(24,hr),Farbe,bf
  next Farbe

  for Farbe=0 to 15
    for j=0 to 9
      if j/2=int(j/2) then k1=0 else k1=1
      for k=k1 to 24*Farbe step 2
        pset (xr+k,yr+10*Farbe+j),Farbe
      next k
    next j
  next Farbe

' Farbwerte anzeigen
' ------------------
gosub Menu
for Farbe=0 to 15
  y=1: x=10*Farbe+2
  if Farbe>7 then y=8: x=10*(Farbe-8)+2
  locate y,x: print Farbe;
  gosub HolFarben
  gosub SetzFarben
next Farbe
Farbe=1
gosub Unterstreichen


' Abfrage
' -------
Taste:
t$=ucase$(inkey$): if t$="" then goto Taste
if t$=" " then Farbe = (Farbe+1) and &H0F: gosub Unterstreichen: goto Taste
if t$="+" or t$="-" then gosub Grau
if t$="S" then gosub Stufen
if t$="I" then gosub Interpolieren
if t$="D" then gosub Ausdruck
if t$="Q" then goto FertigQ
if t$<"1" or t$>"6" then goto Taste
t=val(t$)
  if t=1 then rot% = (rot%+1) and &H3F
  if t=2 then rot% = (rot%-1) and &H3F
  if t=3 then gru% = (gru%+1) and &H3F
  if t=4 then gru% = (gru%-1) and &H3F
  if t=5 then bla% = (bla%+1) and &H3F
  if t=6 then bla% = (bla%-1) and &H3F
gosub SetzFarben
goto Taste

' aktuelle Farbwerte markieren
' ----------------------------
Unterstreichen:
  alteFarbe = (Farbe-1) and &H0F
  y = ersteZeile+1: x = 10*alteFarbe+2
  if alteFarbe>7 then y = zweiteZeile+1: x = 10*(alteFarbe-8)+2
  locate y,x: print space$(8);
  y = ersteZeile+1: x = 10*Farbe+2
  if Farbe>7 then y = zweiteZeile+1: x=10*(Farbe-8)+2
  locate y,x: print string$(8,chr$(223));
  gosub HolFarben
  gosub SetzFarben
  if Farbe=15 or Farbe=0 then Farbe=(Farbe+1) and &H0F: goto Unterstreichen
return
```

nem Bildfeld von 400x300 Punkten gezeichnet und per Menüwahl in den drei Auflösungsmodi vierfach dargestellt. Mit der Taste S (Stufen) werden die Grafiken in fliessenden Farbübergängen dargestellt, jeweils 15 Farben aus 262'144. Die sechzehnte Farbe (Schwarz) bleibt dem Hintergrund vorbehalten.

Zwei Grafiken stehen zur Auswahl. Es handelt sich um Pseudofiguren, denen jeweils eine Lissajoussche Figur zugrunde liegt: ein Band aus Farbstreifen sowie eine Folge von Kugeln, die zonenweise gefärbt sind. Die Zeichnung der Grafik nimmt abhängig von der Schrittweite mehr oder weniger Zeit in Anspruch.

Es besteht zur Weiterverarbeitung die Möglichkeit, die Grafik auf Diskette zu speichern. Zum Rücklesen wäre eine INLINE-Routine notwendig, da ein «Gepoke» zeitaufwendig ist. Im Programm wurde der Weg gewählt, die Grafik in einem Ganzzahlarray abzuspeichern (60'004 Bytes), sodass sie bei einer Auflösung von 640x350 oder 640x480 Punkten über Turbo BASIC leicht rückgeschrieben werden kann. Da Turbo BASIC die Auflösung von 800x600 Punkten nicht unterstützt, wird in diesem Fall das Ganzzahlarray in den VGA-Speicher gepokt. Dazu muss dies in jede der vier Farbebenen geschehen, ausserdem wird die 400x300-Punkte-Grafik zusätzlich vervierfacht, um 800x600 Punkte zu füllen. Dies dauert natürlich einige Minuten. Anschliessend können mit der Taste S beliebige Farbübergänge aus den 262'144 Farben erzeugt werden. Dies ist eine Demonstration der Leistungsfähigkeit, welche die Video Seven VEGA VGA oder eine ähnliche VGA-Karte erbringt.

Ausbau der Programme

Die beiden Programme können leicht ausgebaut werden: es ist sehr einfach, Bildschirmgrafiken von einer EGA- oder VGA-Karte abzuspeichern. Bei einer Auflösung von 320x200 Punkten und 256 Farben muss man wissen, dass nur eine Farbebene verwendet wird, jedem Farbpunkt entsprechen acht Bits, also ein Byte, im Speicher liegen diese 320x200 Bytes direkt nacheinander ab A000:0000.

Bei einer EGA-/VGA-Grafik mit 640x350 oder 640x480 oder 800x600 Punkten ist jeder Farbpunkt durch vier Bits bestimmt, die in vier Parallelebenen liegen, die über Portbefehle zum Schreiben oder Lesen angesprochen werden müssen.

In der nächsten M+K-Ausgabe finden Sie Programme, die es gestatten, beliebige EGA-/VGA-Grafiken auf Diskette abzuspeichern und von Diskette zu laden. Es genügen dazu ein wenig EGA-/VGA-Knowhow und die Befehle BSAVE und BLOAD. Selbstverständlich werden die zum Bild gehörigen 16/256 Farben aus der 262'144-Palette mitgespeichert. Diese Routinen können Sie in Ihre Programme einbauen - kein Problem mit Turbo BASIC oder Turbo Pascal.

Eine Anmerkung zum Schluss: beachten Sie den Testbericht über die Video Seven VEGA VGA in dieser Ausgabe! □

```
' Graustufen
' ----------
Grau:
  if t$="-" then Eins = -1 else Eins = 1
  rot% = (rot%+Eins) and &H3F
  gru% = rot%: bla% = rot%
  gosub SetzFarben
return

' Menu
' ----
Menu:
  y=15: x=1
  locate y,x:      print  "Farben für VGA 640x480";
  locate y+1,x:    print  "16 Farben aus 262144";
  locate y+2,x:    print  "----------------------";
  y=15: x=30
  locate y,x:      print  "1/2..Rot  3/4..Grün  5/6..Blau  +/-..Graustufen";
  locate y+1,x:    print  "SPACE..nächste Farbe   S..Stufen  D..Ausdruck";
  locate y+2,x:    print  "I..Interpolieren 1-14             Q..Ende";
  locate y+4,x+2:  print " 16 Grund- und 120 Mischfarben 1:1 ";
return

' Farbabstufung
' -------------
Stufen:
  ff=Farbe
  Farbe=1:  rot%=rnd*63: gru%=rnd*63: bla%=rnd*63
  gosub SetzFarben
  Farbe=14: rot%=rnd*63: gru%=rnd*63: bla%=rnd63
  gosub SetzFarben
  gosub Interpolieren
  Farbe=ff: gosub HolFarben
  return

Interpolieren:
  f=Farbe
  Farbe=1: gosub HolFarben
  r1=rot%: g1=gru%: b1=bla%
  Farbe=14:gosub Holfarben
  r2=rot%: g2=gru%: b2=bla%
  r=(r2-r1)/13: g=(g2-g1)/13: b=(b2-b1)/13

  for Farbe=2 to 13
    rot%=int(r1+(Farbe-1)*r+0.5)
    gru%=int(g1+(Farbe-1)*g+0.5)
    bla%=int(b1+(Farbe-1)*b+0.5)
    gosub SetzFarben
  next Farbe
  Farbe=f: gosub HolFarben
  return


' Ausdruck der Farbwerte
' ----------------------
'Ausdruck:
  for d=0 to 3
    locate 15+d,30
    print space$(50);
  next d
  locate 15,30
  print "Drucker einschalten!  ---  SPACE drücken!";
  locate 17,30
  print "(Abbruch mit E)";

  Druckabfrage:
    e$=ucase$(inkey$): if e$="" then goto Druckabfrage
    if e$=" " goto Druckstart
    if e$="E" goto Druckfertig
    goto Druckabfrage



Druckstart:
  f=Farbe
  lprint
  lprint "VGA-Farben 640x480 - 16 aus 262144"
  lprint "-----------------------------------"
  lprint " (";date$;"  ";left$(time$,5);")"
  lprint
  lprint "Rot(DH) / Grün(CH) / Blau(CL)"
  lprint
  lprint tab(16); "R    G    B    FarbNr."
  lprint
  for Farbe = 0 to 15
    gosub HolFarben
    lprint "Farbe";
    lprint using "###"; Farbe;
    lprint "    ";
    lprint using "#####"; rot%; gru%; bla%;
    ro=rot%: gr=gru%: bl=bla%
    lprint using "##########"; 4096*ro+64*gr+bl
  next Farbe
  lprint: lprint
```

COMPUTER-SPLITTER

Back-up-Kassette für PCs mit grossen Festplatten

Der Trend bei Personal Computern geht zur Ausrüstung mit Festplatten-Laufwerken. Die damit entstehenden Back-up-Probleme bei hohen Speicherkapazitäten lassen sich durch externe Bandlaufwerke - sogenannte Streamer - elegant lösen. Neue Bandkassetten, wie die BASF Date Cartridge DC 600 HS, können bis zu 320 MB Daten speichern. Das ist mindestens das 200fache Speichervermögen einer 5.25-Zoll-High-Density-Diskette. Zur Datensicherung muss der Besitzer eines Festplatten-Laufwerkes also nicht mehr «Disc-Jockey» spielen. Er kann mit einem Befehl eine komplette Sicherungskopie anlegen.

Insgesamt sechs verschiedene Data Cartridges mit Speicherkapazitäten zwischen 2,9 und 320 MB nimmt die BASF-Datentechnik ab sofort in ihr Sortiment auf. Eingesetzt werden

sie beispielsweise auf Laufwerken von Tandberg, Cipher, Irwin und Tallgrass. BASF-Untersuchungen nach beträgt die Grösse des Marktes für solche Kassetten in Europa 1988 etwa sechs Millionen Stück.

Technologisch betrachtet, ist die BASF Data Cartridge DC 600 HS eine Spitzenleistung. Auf ca. 180 Metern hochkoerzitivem 0.25-Zoll-Magnetband werden die 320 Millionen Bytes in 26 parallel zur Bandrichtung verlaufenden Spuren serpentinartig mit 20'000 Flusswechseln pro Inch (FTPI) aufgenommen. Dabei ist sie nur 10x15 cm gross, also kleiner als beispielsweise eine VHS-Videokassette. Sie hat aber im Vergleich dazu wegen des Antriebsmechanismus ein aufwendigeres Innenleben. Die DC 2000 HS ist sogar mit 8x6 cm nur so gross wie zwei Streichholzschachteln, bringt es aber immerhin je nach Laufwerk auf 40 bis 60 MB.

Die BASF legt bei ihrer neuen Cartridge besonders grossen Wert auf die technische Abstimmung zwischen Magnetband, Antriebsriemen und Umlenkrollen. Die Schwankungen des Bandzuges zwischen Bandanfang und Bandende werden damit minimiert. Dies erhöht wesentlich die Zuverlässigkeit und die Lebensdauer. Alle BASF Data Cartridges sind vor dem Ausliefern auf 100prozentige Fehlerfreiheit geprüft. Die Lebensdauer beträgt mehr als 5'000 Durchläufe. Die Magnetbänder selbst haben eine neuartige, besonders widerstandsfähige Beschichtung mit 550 Oersted Koerzitivkraft. Diese gewährleistet niedrige Drop-out-Raten und minimalen Abschliff des Schreib/Lesekopfes. Die Betriebstemperatur liegt zwischen 5°C und 45°C bei einer relativen Luftfeuchte von 20 bis 80 Prozent.

Im Unterschied zu anderen Anbietern liefert die BASF ohne Aufpreis spezielle Schuber zum Aufbewahren. Sie bieten einen zusätzlichen Schutz und bessere Archivierbarkeit. Info: BASF (Schweiz) AG, Appital, 8820 Wädenswil/Au, Tel. 01/783'91'11. □

PC/AT-Harddisk mit hoher Kapazität

Mit dem Festplatten-Einbausatz, bestehend aus dem Winchester-Laufwerk Micropolis 1355 und dem Controller Adaptec ACB 2322 lassen sich AT-kompatible PCs (286 und 386) punkto Kapazität und Geschwindigkeit stark aufwerten.

Der 5.25 Zoll grosse Micropolis 1355 Hochleistungswinchester hat eine ESDI-Schnittstelle und ist mit einer Datenrate von 10 MB/sek. und einem

```
  Druckfertig:
  Farbe=f
  gosub HolFarben
  gosub Menu
return


SetzFarben:
  if Farbe=0  then rot%=0:  gru%=0:  bla%=0
  if Farbe=15 then rot%=63: gru%=63: bla%=63
  reg BX,Nummer(Farbe)
  reg DX, rot%*256
  reg CX, gru%*256 + bla%
  reg AX, &H1010
  call interrupt VIDEO

  y = ersteZeile: x = 10*Farbe+2
  if Farbe>7 then y=zweiteZeile: x =10*(Farbe-8)+2
  locate y,x
  print using "##"; rot%;
  locate y,x+3
  print using "##"; gru%;
  locate y,x+6
  print using "##"; bla%;
return

HolFarben:
  reg BX,Nummer(Farbe)
  reg AX,&H1015
  call interrupt VIDEO
  rot%=int(reg(DX)/256)  : rem Wert von DH
  gru%=int(reg(CX)/256)  : rem Wert von CH
  bla%=reg(CX) and &H00FF: rem Wert von CL
return

FertigQ:
  screen 9: goto Start

Fertig:
  screen 9: end


' *********
   Mode13:
' *********
  screen 7
  def seg = &HA000


' Bildschirmspeicher 320x200 ab A000:0000

  Fall=(Fall+1) and 1
  if Fall=1 then call Make256Schraeg else call Make256Farben: locate 23,1:_
                        print "VGA-Demo für 256 Farben aus 262144"
  locate 25,1
  print "Demo         SPACE...weiter    Q...Ende";

DemoStart:
  gosub Pause
  gosub Pause
  t$=ucase$(inkey$)
  if t$="Q" goto FertigQ

'--- Demo 1 ----------------------------------------
  locate 25,6: print "1";
  if Fall=1 then f1=1 else f1=64
  for Farbe=f1 to 255
    rot%=Farbe and &H3F
    if Farbe<128 then rot%=127-rot%
    gru%=rot%: bla%=rot%
    gosub Setz256Farben
  next Farbe


Demo1:
  gosub Pause
  zz=rnd
  if zz>0.5 then zz=2 else zz=3
  aa=rnd
  if aa>0.5 then aa=0:bb=63:cc=1 else aa=63:bb=0:cc=-1
  dd=rnd
  if dd>0.5 then bla%=rnd*25+10:gru%=rnd*25+10 else_
                 rot%=rnd*40+20:gru%=rnd*30+15

  for Farbe=zz*64+aa to zz*64+bb  step cc
    if dd>0.5 then rot%=Farbe and &H3F else bla%=Farbe and &H3F
    if dd<0.1 then rot%=Farbe and &H3F: gru%=rot%: bla%=rot%
    gosub Setz256Farben
  next Farbe
  gosub Pause

  t$=ucase$(inkey$): if t$="" then goto Demo1
  if t$="Q" goto FertigQ
```

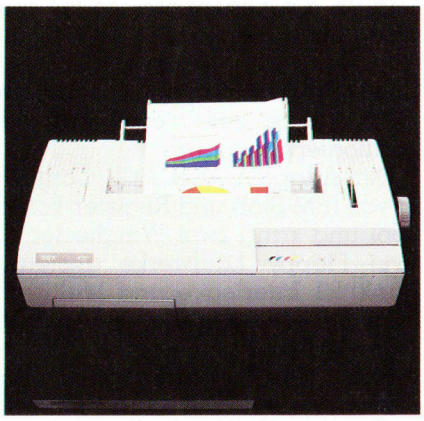
DAS FARBIGE TRIO!

Micro Distribution & Trading Inc.

Xerox Ventura Desktop Publisher, GEM-Palette und Xerox Farb-Tintenstrahldrucker 4020

– Erstellt CAD-Drucke und Kunstgrafiken auf Papier oder Transparentfolien
– Mit Hilfe von 20 Düsen können mehr als 4000 Farbnuancen erstellt werden
– Einzelblatteinzug

Dank dem Selbstdiagnose-System und der Selbstreinigung der Farbdüsen erübrigt sich die Wartung.



Xerox Farb-Tintenstrahldrucker 4020,
GEM-Palette und
Xerox Ventura Publisher
bringen Farbe in Ihren Alltag.

Überzeugen Sie sich bei Ihrem Fachhändler.

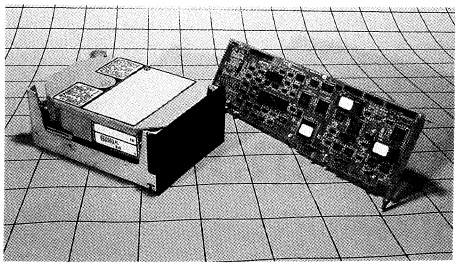
M D & T

Distributor für: Ashton Tate, Digital Research, Micro Pro, Boeing, Xerox Ventura, Tecmar, Scientific Solutions, Quadram, Core Int.

Industriestrasse 404
5242 Birr/Lupfig

Telefon 056/94 01 05
Verkauf 056/94 01 01

Telex 825 146 mdt ch
Fax 056/94 95 65



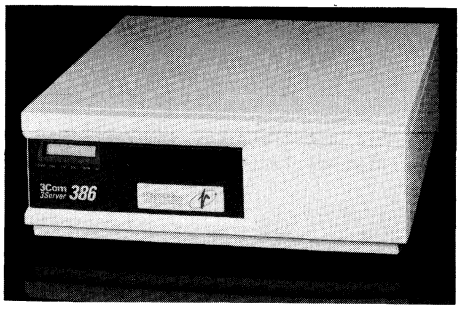
Zugriff von 23 msec wesentlich schneller als herkömmliche Laufwerke. Die formatierte Kapazität beträgt ca. 150 MB.

Die hohe Zuverlässigkeit von 30'000 Stunden MTBF wurde durch zahlreiche innovative technische Lösungen erreicht wie: Lande/Parkzone ausserhalb des Datenbereichs mit automatischem Blockieren des Positionierungsmechanismus; Chassis-in-Chassis Konstruktion; balanced rotary voice coil closed-loop-servo Positionierungsprinzip.

Der Adaptec ACB 2322 Controller ist voll IBM Slot- und Register-kompatibel und kann zwei Winchester und zwei Floppy Laufwerke ansteuern. Dank 1:1 Interleave und Multisector Data Transfer, sind Daten-Druchsatz-Raten von nahezu 1 MB pro Sekunde möglich. Die Software-Kompatibilität erstreckt sich vom DOS 3.X, SCO Xenix bis ISC UNIX 386/v.3. Der eigene BIOS beinhaltet sowohl eine Low-Level Format Routine, wie auch I/O Driver. Info: MST Micro-System-Technik AG, Albisriederstrasse 226, 8049 Zürich, Tel. 01/492'03'55. □

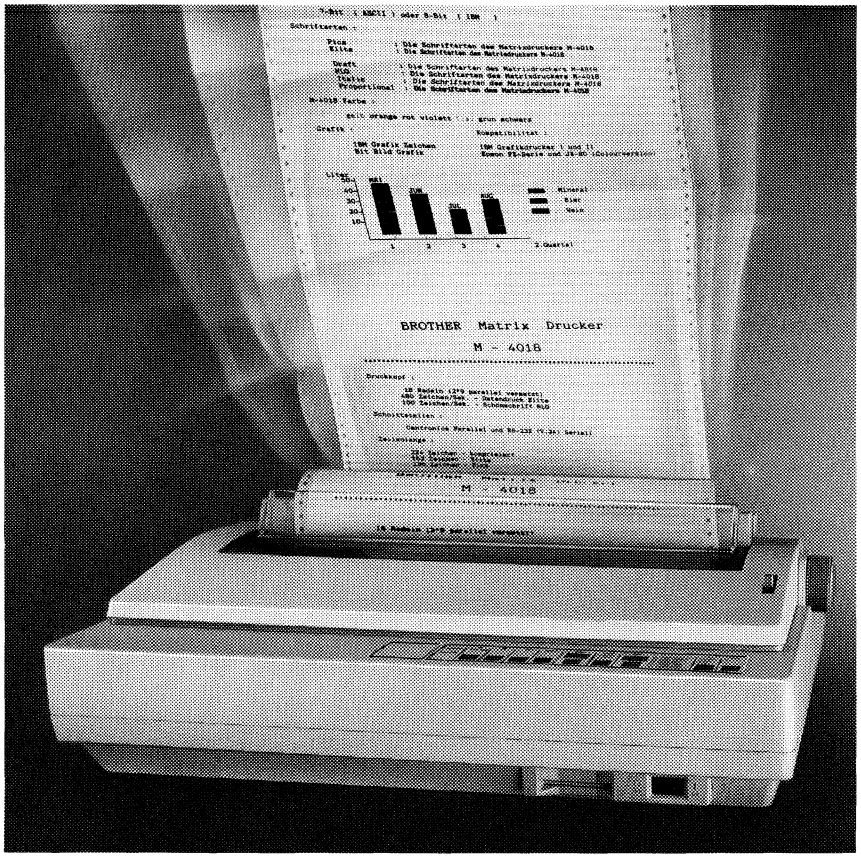
Neuer Netzwerk-Server von 3Com

Der 3S/400-Netzwerk-Server von 3Com ist der leistungsfähigste Server dieser Produkte-Familie. Der 3S/400 kann praktisch in jedem beliebigen Netzwerk installiert werden. Er arbeitet sowohl mit Ethernet als auch mit TokenRing und Apple Talk. Das Modell 3S/401 ist mit einem integrierten, leistungsfähigen Kassetten-Bandgerät ausgestattet, so dass eine automatische Datensicherung im gesamten Netzwerk möglich ist. Eine 150 MB-Festplatte mit extrem schnellem Zugriff erhöht die Leistungsfähigkeit des Servers wesentlich. Durch die offene



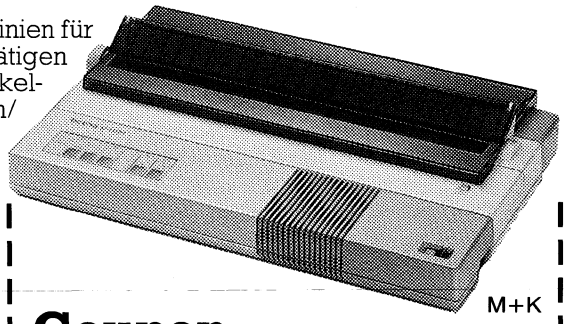

```
'--- Demo 2 ----------------------------------------
Demo2:
  gosub Pause
  locate 25,6: print "2";
  for Farbe=64 to 255
    f=Farbe and &H3F: voll=63
    if Farbe<128  then rot%=voll: gru%=f:    bla%=f
    if Farbe>127  then rot%=f:    gru%=f:    bla%=voll
    if Farbe>191  then rot%=f:    gru%=voll: bla%=f
    gosub Setz256Farben
  next Farbe
  gosub Pause

'--- Demo 3 ----------------------------------------
  locate 25,6: print "3";

  for Farbe=64 to 255
    f=Farbe and &H3F: voll=63
    if Farbe<128  then rot%=f:    gru%=voll: bla%=voll
    if Farbe>127  then rot%=voll: gru%=f:    bla%=voll
    if Farbe>191  then rot%=voll: gru%=voll: bla%=f
    gosub Setz256Farben
  next Farbe
  gosub Pause

  t$=ucase$(inkey$): if t$="" then goto Demo2
  if t$="Q" then goto FertigQ

'--- Demo 4 ----------------------------------------
  r1=rnd*63: g1=rnd*63: b1=rnd*63
Demo4:
  locate 25,6: print "4";
  r2=rnd*63:g2=rnd*63:b2=rnd*63
  r=(r2-r1)/192: g=(g2-g1)/192: b=(b2-b1)/192
  for Farbe=64 to 255
    rot%=r1+(Farbe-64)*r+0.5
    gru%=g1+(Farbe-64)*g+0.5
    bla%=b1+(Farbe-64)*b+0.5
    gosub Setz256Farben
  next Farbe
  gosub Pause

  r3=rnd*63: g3=rnd*63: b3=rnd*63
  r=(r3-r2)/192: g=(g3-g2)/192: b=(b3-b2)/192
  for Farbe=255 to 64 step -1
    rot%=r2+(255-Farbe)*r+0.5
    gru%=g2+(255-Farbe)*g+0.5
    bla%=b2+(255-Farbe)*b+0.5
    gosub Setz256Farben
  next Farbe
  r1=r3: g1=g3: b1=b3
  gosub Pause

  t$=ucase$(inkey$): if t$="" then goto Demo4
  if t$="Q" goto FertigQ

'--- Demo 5 ----------------------------------------
Demo5:
  gosub Pause
  locate 25,6: print "5";

  if Fall=1 then f1=1 else f1=64
  for Farbe=f1 to 255
    rot%=rnd*63: gru%=rnd*63: bla%=rnd*63
    gosub Setz256Farben
  next Farbe
  t$=ucase$(inkey$): if t$="" then goto Demo5
  if t$="Q" goto FertigQ

goto Mode13

'--- Demo Ende -------------------------------------

Pause:
  for pp=1 to 800: next pp
  return

Setz256Farben:
  reg BX, Farbe
  reg DX, rot%*256
  reg CX, gru%*256 + bla%
  reg AX, &H1010
  call interrupt VIDEO
  return

' 256 Farben zeichnen
' -------------------
' WICHTIG - inline-Code richtig abtippen!

sub Make256Farben inline
  $inline  &H50, &H51, &H1E, &H57, &HB8, &H13, &H00, &HCD
  $inline  &H10, &HB8, &H00, &HA0, &H8E, &HD8, &HBF, &H00
  $inline  &H00, &HB0, &H00, &HB9, &H04, &H00, &H51, &HB9
```

Höchstgeschwindigkeit: 480 Zeichen/Sek.



Trotz ungezügeltem Temperament überzeugt der Matrixprinter M-4018 von Brother durch absolute Laufruhe.

Einmal mehr setzt Brother mit innovativer Technologie neue Richtlinien für die Leistungsfähigkeit eines Druckers. Denn brillante Werte bestätigen die Klasse des neuen M-4018: in 3 Stufen beschleunigt er von makelloser Briefqualität bis zur Höchstgeschwindigkeit von 480 Zeichen/Sekunde. Ungewöhnlich einfach, durch leichten Tastendruck, kann die Gestaltung der Schriftstücke variiert werden. Verschiedene Schriften und Sonderdruckarten lassen sich ohne Programmierkenntnisse direkt anwählen. Sieben unterschiedliche Farben bieten zusätzliche Attraktivität beim Drucken. Über all diesen leistungsbezogenen Merkmalen steht beim Brother M-4018 wie auch bei allen anderen Printern der hohe Qualitätsstandard und ein erfolgreiches, bewährtes Konzept. Ob Matrixprinter, Laserprinter, Typenraddrucker oder Twinriter, bei dem gleich zwei Drucksysteme kombiniert wurden. – Ihr Brother-Händler weiss mehr darüber.



M+K

Coupon

Wir wünschen Informationen über:

- ☐ Matrixprinter
- ☐ Laserprinter
- ☐ Typenraddrucker
- ☐ Twinriter (2 in einem)
- ☐ Schreibmaschinen
- ☐ Zubehör

Name:

Firma:

Adresse:

PLZ/Ort:

Bitte einsenden an: Brother Handels AG, 5405 Baden

brother®

Der Zeit voraus.

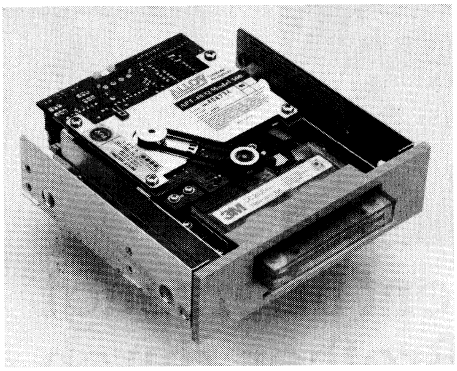
Brother Industries Ltd., Nagoya/Japan, Brother Handels AG, 5405 Baden

BW

Busarchitektur ist es nun auch möglich, AT-kompatible Adapter, Erweiterungs-, sowie Kommunikationskarten einzusetzen.

Prozessor: 80386 mit 16 MHz Taktfrequenz ohne Wartestatus; Triple-Port-32 bit-RAM-Speicher, 82586 Ethernet-Coprozessor, AT-Bus mit geteiltem Speicherzugriff; 2 MB RAM-Arbeitsspeicher als Standard, auf 14 MB aufrüstbar, auf zwei Zusatzplatinen mit 32 bit-Struktur.

Platten- und Bandspeichereinheiten: 150 MB-Festplatte mit 16 msec. Zugriffszeit, integriertem SCSI-Controller und 1:1-Interleave-Faktor; 150 MB 0.25-Zoll-Kassetten-Bandstation (Flachbauweise) zur Datensicherung (bei 3S/401 serienmässig); Erweiterungseinheit für vier zusätzliche Plattenspeicher, der SCSI-Controller unterstützt bis zu sechs Platten- und/oder Bandstationen. Info: Positronika AG, Bösch 61, 6331 Hünenberg, Tel. 042/38'18'28. □



Datensicherung nicht vergessen!

Der ATP-40/Q-Streamer für Personal Computer speichert 40 MB formatierte Daten auf einer kompakten Minikassette der DC 2000-Serie. Eine Standard SA-450/475 Schnittstelle ermöglicht den direkten Anschluss an existierende Floppy-Disc-Controller-Karten, so dass ein Extra-Controller entfällt. Des weiteren wird eine spezielle Applikationssoftware für Backup und Restore-Funktionen mitgeliefert; so unterstützt die ResQNET Software unter anderem aktiven Netzwerk-Backup/Restore von Novell files. Mit dem Produkt wird eine vollständige Dokumentation sowie Diagnostic Software zur Verfügung gestellt. Was eine seriöse Datensicherung wert ist, erfährt man spätestens dann, wenn Massenspeicher mit wichtigen Daten defekt gegangen sind und diese nicht regelmässig gespeichert wurden. Auch hier gilt: Vorbeugen ist besser als Heilen Info: Panatronic AG, Industriestrasse 59, 8152 Glattbrugg, Tel. 01/810'32'10. □

```
  $inline  &H18, &H00, &H51, &HB9, &H40, &H00, &H51, &HB9
  $inline  &H05, &H00, &H88, &H05, &H47, &HE2, &HFB, &HFE
  $inline  &HC0, &H59, &HE2, &HF2, &H2C, &H40, &H59, &HE2
  $inline  &HE9, &H81, &HC7, &H00, &H05, &H04, &H40, &H59
  $inline  &HE2, &HDC, &H5F, &H1F, &H59, &H58
end sub

sub Make256Schraeg inline
  $inline  &H50, &H51, &H1E, &H57, &HB8, &H13, &H00, &HCD
  $inline  &H10, &HB8, &H00, &HA0, &H8E, &HD8, &HBF, &H00
  $inline  &H00, &HB0, &H00, &HB9, &HB4, &H00, &H51, &HB9
  $inline  &H3F, &H01, &H88, &H05, &HFE, &HC0, &H47, &HE2
  $inline  &HF9, &H2C, &H3F, &H59, &HE2, &HF0, &H5F, &H1F
  $inline  &H59, &H58
end sub

' *************************
' *   Ende des Programms  *
' *************************
```

Programm 2

```
' ------------------------------------
'  V G A - Demo  800x600, 262144 Farben
' ------------------------------------
'  Leopold Asböck              15.3.1988
' ------------------------------------
' Hardware - VGA-Karte (Video Seven VEGA VGA)
'            Farbmonitor (800x600), analog
' Software - Turbo-BASIC

cls
print " Dieses Programm bedingt eine VGA-Karte (VIDEO SEVEN VEGA VGA)"
print " mit einer Auflösung von 800x600 Punkten sowie einen"
print " Farbmonitor (Multisync) mit Analogeingang!"
print
print " E ... Programm beEnden"
print
print " k ... Start (Kugel, Schrittweite 1)"
print " K ... Start (Kugel, Schrittweite 5)"
print " b ... Start (Band,  Schrittweite 1)"
print " B ... Start (Band,  Schrittweite 5)"

Start:
t$=inkey$: if t$="" goto Start
Weite=1
if t$="k" then Fall=1: goto Anfangen
if t$="b" then Fall=2: goto Anfangen
Weite=5
if t$="K" then Fall=1: goto Anfangen
if t$="B" then Fall=2: goto Anfangen
if t$="E" or t$="e" then print: print: end
goto Start



Anfangen:

  AX=1: BX=2: CX=3: DX=4: VIDEO=&H10
  Pi=3.14159
  dim Nummer(255)
  for i=0 to 15
    read Nummer(i)
  next i
  data 0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63
  randomize timer

' Bild zeichnen 400x300

screen 9      : ' Format 640x350
dim z%(30001) : ' 4+4*(400*300/8) Bytes = 30002 Ganzzahl-Variable

' Rahmen zeichnen
cls
line (0,0)-(399,299),14,b
  locate 1,24: print "400"
  locate 10,1: print "300"
  for i=1 to 15
    line (370,5+18*i)-step(24,18),i,bf
  next i
  locate 3,55: print "Geduld ... (A..Abbruch)"

' Bild zeichnen

if Fall=2 goto Lissa2

Lissa1:
for i=0 to 1.8*Pi  step 0.0005*Weite
  Farbe=int(i*180) mod 32
```

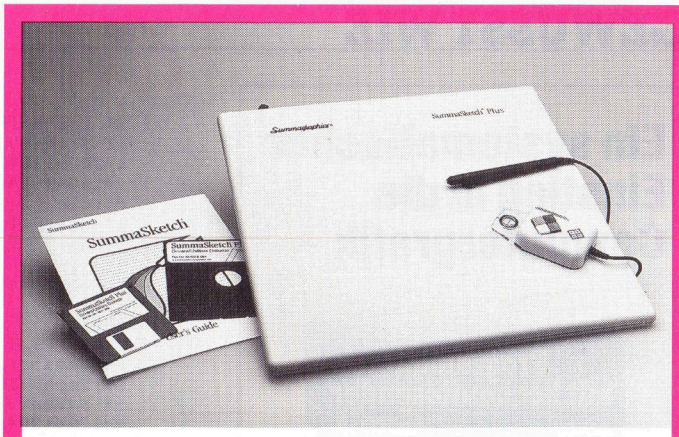
Alles für Ihren PC

	XT	AT
Hercules kompatible Graphics-Printercard	139.—	139.—
EGA Multi. Sync Card 640 × 480	419.90	419.90
EGA Card	362.90	362.90
Harddiskcontroller OMTI	191.—	
Harddisk-Floppycontroller WESTERN-DIGITAL		452.70
2 MB RAM-Karte EMS ohne RAM	265.—	299.—
RAM Card 512 K auf 640 K		144.60
Serial-Parallel Karte		164.30
RS-232 (Serial) Karte	62.50	62.50
Uhr/Datum Karte inkl. Software	69.90	
LOGI-MAUS inkl. Treiber-software	193.50	193.50
I/O Plus Karte Uhr/Par.Ser. Gameport	169.—	
Powersupply 150 resp. 200 Watt	197.40	289.40
3.5" Floppy 720 K (einbaubereit)	380.50	380.50
3.5" Floppy 1,44 MB (einbaubereit)	465.—	465.—
19" EGA-Monitor	3450.—	3450.—

Es hat vieles mehr: Streamer, Harddisk, Software, Disketten, Prozessoren, RAM, Systeme, Modem, Printer, usw.

12 Monate Garantie! Lieferung ab Lager gegen Nachnahme!

Telefon 057/44 47 40



SUMMAGRAPHICS

Die meist verkauften Digitalisiertabletts

sind in allen Grössen, für alle Applikationen und jede Art von Software erhältlich. Wird komplett mit Zubehör geliefert.

Generalvertretung Schweiz

captronix s.a

SYSTEMS · COMPUTERS · PERIPHERALS

1202 **GENÈVE,** RUE DU VALAIS 9, TÉL. 022/32 21 21.
1022 **CHAVANNES**-PRÈS-RENENS, CHEMIN DE LA MOULINE 8, CP 153, TÉL. 021/35 13 42.
8065 **ZÜRICH-GLATTBRUGG,** TALACKERSTRASSE 9, TEL. 01/810 80 00.

SMART EGA™ PLUS

Extended EGA Plus High Resolution and VGA Graphics

Intelligente Farbgrafikkarte für jeden Monitor (auch Monochrom)

Einführungspreis **nur** Fr. 600.–

PERIMATIC AG
Alberich-Zwyssig-Strasse 49, 5430 Wettingen, Telefon 056/27 12 91

Direktimport von Computern und Elektronik

AT 386, AT 286, XT 8088
Speichererweiterungen Graphikkarten,
Harddisk, Floppylaufwerke, Speicherchips

Günstige Preise
Interessant auch für Wiederverkäufer
Verlangen Sie die Preisliste.

SWIF TRADE AG

Software- und Informatik-Fachhandel
4800 Zofingen, Pomerngut, ✆ 062/51 82 00

Software-Post®

DIE HARD-& SOFTWARE-PROFIS

PROGRAMMIERSPRACHEN
TURBO PASCAL 8087 + GS 4.0
TURBO PROLOG
TURBO BASIC
TURBO C
TURBO DATABASE TOOLBOX
TURBO GRAPHIX, EDITOR JE
TURBO TUTOR
MS MACRO ASSEMBLER
MS QUICK C
MS QUICKBASIC COMPILER
MS C COMPILER
MS PASCAL COMPILER
LATTICE C COMPILER
MODULA-2 (M2SDS)

INTEGRIERTE SYSTEME
ACCESS FOUR
ENABLE
FRAMEWORK II
OPEN ACCESS II
LOTUS SYMPHONY

TEXTVERARBEITUNG
WORDPERFECT 4.2
WORDPERFECT LIBRARY
WORDSTAR 2000 PLUS
MS WORD
MULTIMATE (ADVANTAGE)
TEX-ASS WINDOW PLUS

GRAFIKPROGRAMME
DR DRAW
DR GRAPH
GEM DRAW
GEM GRAPH
MS CHART
IN-A-VISION
ENERGRAPHICS 2.0
CLICKART. PER. PUBL.
AUTO CAD

DESKTOP PUBLISHING
PAGEMAKER
ARTEXT dt.
VENTURA
HARVARD PROF. PUBLISHER

Überleg nicht lang, ruf doch an!

TABELLENKALKULATIONEN
LOTUS 1-2-3
SUPERCALC 4
MS MULTIPLAN
PLAN PERFECT

DATENBANKSYSTEME
DBASE III PLUS
CIIPPER (DBASE COMPILER)
REFLEX BORLAND
PARADOX

PROJECT MANAGEMENT
TIMELINE
TOTAL PROJECT MANAGER
MS PROJECT
SUPERPROJECT PLUS

HILFSPROGRAMME / UTILITIES
MS WINDOWS 2.0
HAL
GEM COLLECTION
SIDEWAYS 3.0
COPY II PC
OPTION BOARD
PC TOOLS (neu mit dt. HB)
COPYWRITE mit ZERODISK
NORTON ADVANCED EDITION
NORTON COMMANDER
NORTON EDITORS
DIRECT ACCESS (Obf.Menupr.)
FASTBACK HARDDISK BACKUP
XENOCOPY-PC
DS BACKUP
TURBO LIGHTNING
SUPERKEY
SIDEKICK
ORTHOCHECK I + II
AUTOSKETCH

KOMMUNIKATION
CROSSTALK XVI
CARBON COPY PLUS
REMOTE
SMARTTERM
SOFTTERM
MIRROR

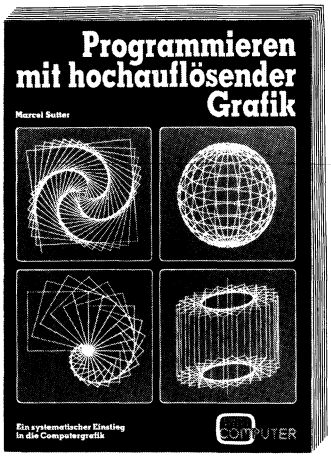
Überleg nicht lang, ruf doch an!

- **Express-Lieferung auf Wunsch** ● **Lieferung mit Rechnung**
- 2% Skonto mit Check und Überweisung im voraus. ● SBG, Grenchen, Konto Nr. 707.504.05 T
- Preis- und Verkaufsprogrammänderungen vorbehalten.
- **Fragen Sie nach unseren Hardwarelösungen ☎**

CH-Worblaufen

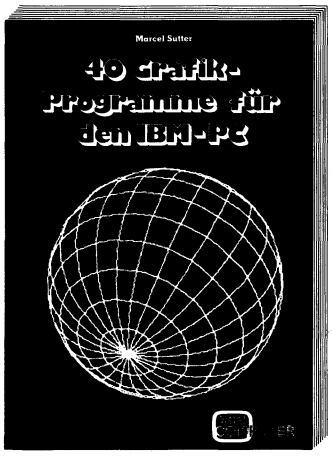
Softwarepost und Versand
Toni Smith GmbH, Solothurnstr. 12
Postfach 1157, CH-2540 Grenchen
Tel. 065 / 53 02 22

Ein systematischer Einstieg in die Computergrafik



Die kurzen BASIC-Programme verwenden als Besonderheit nur zwei, auf jedes Computersystem problemlos anwendbare Grafikbefehle, sind selbsterklärend und können top-down gelesen werden. Bescheidene Kenntnisse in BASIC werden vorausgesetzt. Damit es aber noch einfacher wird, sind in einem Anhang zusätzlich die Listings für den **Commodore C-64/128** und den **ATARI ST** gleich mit abgedruckt.

248 Seiten, Paperback, Fr. 45.–



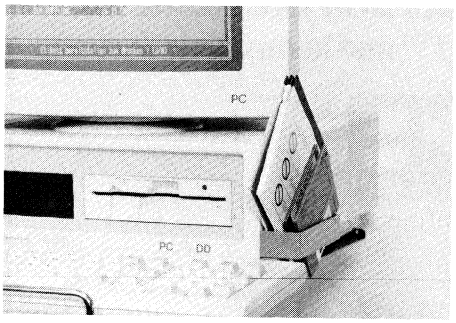
Anhand von 40 Beispielen für den IBM-PC und Kompatible wird der interessierte PC-Anwender Schritt um Schritt in die faszinierende Welt der Computergrafik eingeführt. Dem Leser gelingt es in ganz kurzer Zeit und in ständigem Vergleich von Aufgabe und Lösungsweg mit kurzen Programmlistings, die ihm neben dem beabsichtigten Aha-Effekt sofort ein Erfolgserlebnis vermitteln, das grundlegende Prinzip der Grafikprogrammierung zu erlernen.

168 Seiten, Paperback, Fr. 35.–

Bestellkarte vorne im Heft

```
  if Farbe>15 then Farbe=31-Farbe
  if Farbe=0 then Farbe=1
  x=200+120*sin(i)
  y=150+100*sin(2*i)
  circle (x,y),abs(40*sin(10*i))+2*i,Farbe
  if ucase$(inkey$)="A" goto Speichern
next i

goto Speichern

Lissa2:
for i=0-Pi/4 to 2*Pi-Pi/4  step 0.002*Weite
  Farbe=abs(int(i*100)) mod 32
  if Farbe>15 then Farbe=31-Farbe
  if Farbe=0 then Farbe=1
  x=170+130*sin(3*i): xx=1.2*x
  y=150+120*sin(2*i): yy=0.8*y
  line (x,y)-(xx,yy),Farbe
  if ucase$(inkey$)="A" goto Speichern
next i


' Bild in Ganzzahl-Array speichern (60004 Bytes)

Speichern:
  locate 3,55: print space$(24)
  get (0,0)-(399,299),z%

Wahl123:
  locate 3,60: print "1 .. 640x350"
  locate 4,60: print "2 .. 640x480"
  locate 5,60: print "3 .. 800x600"
  locate 7,60: print "E .. Ende"
  locate 9,55: print "Drücken Sie 1,2,3 oder E";

Fall123:
  t$=ucase$(inkey$): if t$="" goto Fall123
  if t$="1" goto Auswahl1
  if t$="2" goto Auswahl2
  if t$="3" goto Auswahl3
  if t$="E" then screen 12: end
  goto Fall123



Auswahl1:
  Bild$="640x350"
  screen 12: cls
  screen 9
put (0,0),z%
get (0,0)-(239,299),z%
put (400,0),z%
get (0,0)-(400,49),z%
put (0,300),z%
get (0,0)-(239,49),z%
put (400,300),z%
get (0,0)-(399,299),z%
goto Abfrage



Auswahl2:
  Bild$="640x480"
  screen 12: cls
put (0,0),z%
get (0,0)-(239,299),z%
put (400,0),z%
get (0,0)-(400,179),z%
put (0,300),z%
get (0,0)-(239,179),z%
put (400,300),z%
get (0,0)-(399,299),z%
goto Abfrage


Auswahl3:
  Bild$="800x600"

' Aufruf 800x600 für VIDEO SEVEN VEGA VGA
' ----------------------------------------
'       WICHTIG - für andere VGA-Karten ist der
'                 Aufruf entsprechend zu ändern!
  reg AX, &H6F05
  reg BX, &H0062
  call interrupt VIDEO
' ----------------------------------------

locate 22,1
print "800x600 Punkte, 16 aus 262144 Farben"
print
print "Geduld ... 4 Farbebenen schreiben (A..Abbruch)";

  def seg = &HA000
```

COMPUTER-SPLITTER



Disketten griffbereit

Der oft in der Werbung gezeigte aufgeräumte Computer-Arbeitsplatz entspricht wohl selten der Realität. In der Regel häufen sich auf dem Schreibtisch eine Vielzahl von Papieren gut gemischt mit einer Anzahl von Disketten. Da ist es schnell geschehen: Der Anwender zieht unter einem Stapel eine Diskette hervor und erwischt diese gerade an ihrer griffigsten Stelle, der Lese-Schreiböffnung...

Die Firma Lindy versucht in jüngster Zeit mit kleineren Hilfen grösseren Missständen Paroli zu bieten. Für Anwender ohne Festplatte, oder gar mit nur einem Laufwerk ist neuerdings ein unscheinbares Utensil im Sortiment, das genau so einfach wie genial ist: Die «Plonker-Box». Die preiswerte Box wird seitlich mit einem Selbstklebeband am Rechnergehäuse befestigt und kann bis zu acht häufig gebrauchte Disketten aufnehmen. Die Datenträger sind so im wahrsten Sinne des Wortes «griffbereit» und es besteht keine Gefahr mehr, versehentlich in die Lese-Schreiböffnung der Diskette zu fassen. Die Box fasst sowohl 5.25 wie auch 3.5 Zoll-Disketten. Info: LINDY-Elektronik GmbH, Karl-Kuntz-Weg 9, D-6800 Mannheim 25. □

Swiss Computer Graphics Association

Am 14. Juni 1988 findet die achte Jahrestagung der SCGA statt. Zahlreiche Referenten werden dabei über aktuelle Themen aus dem Gabiet der Computergrafik sprechen. Nicht weniger als drei Referate werden von Carl Machover, dem amerikanischen «Computer-Grafik Guru», gehalten. Ein Muss für jeden Computergrafiker. Info: Frau D. Koller, Sekretariat SCGA, Geographisches Institut Universität Zürich-Irchel, Winterthurerstrasse 190, 8057 Zürich, Tel. 01/257'52'57. □

```
out &H3C4,2: out &H3C5,8: Plane=0: gosub ZeichnePlane
out &H3C4,2: out &H3C5,4: Plane=1: gosub ZeichnePlane
out &H3C4,2: out &H3C5,2: Plane=2: gosub ZeichnePlane
out &H3C4,2: out &H3C5,1: Plane=3: gosub ZeichnePlane
out &H3C4,2: out &H3C5,&H0F
if tt$="A" then t$="Q": goto Taste1

Abfrage:
  locate 1,1
  print Bild$; "  S..Stufen  Q..Quit "

Taste:
  t$=ucase$(inkey$): if t$="" goto Taste
  if t$="S" then gosub Stufen
Taste1:
  if t$="Q" then screen 12: cls: screen 9: put (0,0),z%: goto Wahl123
  goto Taste

ZeichnePlane:

for i=0 to 299
  for j=0 to 24
    a%=z%(2+25*Plane+100*i+j)
    a$=mki$(a%)
    b1=asc(left$(a$,1))
    b2=asc(right$(a$,1))
    ii=100*i+2*j
 poke ii,    b1
 poke ii+1,  b2
 poke ii+50, b1
 poke ii+51, b2
 poke ii+30000, b1
 poke ii+30001, b2
 poke ii+30050, b1
 poke ii+30051, b2
    if tt$="A" goto Abbruch
    t$=ucase$(inkey$): if t$="A" then tt$=t$: goto Abbruch
  next j
next i

  Abbruch:
 return


' Farbabstufung
' -------------
Stufen:
  Farbe=1
  rot%=rnd*63: bla%=rnd*63: gru%=rnd*63
  gosub SetzFarben
  Farbe=15
  rot%=rnd*63: bla%=rnd*63: gru%=rnd63
  gosub SetzFarben
  gosub Interpolieren
  return

Interpolieren:
  Farbe=1: gosub HolFarben
  r1=rot%: b1=bla%: g1=gru%
  Farbe=15:gosub Holfarben
  r2=rot%: b2=bla%: g2=gru%
  r=(r2-r1)/13: g=(g2-g1)/13: b=(b2-b1)/13

  for Farbe=2 to 14
    rot%=int(r1+(Farbe-1)*r+0.5)
    bla%=int(b1+(Farbe-1)*b+0.5)
    gru%=int(g1+(Farbe-1)*g+0.5)
    gosub SetzFarben
  next Farbe
  return

SetzFarben:
  if Farbe=0  then rot%=0:  gru%=0:  bla%=0
  reg BX, Nummer(Farbe)
  reg DX, rot%*256
  reg CX, gru%*256 + bla%
  reg AX, &H1010
  call interrupt VIDEO
return

HolFarben:
  reg BX, Nummer(Farbe)
  reg AX, &H1015
  call interrupt VIDEO
  rot%=int(reg(DX)/256)
  gru%=int(reg(CX)/256)
  bla%=reg(CX) and &H00FF
return

' *** Ende des Programms ***
```

Das Programm START

Die Zugriffsgeschwindigkeit und die Aufnahmekapazität der Datenträger nimmt ständig zu. Dem PC-Benützer stehen immer mehr Programme zur Verfügung, die er auch täglich anwendet. Während man sich früher damit abgab, ständig Disketten zu wechseln um das neue Programm zu laden, hat man es heute bequemer, indem man eine sehr grosse Anzahl von Programmen gleichzeitig, z.B. auf der Festplatte, speichern kann. Um die Uebersicht über den Inhalt eines Datenträgers nicht zu verlieren, werden Verzeichnisse gebildet und hier nach irgendeinem Ordnungsprinzip Dateien und Programme abgelegt. Zwei Probleme ergeben sich hier: Das Auffinden von Programmen und Dateien und das Aufrufen von Programmen. Mit Hilfe des in M+K 88-1 vorgestellten Programmes WO kann man schnell und sicher das Gewünschte finden.

Dr. Herbert Steiner

Wenn man ein Programm abruft, verlangt DOS, dass man den Pfad des Programmes angibt. Befindet sich das Programm LIES im Unterverzeichnis LIES des Verzeichnisses MICRO, müsste man folgenden Befehl eingeben, um das Programm abzurufen:

\MICRO\LIES\LIES MEIN.TXT

Das heisst, mit der Anzahl der Ebenen, in denen man Verzeichnisse bildet und entsprechend für Ordnung sorgt, wächst die Schreibarbeit beim Abrufen eines Programmes. Mit Hilfe des Befehles PATH kann man DOS mitteilen, wie er bestimmte Programme suchen soll und in welcher Reihenfolge diese Suche gestaltet wird. Dies bedeutet aber wiederum: Jedesmal wenn ein Programm in ein Verzeichnis abgelegt wird, muss man dazu Sorge tragen, dass die entsprechenden Pfadbefehle im Hauptverzeichnis eingegeben werden und vor Abruf des Programmes aktiv sind. Viele PC-Benützer haben sich schon des öfteren überlegt, ob sie einfach die ganze Ordnung weglassen und alles mögliche im Hauptverzeichnis lagern sollen, weil man es dort am einfachsten abrufen kann.

Ein weiteres Problem kommt hinzu: Wenn man sich im Hauptverzeichnis befindet und wie oben dargestellt ein Programm abruft, indem man Pfad- und Programmname angibt, kann man sehr oft lästige Fehlermeldungen vorfinden, z.B. die Meldung, dass das Hauptprogramm irgendwelche Dateien nicht findet, die es benötigt, oder dass das Laden von OVERLAYS (Programmteile, die sich im gleichen Verzeichnis befinden müssen) nicht funktioniert. Dies ist auf folgendes zurückzuführen:

Das Programm wird gemäss Pfadangabe gerufen, das Verzeichnis wird aber nicht gewechselt. Da das Verzeichnis nicht gewechselt wird, werden Dateien, die benötigt werden, nur im aktuellen Verzeichnis gesucht. Da dies meistens das Hauptverzeichnis ist, werden diese auch entsprechend nicht gefunden. Auch hier hilft nur eines: Eine Batch-Datei aufzubauen, die nicht nur das Programm ruft, sondern auch dafür sorgt, dass ein Verzeichniswechsel stattfindet.

Das Dilemma lautet: Entweder Ordnung, um den Datenträger übersichtlicher zu gestalten. Oder alles mögliche im Hauptverzeichnis lagern, um Programme einfach abzurufen und Fehler und Tipparbeit zu vermeiden.

Aus diesem Dilemma hilft das Programm START. Man hat die Möglichkeit, die Festplatte ordentlich und übersichtlich zu gestalten, und gleichzeitig Programme einfach und bequem abzurufen.

Programm-Aufbau

Das Programm erledigt der Reihe nach folgende Tätigkeiten:

1. Die eingegebene Kommandozeile (START + Argumente) wird programmintern übersetzt. Das Programm muss wissen, in welcher Art welches Programm aufgerufen werden soll.
2. Das Programm muss wissen, in welchem Laufwerk und in welchem Verzeichnis man sich bei Beginn des Programmes befindet. Diese Werte werden gesichert, um nach Beendigung des Programmes genau zum Ausgangspunkt zurückzukehren.
3. Das aufgerufene Programm wird gesucht. Bei nicht Auffinden muss ein Fehler gemeldet werden.
4. Das gefundene Programm soll gemäss Kommandozeile gestartet werden.
5. Nach Beendigung des abgerufenen Programmes wird der Ursprungszustand wieder hergestellt. Man soll sich wieder in dem Laufwerk und in dem Verzeichnis befinden, wo man eingangs war.

Einlesen der Kommandozeile

Zuerst soll die gewünschte Syntax für den Aufrufbefehl des Programmes festgelegt werden:

Syntax: START [/V] [LAUFWERK:]Datei [PARS]
/V = Datei ohne Verzeichniswechsel abrufen
[Pars] sind Parameter für die abgerufene Datei

START sorgt im Normalfall bei Aufruf eines Programmes dafür, dass man sich im Laufwerk und Verzeichnis des aufgerufenen Programmes befindet. Dies ist auch die vernünftigste Lösung, da, wie eingangs dargestellt, Programme sehr oft Dateien im gleichen Verzeichnis wo sie sich befinden, suchen.

START gestattet auch, ein Programm so abzurufen, dass kein Verzeichniswechsel vorgenommen wird.

Disketten-Service

Auch das Programm START können Sie, wie auch die bisher vorgestellten Programme WO und LIES, auf Diskette anfordern. Die Diskette enthält eine Datei mit dem Quellcode, der von Ihnen mit jedem Texteditor verändert werden kann, die zugehörige Cross-Referenz-Datei sowie das Object-File. Und, für viele das Wichtigste, eine auf allen IBM-kompatiblen Computern lauffähige Version des Programmes. Dies alles für den geringen Unkostenbeitrag von Fr. 10.-- pro Diskette. Bestellen Sie Ihre Diskette mit der dem Heft am Schluss beigehefteten Karte «Disketten-Service» und vergessen Sie bitte nicht anzugeben, ob Sie eine 5.25- oder 3.5-Zoll-Diskette wünschen. Senden Sie uns kein Geld zum voraus. □

Man befindet sich im Unterverzeichnis PROGRAMM des Verzeichnisses PASCAL. START wurde im Hauptverzeichnis gespeichert. Nun möchte man «MEINPROG.PAS» lesen (mit Hilfe des Programmes LIES, M+K 88-2, ist dies sehr einfach). Normalwerweise verlangt START folgende Schreibweise:

\START LIES\PASCAL\PROGRAMM\MEINPROG.PAS

Dank der Option /V (kein Verzeichniswechsel) ist dies noch einfacher mit folgendem Befehl:

\START /V LIES MEINPROG.PAS

Mit Hilfe dieser Option ist es daher möglich, das Schreibpensum bei Aufruf eines Programmes auf ein Minimum zu beschränken.

START verlangt in der Kommandozeile nur die Eingabe des Programmes (ohne Endung oder Pfadangabe) und des Laufwerkes wo sich das Programm befindet. Dies ist jedoch nur dann notwendig, wenn sich das Programm nicht im aktuellen Laufwerk befindet.

Wenn notwendig, verlangt die Kommandozeile die Argumente (auch Parameter genannt) für das abgerufene Programm. Diese Argumente müssen genau der Schreibweise entsprechen, die das abgerufene Programm verlangt. Logischerweise können diese Argumente, da sie für ein fremdes Programm bestimmt sind, nicht kontrolliert werden.

Die Routine «PARAMETER_LADEN» liest die Kommandozeile. Zuerst wird ermittelt, ob überhaupt Angaben (Parameter für START) eingegeben wurden. Wenn keine Angaben geleistet wurden, wird mit Fehler abgebrochen.

Gemäss Syntax wird danach untersucht, ob eine Option gewünscht wird. Da als Option nur «/V» erlaubt ist, wird, wenn etwas anderes eingegeben wurde, ein Fehler gemeldet. Wenn die Option gewünscht wird, wird entsprechend «opt_v_fl» geladen. Im Zusammenhang mit dieser Option muss festgestellt werden, ob die benutzte DOS-Version 3.0 oder höher ist. Auf die Eigenarten älterer DOS-Versionen, insbesondere der Version 2.11, soll später eingegangen werden. Die Benutzung der Option /V ist nur in Zusammenhang mit DOS-Versionen ab 3.0 möglich.

Gemäss Syntax ist es erlaubt, einen Dateinamen mit vorgestelltem Laufwerk einzugeben. Diese Eingabe eines Laufwerks bedeutet, dass das Programm zuallererst einen Laufwerkwechsel vornehmen muss.

Bei der Eingabe des gesuchten Programmes werden keine Pfadangaben erlaubt. Dies soll die Möglichkeit von Tippfehlern auf ein Minimum reduzieren. Da START ohnedies mit seiner Suche im Hauptverzeichnis beginnen muss, ist die Pfadangabe überflüssig.

Bei der Eingabe des abzurufenden Programmes wird ebenfalls darauf Wert gelegt, dass keine Endung miteingegeben wird. Bei der direkten Eingabe eines Programmes bei der DOS-Zeile, kann man die Endung miteingeben. In der Tat nimmt sie DOS aber gar nicht wahr, sondern sucht Dateien mit dem Namen des abrufenden Programmes der Reihe nach, zuerst mit der Endung COM, danach mit der Endung EXEC und zuletzt mit der Endung BAT. Das Erstgefundene wird abgerufen.

Wenn man z.B. den Befehl «CHKDSK.BAT» eingibt, wird DOS diesen Befehl korrekt durchführen, obwohl CHKDSK eine COM ist. Befinden sich in einem Verzeichnis zwei Dateien gleichen Namens, eine mit der Endung BAT und eine mit der Endung EXEC, kann man tun was man will: DOS wird immer nur die eine mit der Endung EXEC abrufen. Das Programm START zeigt mit dem Verzicht von Punkt

```
TITLE START Version 0
PAGE ,132

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  PROGRAMM      START                          ;
;  AUTOR         DR. HERBERT STEINER            ;
;  DATUM         12.04.1988                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; NUMERISCHE KONSTANTEN

DATEI_ATTRIBUT  EQU     007H
TABULATOR       EQU     009H
VERZ_ATTRIBUT   EQU     010H
ENTER           EQU     00DH
LEER            EQU     020H
PSP_PAR_SEG     EQU     02CH
DAT_PAR_ANFANG  EQU     080H

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; ALPHA KONSTANTEN

SUMME           EQU     ADD
ABRUF           EQU     CALL
VORWAERTS       EQU     CLD
VERGLEICH       EQU     CMP
MINUS_1         EQU     DEC
ERHOEHE         EQU     INC
W_GLEICH_KLEIN  EQU     JBE
WENN_FEHLER     EQU     JC      ;DOS = CARRY FLAG
WEITER_MIT      EQU     JMP
WENN_K_FEHLER   EQU     JNC     ;DOS = CARRY FLAG
WENN_UNGLEICH   EQU     JNZ
WENN_GLEICH     EQU     JZ
ZEIGT_AUF       EQU     LEA
VON_SI_NACH_AL  EQU     LODSB
NACH_VON        EQU     MOV
BEHALTE         EQU     PUSH
RUECKRUF        EQU     POP
ZURUECK         EQU     RET
FEHLER_MELDUNG  EQU     STC     ;DOS = CARRY FLAG
VON_AL_NACH_DI  EQU     STOSB
MINUS           EQU     SUB
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INCLUDE START.MAC


CODE SEGMENT
ASSUME CS:CODE,DS:CODE
ORG 100H

ANFANG:         WEITER_MIT    HAUPTPROGRAMM


;;;;;;;;;;;;;; INTEGER      ;;;;;;;;;;;;;;;;;;;;;;

bat_fl          DB      00H
error_level     DB      00H
laufwerk        DB      ?
neu_laufwerk    DB      0FFH
opt_v_fl        DB      00H


;;;;;;;;;;;;;; ZEIGER       ;;;;;;;;;;;;;;;;;;;;;;

anf_datei       DW      ?
dat_parameter   DW      ?
end_datei       DW      ?
start_sp        DW      ?

;;;;;;;;;;;;;; KONSTANTE    ;;;;;;;;;;;;;;;;;;;;;;

all_dat_t       DB      "*.*",0
bat_t           DB      "BAT",0
comspec_t       DB      "COMSPEC="
com_t           DB      "COM",0
c_opt_t         DB      " /C "
exe_t           DB      "EXE",0
haupt_v_t       DB      "\",0
retour_v_t      DB      "..",0
trenn_zeichen   DB      TABULATOR,ENTER,LEER,"/<>"

;;;;;;;;;;;;;; REIHEN       ;;;;;;;;;;;;;;;;;;;;;;

datei_name      DB      13 DUP (00)
datei_pfad      DB      64  DUP (00)
par_zeile_5c    DB      16  DUP (00)
par_zeile_6c    DB      16  DUP (00)
verzeichnis     DB      "\", 64 DUP (00H)
verz_kennung    DB      64  DUP (01)
zweit_verz      DB      "\", 64 DUP (00H)

;;;;;;;;;;;;;; ABRUF BLOCK ;;;;;;;;;;;;;;;;;;;;;;

dos_abruf_bl    DW      0
abruf_dat_par   DW      OFFSET par_laenge
abruf_dat_seg   DW      ?
erst_par_alt    DW      OFFSET par_zeile_5c
erst_par_seg    DW      ?
zweit_par_alt   DW      OFFSET par_zeile_6c
zweit_par_seg   DW      ?

;;;;;;;;;;;;;; DTA BLOCK    ;;;;;;;;;;;;;;;;;;;;;;

dta_puffer      LABEL   BYTE
                DB      21 DUP (00)
```

```
dta_dat_atr     DB      00
                DB       8 DUP (00)
dta_dat_name    DB      13 DUP (00)

;;;;;;;;;;;;;;; PARAMETER BLOCK ;;;;;;;;;;;;;;;;;;;

par_laenge      DB      00
par_text        DB      80  DUP (00)

;;;;;;;;;;;;;;; STACK  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                DB       16 DUP ("*STACK* ")
programm_stack  DB       "* ENDE *"

;;;;;;;;;;;;;;; TEXTE  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;

INCLUDE START.TXT

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PROGRAMM AUFBAU
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

HAUPTPROGRAMM:

                NACH_VON        SP,OFFSET programm_stack
                ABRUF           WERTE_UND_SPEICHER
                ABRUF           PARAMETER_LADEN
                ABRUF           WO_DATEI
                WENN_K_FEHLER   DATEI_GEFUNDEN
                ZEIGT_AUF       SI, fehler_t
                WEITER_MIT      FEHLER
DATEI_GEFUNDEN:
                ABRUF           PROGRAMM_ABRUF
PROGRAMM_VERLASSEN:
                ABRUF           WERTE_ZURUECK
                NACH_VON        AL,error_level
                PROGRAMM_ENDE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; HAUPTROUTINEN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WERTE_UND_SPEICHER:
;;;;;;;;;;;;;;;;;;;
                ZEIGT_AUF       DX,dta_puffer
                DOS_SPEICHER

                ZEIGT_AUF       BX, ENDE
                NEU_SPEICHER
                LAUFW_HOLEN
                NACH_VON        laufwerk,AL
                ZEIGT_AUF       SI,verzeichnis
                ERHOEHE         SI
                VERZ_HOLEN
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;;;

PARAMETER_LADEN:
;;;;;;;;;;;;;;;;
                VORWAERTS
                NACH_VON        SI,DAT_PAR_ANFANG
                VON_SI_NACH_AL
                VERGLEICH       AL,0
                WENN_UNGLEICH   SUCHE_NACH_PAR_V
                ZEIGT_AUF       SI, k_angaben_t
                WEITER_MIT      FEHLER

SUCHE_NACH_PAR_V:
                VON_SI_NACH_AL
                VERGLEICH       AL,"/"
                WENN_GLEICH     OB_V
                VERGLEICH       AL,ENTER
                WENN_UNGLEICH   NICHT_BEENDET
                ZEIGT_AUF       SI, k_angaben_t
                WEITER_MIT      FEHLER
NICHT_BEENDET:
                VERGLEICH       AL,LEER
                W_GLEICH_KLEIN  SUCHE_NACH_PAR_V
                ERHOEHE         opt_v_fl
                WEITER_MIT      PAR_DATEINAME
OB_V:
                DOS_VERSION
                VERGLEICH       AL, 3
                WENN_GLEICH     OPTION_ERLAUBT
                ZEIGT_AUF       SI,option_t
                WEITER_MIT      FEHLER
OPTION_ERLAUBT:
                VON_SI_NACH_AL
                GROSS_BUCHST    AL
                VERGLEICH       AL,"V"
                WENN_GLEICH     NUR_V_ERLAUBT
                ZEIGT_AUF       SI, nur_v_t
                WEITER_MIT      FEHLER
NUR_V_ERLAUBT:
                NACH_VON        opt_v_fl,0
WO_NAECHST:
                VON_SI_NACH_AL
                VERGLEICH       AL,ENTER
                WENN_UNGLEICH   WEITER_NAECHST
                ZEIGT_AUF       SI, k_angaben_t
                WEITER_MIT      FEHLER
WEITER_NAECHST:
                VERGLEICH       AL,LEER
                W_GLEICH_KLEIN  WO_NAECHST

PAR_DATEINAME:
                VERGLEICH       AL,"\"
                WENN_UNGLEICH   OB_PUNKT
                ZEIGT_AUF       SI, kein_pfad_t
                WEITER_MIT      FEHLER
```

und Endung deutlich, dass es beim Abruf einer Datei nur auf den Dateinamen ankommt.

Nach Feststellung und Uebernahme des Dateinamens in der Variable datei_name, werden die Parameter für die abzurufende Datei festgehalten.

Der Ablauf der Uebernahme ist klar ersichtlich; nur ein Punkt scheint manchmal unklar. Bei der Eingabe von Parametern ist es nicht notwendig, Teile der Parameterzeile durch eine Leerstelle zu trennen. Dies kann man auch mit anderen Trennzeichen. Eines dieser Trennzeichen charakterisiert immer das Ende der Kommandozeile, nämlich die ENTER-Taste. Zwischen Teilen der Parameterzeile, die getrennt werden müssen, können ausser einer Leerstelle Tabulator (d.h. acht Leerstellen) oder die Zeichen / < oder > verwendet werden. Deswegen werden bei der Untersuchung der Kommandozeile nicht nur die Leerstellen und die ENTER-Taste brücksichtigt, sondern auch die anderen denkbaren Trennzeichen.

Ermittlung und Wechsel von Laufwerk und Pfad

Zwei DOS-Funktionen erlauben festzustellen, in welchem Laufwerk und in welchem Pfad man sich befindet. Mit Hilfe des Macros «LAUF_HOLEN» wird ermittelt, welches Laufwerk aktiv ist.

Ermittlung des aktuellen Laufwerkes	
Register	Inhalt
AH	hex 19
Rückkehr	
AL	Kennziffer des Laufwerkes A=0, B=1, usw.

Als Rückmeldung erhalten wir eine Ziffer, die das Laufwerk charakterisiert. Durch Addition von 65 (ASCII-Wert für A) erhält man das entsprechend darstellbare Zeichen. START benutzt das Macro «LAUF_WECHSEL» um den im Programm benötigten Laufwerkwechsel vorzunehmen.

Laufwerkwechsel	
Register	Inhalt
DL	Kennziffer des Laufwerkes A=0, B=1, usw.
AH	hex 0E
Rückkehr	
AL	Anzahl der ansprechbaren Laufwerke

Diese Funktion hat mehrere Nachteile, die des öfteren zu schweren Programmfehlern führen. Wenn eine Kennziffer eines nicht ansprechbaren Laufwerkes eingegeben wird, meldet diese Funktion bei Rückkehr keinen Fehler. Es wird einfach kein Laufwerkwechsel vorgenommen. Um zu wissen ob der Laufwerkwechsel tatsächlich stattgefunden hat, muss man nochmals fragen, welches das aktuelle Laufwerk ist. Bei keiner Aenderung kann man auf eine falsche Eingabe des Laufwerkes schliessen.

Andererseits ergibt sich auch ein Fehler bei der Meldung der Anzahl der ansprechbaren Laufwerke. Ab DOS-Version 3.0 meldet diese Funktion fünf ansprechbare Laufwerke, unabhängig davon ob sie in der Tat aktiv sind oder nicht. Dies hat im vorliegenden Programm keine Bedeutung.

Zuletzt sei noch darauf hingewiesen, dass, während die zwei abgehandelten Laufwerkfunktionen eine einheitliche Kennziffer benutzen, dies nicht immer der Fall ist.

Das Macro «VERZ_HOLEN» gestattet, im Programm das aktuelle Verzeichnis festzustellen, in welchem man sich befindet.

Ermittlung des aktuellen Pfades

Register	Inhalt
DL	Kennzahl des Laufwerkes Aktiv = 00, A=1, B=2, usw.
l DS:DI	Adresse des Puffers, wohin der Pfad gespeichert wird (mindestens 64 Bytes lang)
AH	hex 47
Rückkehr	
AX	(wenn CARRY FLAG gesetzt) Fehlercode ansonsten Pfadangabe ohne anfänglichen «\». Wird mit 00 beendet (ASCIIZ-String)

Wie schon darauf hingewiesen, wird hier zur Kennzeichnung des Laufwerkes eine andere Methode benutzt als bei den zwei gezeigten Laufwerkfunktionen. Hier wird ein Fehler gemeldet, wenn eine falsche Laufwerkangabe eingegeben wurde. Zu beachten ist auch, dass die zurückgegebene Pfadangabe nicht das anfängliche «\» beinhaltet. Programmintern muss darauf geachtet werden, dass diese fehlende Angabe ergänzt wird, da ansonsten bei dem Versuch, einen Pfad unter Benutzung dieser Pfadangabe zu wechseln zu Fehlern führen kann.

Das Macro «VERZ_WECHSEL» erlaubt es, einen Pfadwechsel vorzunehmen.

Wechsel des Pfades

Register	Inhalt
DS:DX	Adresse des Puffers, wo die neue Pfadangabe gespeichert wurde (kann Laufwerk beinhalten)
AH	hex 38
Rückkehr	
AX	(wenn CARRY FLAG gesetzt) Fehlercode

Diese Funktion verlangt einen Zeiger auf die neue Pfadangabe. Diese muss auch das anfängliche «\» beinhalten. Ausserdem ist es möglich, eine Laufwerkangabe vorzustellen. Dies kann jedoch zu einem Programmfehler führen. Der Pfadwechsel wird im Laufwerk vorgenommen, ohne dass ein Laufwerk stattfindet. Man kann den aktuellen Pfad eines Laufwerkes ändern, aber kein Laufwerk-

```
OB_PUNKT:
                VERGLEICH       AL,"."
                WENN_UNGLEICH   OB_LAUFWERK
                ZEIGT_AUF       SI, punkt_t
                WEITER_MIT      FEHLER
OB_LAUFWERK:
                NACH_VON        anf_datei,SI
                MINUS_1         anf_datei
                VON_SI_NACH_AL
                VERGLEICH       AL,":"
                WENN_GLEICH     LAUFWERK_ANGABE
                NACH_VON        DL, laufwerk
                NACH_VON        neu_laufwerk, DL
                WEITER_MIT      OHNE_LAUFWERK
LAUFWERK_ANGABE:
                NACH_VON        DL,[SI-2]
                GROSS_BUCHST    DL
                MINUS           DL,"A"
                NACH_VON        neu_laufwerk, DL
                LAUFW_WECHSEL
                LAUFW_HOLEN
                VERGLEICH       AL, DL
                WENN_GLEICH     WECHSEL_OK
                NACH_VON        neu_laufwerk, AL
                ZEIGT_AUF       SI, laufwerk_t
                WEITER_MIT      FEHLER
WECHSEL_OK:
                NACH_VON        anf_datei,SI
                BEHALTE         SI
                ZEIGT_AUF       SI, zweit_verz
                ERHOEHE         SI
                VERZ_HOLEN
                RUECKRUF        SI
OHNE_LAUFWERK:
                VERGLEICH       AL,"\"
                WENN_UNGLEICH   PRUEF_ENDE_DATEI
                ZEIGT_AUF       SI,kein_pfad_t
                WEITER_MIT      FEHLER
PRUEF_ENDE_DATEI:
                SUCH_OB_SI_IN   trenn_zeichen,6
                WENN_GLEICH     ARBEITS_PAR
                VON_SI_NACH_AL
                WEITER_MIT      OHNE_LAUFWERK

ARBEITS_PAR:
                MINUS_1         SI
                NACH_VON        dat_parameter,SI

                NACH_VON        SI,anf_datei
                ZEIGT_AUF       DI,datei_name
LADEN_DATEINAME:
                VON_SI_NACH_AL
                VERGLEICH       AL, "."
                WENN_UNGLEICH   WEITER_LADEN
                ZEIGT_AUF       SI, zuviel_punkt_t
                WEITER_MIT      FEHLER
WEITER_LADEN:
                VON_AL_NACH_DI
                VERGLEICH       SI, dat_parameter
                WENN_UNGLEICH   LADEN_DATEINAME

PUNKT_LADEN:
                NACH_VON        AL,"."
                VON_AL_NACH_DI
                VERGLEICH       BYTE PTR [DI+3], 00
                WENN_GLEICH     PAR_OHNE_FEHLER
                ZEIGT_AUF       SI, zu_lang_t
                WEITER_MIT      FEHLER
PAR_OHNE_FEHLER:
                NACH_VON        end_datei,DI
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;;;
WO_DATEI:
;;;;;;;;;
                ZEIGT_AUF       DX, haupt_v_t
                VERZ_WECHSEL
                ZEIGT_AUF       DI,verz_kennung
                NACH_VON        BP,DI

ANFANG_DAT_SUCHE:
                ABRUF           DATEI_ABRUF
                WENN_FEHLER     ANFANGS_VERZ
                ZURUECK         ;;;;;;;;;;;;;

VERZEICHNIS_RAUF:
                VERGLEICH       BP,OFFSET verz_kennung
                WENN_UNGLEICH   SUCH_FORTSETZUNG
ENDE_SUCHE:
                FEHLER_MELDUNG
                ZURUECK         ;;;;;;;;;;;;;

SUCH_FORTSETZUNG:
                ZEIGT_AUF       DX, retour_v_t
                VERZ_WECHSEL
                NACH_VON        BYTE PTR [BP],1
                MINUS_1         BP

ANFANGS_VERZ:
                REG_NULL        BL
                ZEIGT_AUF       DX,all_dat_t
                NACH_VON        CX,VERZ_ATTRIBUT
                SUCHE_ERST
                WENN_FEHLER     VERZEICHNIS_RAUF
PRUEFEN_OB_VERZ:
                VERGLEICH       BYTE PTR [dta_dat_atr],VERZ_ATTRIBUT
                WENN_UNGLEICH   WEITER_VERZ
                VERGLEICH       BYTE PTR [dta_dat_name],'.'
                WENN_GLEICH     WEITER_VERZ
                ERHOEHE         BL
                VERGLEICH       BL,[BP]
                WENN_UNGLEICH   WEITER_VERZ
```

```
                ERHOEHE         BYTE PTR [BP]
                ZEIGT_AUF       DX,dta_dat_name
                VERZ_WECHSEL
                ERHOEHE         BP
                WEITER_MIT      ANFANG_DAT_SUCHE
WEITER_VERZ:
                SUCHE_NAECHST
                WENN_FEHLER     VERZEICHNIS_RAUF
                WEITER_MIT      PRUEFEN_OB_VERZ
                                ;;;;;;;;;;;;;;;;;;;;;;
PROGRAMM_ABRUF:
;;;;;;;;;;;;;;;;
                NACH_VON        SI,dat_parameter
                ZEIGT_AUF       DI,par_zeile_5c
                PAR_ZEILE
                ZEIGT_AUF       DI,par_zeile_6c
                PAR_ZEILE

                NACH_VON        start_sp,SP

                NACH_VON        abruf_dat_seg,DS
                NACH_VON        erst_par_seg,DS
                NACH_VON        zweit_par_seg,DS

                VERGLEICH       bat_fl,1
                WENN_UNGLEICH   VORBEREITUNG
                ZEIGT_AUF       SI,c_opt_t
                ZEIGT_AUF       DI,par_text
                DOPPELW_SI_DI
                ABRUF           SETZ_DAT_PFAD
                MINUS_1         DI
                ABRUF           SETZ_PARAMETER

                NACH_VON        BX,PSP_PAR_SEG
                NACH_VON        DS,[BX]
                REG_NULL        AX
SUCH_COMSPEC:
                NACH_VON        SI,AX
                ERHOEHE         AX
                VERGLEICH_SI    comspec_t, 8
                WENN_UNGLEICH   SUCH_COMSPEC
                NACH_VON        DX,SI
                WEITER_MIT      DURCHFUEHRUNG
VORBEREITUNG:
                ZEIGT_AUF       DI,datei_pfad
                ABRUF           SETZ_DAT_PFAD
                ZEIGT_AUF       DI,par_text
                ABRUF           SETZ_PARAMETER
                ZEIGT_AUF       DX,datei_pfad
DURCHFUEHRUNG:
                ZEIGT_AUF       BX,dos_abruf_bl
                NEU_PROGRAMM
                NACH_VON        AX,CS
                NACH_VON        SS,AX
                NACH_VON        SP,CS:start_sp
                NACH_VON        ES,AX
                NACH_VON        DS,AX
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;


FEHLER:
;;;;;;;
                ERHOEHE         error_level
                ABRUF           ZEIG_TEXT
                ZEIGT_AUF       SI,start_t
                ABRUF           ZEIG_TEXT
                WEITER_MIT      PROGRAMM_VERLASSEN
                                ;;;;;;;;;;;;;;;;;;;;;;
WERTE_ZURUECK:
;;;;;;;;;;;;;;
                NACH_VON        DL, laufwerk
                VERGLEICH       neu_laufwerk,DL
                WENN_GLEICH     VERZ_ZURUECK
                ZEIGT_AUF       DX, zweit_verz
                VERZ_WECHSEL
                NACH_VON        DL,laufwerk
                LAUFW_WECHSEL
VERZ_ZURUECK:
                ZEIGT_AUF       DX, verzeichnis
                VERZ_WECHSEL
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; HILFSROUTINEN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DATEI_ABRUF:
                NACH_VON        bat_fl, 00
                ZEIGT_AUF       DX, datei_name
                NACH_VON        BX,end_datei
                ZEIGT_AUF       SI, com_t
                ABRUF           DATEI_VERGLEICH
                WENN_K_FEHLER   DAT_RUF_ENDE
                ZEIGT_AUF       SI, exe_t
                ABRUF           DATEI_VERGLEICH
                WENN_K_FEHLER   DAT_RUF_ENDE
                ZEIGT_AUF       SI, bat_t
                NACH_VON        bat_fl,1
                ABRUF           DATEI_VERGLEICH
DAT_RUF_ENDE:
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;
```

wechsel findet statt. Sollte jedoch später einer stattfinden, wird dann bei diesem Laufwerk der geänderte Pfad als aktueller Pfad erscheinen. Der Aufbau dieser Funktion entspricht dem Befehl «CHDIR» oder «CD». Der Zeiger wird auf das Argument von CD gesetzt.

Im Programmablauf ist es daher mit Hilfe dieser vier Funktionen sehr einfach, den Ausgangspunkt zu ermitteln, programmintern die entsprechenden Wechsel vorzunehmen, um danach zu den eingangs ermittelten Werte durch Wechsel zurückzukehren.

Suchen einer Datei

Wie man eine Datei in einem Lauferk findet, wurde schon ausführlich im Zusammenhang mit dem Programm WO dargestellt. Der gleiche Routine-Aufbau wird im vorliegenden Programm benutzt, um die Datei zu finden. Sollte die Datei nicht gefunden werden, wird eine Fehlermeldung abgegeben.

Beim Suchen einer Datei wird zuerst dem Namen nach gesucht. Genau wie bei DOS wird bei der Suche ermittelt, ob sich eine solche Datei mit der Endung COM, danach mit der Endung EXEC und zuletzt mit der Endung BAT finden lässt. Diese Art der Suche wurde vorgenommen, um volle Kompatibilität mit allen DOS-Befehlen zu gewährleisten.

Abrufen einer Datei

Relativ schwierig gestaltet sich die Aufgabe, innerhalb eines Programmes ein anderes Programm abzurufen. Um diese Tätigkeit durchzuführen sind folgende Schritte notwendig:

1. Das abgerufene Programm muss genügend Speicherplatz zugewiesen erhalten.
2. Vorbereitung und Durchführung des Abrufes.
3. Bei Rückkehr zum ursprünglichen Programm, die Programmfortsetzung sicherstellen.

Wenn DOS ein COM-Programm aufruft, wird der vorhandene Speicher diesem Programm zur Verfügung gestellt. Im Zusammenhang mit dem Programm LIES wurden die entsprechenden Funktionen besprochen, die es ermöglichen, den Speicherbedarf des Programmes auf das benötigte zurückzubringen. Soll ein Programm innerhalb eines anderen Programmes abgerufen werden, wird der Speicherbedarf des ursprünglichen Progammes so manipuliert, dass er auf das Mindestnotwendige reduziert wird.

Programmabruf

Register	Inhalt
DS:DX	Adresse der Dateinamen (mit Laufwerk und Pfadangabe, muss mit 00 enden)
ES:BX	Zeiger auf Abrufblock
AL	Funktionscode 00 = laden und durchführen 03 = nur laden
AH	hex 48
Rückkehr	
AX	(wenn CARRY FLAG gesetzt) Fehlercode

Hier liegt einer der grössten Vorteile, der in Assembler geschriebenen Programme: Der geringe Speicherbedarf solcher Programme. Je weniger Speicher vom ursprünglichen Programm benötigt wird, desto grösser sind die Chancen, dass das abgerufene Programm genügend Speicher für sich selbst vorfindet. Ansonsten folgt die DOS-Meldung, dass das neue Programm nicht genügend Speicherplatz zur Verfügung hat. Diese DOS-Meldung erfolgt über die Funktion, die kritische Fehler verwaltet. Sie muss daher nicht programmintern Berücksichtigung finden.

Das Macro «NEU_PROGRAMM» ermöglicht das Abrufen eines Programmes.

Um ein Programm abzurufen, benötigen wir daher eine Stelle im Programm wo sich der vollständige Name (mit korrekter DOS-Syntax) befindet. Dieser Name muss Laufwerk oder Pfadangabe beinhalten (wenn sich das Programm ausserhalb des aktiven Laufwerkes oder Pfades befindet) und mit 00 enden. Was mit dieser Datei gemacht werden soll, bestimmt der Funktionscode. Zwei Funktionen sind möglich: Einerseits das Programm laden und durchführen, andererseits das Programm nur zu laden. Auf die zweite Funktion soll im Rahmen dieser Programmbesprechung nicht weiter eingegangen werden. Wenn das Programm geladen und durchgeführt werden soll, ist es notwendig, im Programm selbst einen Abrufblock zu bilden, der die notwendige Information für den Ablauf des neuen Programmes beinhaltet.

Struktur des Abrufblocks

Stelle	Länge	Inhalt
1	2	Segmentadresse der Programmgebung (00 bedeutet beim Ursprungsprogramm)
2	4	Segment und Offset der Kommandofile für das abzurufende Programm
3	4	Segment und Offset des ersten Kontrollblocks
2	4	Segment und Offset des zweiten Kontrollblocks

Wenn DOS ein Programm abruft, legt es fest, wo sich die Informationen für das Programm befinden (Programmvorspann). Es legt weiter eine Adresse eines bis zu 32 KByte langen Bereiches fest, in welcher Informationen für das Programm gespeichert werden. Dieser Bereich beinhaltet z.B. wo sich die COMMAND.COM befinden, wie das Programm abgerufen wurde und welche SET- oder PATH-Befehle berücksichtigt werden müssen. Um ein Programm abzurufen, wie es im vorliegenden Fall getan wird, ist es nur notwendig, im Abrufblock eine «00» zu setzen. Das abgerufene Programm wird die gleiche Information erhalten wie das ursprüngliche hatte.

Auf die Bedeutung von Segment- und Offset-Angaben soll in diesem Artikel noch nicht weiter eingegangen werden. Es sei nur festgestellt, dass Segment und Offset die physikalische Adresse darstellen. Im Assembler selbst braucht man die Adressen nicht zu berechnen, sondern kann direkt «SEG» und »OFFSET» benutzen.

Beim Abruf eines Programmes ist es sehr oft notwendig, Argumente für dieses Programm mitzugeben. Die programminterne Darstellung der Kommandozeile für das abgerufene Programm muss der Darstellung entsprechen, die wir programmintern ab Stelle hex 80 vorfinden. Zuerst

```
DATEI_VERGLEICH:
                NACH_VON        DI,BX
                DOPPELW_SI_DI
                NACH_VON        CX,DATEI_ATTRIBUT
                SUCHE_ERST
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;;;

SETZ_DAT_PFAD:
                VERGLEICH       opt_v_fl,1
                WENN_GLEICH     SETZ_DAT_NAME
                NACH_VON        AL,neu_laufwerk
                SUMME           AL,"A"
                VON_AL_NACH_DI
                NACH_VON        AL,":"
                VON_AL_NACH_DI
                NACH_VON        AL,"\"
                VON_AL_NACH_DI
                NACH_VON        SI,DI
                VERZ_HOLEN
                ABRUF           WERTE_ZURUECK
                NACH_VON        DI,SI
                VERGLEICH       BYTE PTR [DI],0
                WENN_GLEICH     SETZ_DAT_NAME
DAT_NAME_ENDE:
                ERHOEHE         DI
                VERGLEICH       BYTE PTR [DI],0
                WENN_UNGLEICH   DAT_NAME_ENDE
                NACH_VON        AL,"\"
                VON_AL_NACH_DI
SETZ_DAT_NAME:
                ZEIGT_AUF       SI,datei_name
WEITER_DAT_NAME:
                VON_SI_NACH_AL
                VON_AL_NACH_DI
                VERGLEICH       AL,0
                WENN_UNGLEICH   WEITER_DAT_NAME
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;;;

SETZ_PARAMETER:
                NACH_VON        SI,dat_parameter
SUCH_PARAMETER:
                VON_SI_NACH_AL
                VON_AL_NACH_DI
                VERGLEICH       AL,ENTER
                WENN_UNGLEICH   SUCH_PARAMETER
                REG_NULL        CL
                ZEIGT_AUF       SI,par_text
ZAEHLUNG_PAR:
                VON_SI_NACH_AL
                VERGLEICH       AL,ENTER
                WENN_GLEICH     PAR_ENDE
                ERHOEHE         CL
                WEITER_MIT      ZAEHLUNG_PAR
PAR_ENDE:
                NACH_VON        par_laenge,CL
                ZURUECK         ;;;;;;;;;;;;;;;;;;;;;;;;

ZEIG_TEXT:      VON_SI_NACH_AL
                VERGLEICH       AL,00
                WENN_GLEICH     ZEIG_ENDE
                NACH_VON        DL,AL
                ZEIG_DL
                WEITER_MIT      ZEIG_TEXT
ZEIG_ENDE:
                ZURUECK     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ENDE    LABEL   BYTE
CODE ENDS
END  ANFANG
```

wird die Länge der Kommandozeile in Bytes angegeben und das ganze, wie von DOS erwartet, mit der ENTER-Taste (hex 0D) abgeschlossen.

Erster und zweiter Kontrollblock sind programminterne Speicherstellen, die für ältere DOS-Versionen bestimmt sind. Sie werden aus Kompatibilitätsgründen beibehalten. Manche Programme in älteren Versionen (z.B. WordStar) benutzen noch immer diese Kontrollblöcke um zu erfahren, welche Argumente an das Programm weitergeleitet wurden. Die Herstellung solcher Kontrollblöcke ist relativ kompliziert und wird im Detail unübersichtlich. Dies dürfte auch einer der Gründe sein, warum diese Technik in neueren DOS-Versionen überhaupt nicht mehr benutzt wird. DOS stellt jedoch eine Funktion zur Vefügung, die es ermöglicht, auf einfache Art und Weise solche Kontrollblöcke herzustellen. Diese Funktion wird von Macro «PAR_ZEILE» abgerufen.

Wenn DOS mit Hilfe der dargestellten Funktionen Programme abruft, muss ein Unterschied zwischen den BATCH-Dateien und den COM- oder EXE-Programmen berücksichtigt werden. Nach Abruf eines COM- oder EXE-

Programmes kehrt DOS automatisch zum Ursprungsprogramm zurück. Bei BATCH-Dateien ist dies nicht der Fall. Da hilft nur eines. Beim Schreiben von BATCH-Dateien ist es möglich, den Befehl «COMMAND /C» zu benützen. Dieses Kommando erlaubt es, dass eine BATCH-Datei abgerufen wird und zum Ursprung zurückkehrt. Im Falle einer BATCH-Datei wird dem eigentlichen Dateinamen das entsprechende «COMMAND /C» vorausgestellt.

Diese Darstellung klingt sehr kompliziert und sie ist es auch. Hier hilft im Laufe der Zeit nur eines: Das vorliegende Programm zu benutzen, um langsam aber sicher zu experimentieren. Versuchen Sie am Anfang nur ein ganz kleines Programm zu schreiben, das ein zweites Programm abruft.

Als Schlussbemerkung zu diesem Thema sei noch darauf hingewiesen, dass sämtliche Arbeitsregister bei Abruf eines neuen Programmes zerstört werden. Man muss daher Sorge tragen, dass man bei Abruf des Programmes die Register programmintern abgesichert hat. Bei der Rückkehr vom abgerufenen Programm müss der Ursprungszustand wieder hergestellt werden.

Fehlermeldungen

Einer der häufigsten Programmierfehler ist die Nicht- oder Minderberücksichtigung der Fehlermöglichkeiten des Programmanwenders. Der Programmierer kennt sein Programm und es fällt ihm daher oft schwer sich vorzustellen, welche Fehlermöglichkeiten sich bei der Programmanwendung ergeben. Bevor ein Programm an die Anwender weitergegeben wird, sollten zwei Analysen über mögliche Fehler stattfinden:

1. Welche Möglichkeiten hat der Verbraucher, falsche Angaben programmintern zu leisten.

2. Welche programminterne Konsequenzen ergeben sich aus solchen Fehlern.

Mir ist folgendes passiert: Ich habe mit einem Programm gearbeitet, das an einer bestimmten Stelle anfragte, ob ein Drucker angeschlossen sei oder nicht. Der Drucker war wohl angeschlossen, aber nicht aktiv (Off-Line). Ich habe eine Menge Arbeiten im Rahmen des Programmes erledigt und als ich voller Erwartung den Ausdruck der Ergebnisse abwartete, stürzte der Computer ab. Das einzig mögliche war, wieder zu starten. Die gesamte Arbeit war verloren. Der Programmierer glaubte, bei der Frage ob ein Drucker angeschlossen sei oder nicht, davon ausgehen zu können, dass der Drucker druckbereit sei. Mein Aerger war gross und sicherlich färbte er auf mein Urteil über das Programm ab. Wem ist nicht schon einmal etwas Aehnliches passiert?

Das vorliegende Programm versucht, eine Methode zu zeigen, wie man Fehlermeldungen gestaltet und wie man programmintern für einen korrekten Abgang sorgt.

Obwohl es manchmal ziemlich viel Arbeit bedeutet, ist das grundlegende Prinzip einer Fehlermeldung, dass diese spezifisch für den entstandenen Fehler ist. Die hier benutzte Technik lautet: Nach Ermittlung des Fehlers die spezifische Meldung suchen und von dort aus zu einem besonderen Programmende zu gehen.

Ein Programm wird mit mehreren DOS-Funktionen beendet. Eine davon wird heute als Standard empfohlen:

Programm beenden	
Register	Inhalt
AL	Fehlercode (für Folgeprogramme oder BATCH-Dateien)
AH	hex 4C

Diese Funktion sollte nun immer benutzt werden, da sie es ermöglicht, eine Fehlermeldung abzugeben, die nachträglich von anderen Programmen erkannt werden kann. Dies wird insbesondere beim Befehl «IF ERRORLEVEL» benötigt. Ausserdem erlaubt es den Einbau von START in allen möglichen BATCH-Dateien.

Randbemerkung über DOS 2.11

Wenn Ihr Computer mit dem Betriebssystem DOS 2.11 arbeitet, geben Sie folgenden Befehl ein:

A: UNSINN

Obwohl sich sicher das Programm UNSINN nicht in Ihrem Programm befindet (sollten Sie ein solches Programm haben, ändern Sie die Bezeichnung UNSINN durch irgendeinen Namen, der nicht als Programmname benutzt wird), werden Sie feststellen, dass keine Fehlermeldung erfolgt.

Dies deutet auf eine Serie von Betriebsfehlern hin, die DOS 2.11 noch inne hat. Am klarsten zeigt es sich am dargestellten Beispiel. DOS 2.11 ist nicht imstande beim Programmabruf eine korrekte Pfadangabe zu berücksichtigen.

DOS 2.11 verlangt daher, dass man den langsamen Weg des stufenweisen Verzeichniswechsels vornimmt. Erst wenn man im Verzeichnis ist wo sich das Programm befindet, ist ein Abruf des Programmes möglich.

Daher wird programmintern untersucht, mit welcher DOS-Version das Programm zusammenarbeiten soll. Wenn diese geringer als drei ist, dann wird exakt gemäss der Version der Programmabruf vorgenommen (notfalls zuerst den entsprechenden Verzeichniswechsel vornehmen).

Technischer Hinweis

Im Zusammenhang mit Assembler und Hochsprachen wird sehr oft von der Möglichkeit des modularen Programmaufbaues gesprochen. Modularer Aufbau heisst, dass man das Programm nicht immer als gesamtes abarbeiten muss. Man bildet Teile und erst nachdem man sicher ist, dass diese funktionieren, fügt man sie zu einem Gesamtprogramm zusammen. Ein erster Schritt in dieser Richtung stellt das Benutzen von INCLUDE-Dateien dar. Sowohl Assembler als auch andere Hochsprachen erlauben es, Teile eines Programmes als selbständige Datei zu schreiben (in unserem Fall «START.MAC» und «START.TXT»). Der Befehl «INCLUDE» sagt dem Compiler, dass die angesprochene Datei an der jeweiligen Stelle eingefügt werden muss. Ein kleines Problem ergibt sich bei der Verwendung von INCLUDE-Dateien. Der Compiler geht davon aus, dass diese Dateien sich im aktuellen Laufwerk und Verzeichnis befinden. Im Zweifelsfalle ist es immer besser, den Namen der Dateien mit Laufwerk und Pfad vollständig einzugeben. □

dataland

Wir suchen unkomplizierte, junge, dynamische und verkaufsorientierte **Mitarbeiterinnen** und **Mitarbeiter**

- PERSONAL-COMPUTER-**Verkäufer** mit qualifizierten Branchenkenntnissen
- SOFTWARE-**Berater** mit qualifizierten Programmierkenntnissen
- EDV-**Techniker** mit qualifizierten Reparatur- und Supportkenntnissen für Printer und Computer
- TELEFON-**Verkaufsberater** mit qualifizierten Markenproduktkenntnissen (PC/Printer/Software) für telefonische Anfragenbearbeitung

Eine solide Einführung bzw. Ausbildung in die Aufgabenbereiche ist gewährleistet; insbesondere **auch für zielstrebige «Berufsbeginner»** mit qualifizierten Computerkenntnissen.

Die Konzeption «DATALAND Informatik-Zentrum» hat sich erfolgreich in der Ostschweiz in Herisau AR bewährt. Dort betreiben wir ein 400 m² grosses DATALAND Informatik-Zentrum». Zur Realisierung unserer **Expansionspläne für 1988** in den Grossräumen

- **Zürich**
- **Zentralschweiz**
- **Aargau**
- **Zürichsee**
- **Berner Oberland**
- **Ostschweiz**

suchen wir noch «aufgestellte und gepflegte» **Teammitglieder!**

Die Struktur ist definiert. Nun gilt es, diese weiterhin **erfolgreich** zu verwirklichen und **kreativ** weiterzuentwickeln.

Möchten Sie uns dabei helfen?

Dann erwarten wir gerne Ihre **Kurzbewerbung mit Foto**. Oder vereinbaren Sie gleich einen Termin für ein **Informationsgespräch** mit unserer Frau S. Brunner, Telefon 071/52 21 22!

INFORMATIK-ZENTRUM

dataland INFORMATIK-ZENTRUM · datatronic INFORMATIK & CONSULTING · datasupport SCHULUNG & TECHNIK

9101 HERISAU/AR · OBERDORFSTRASSE 143 · 071/52 21 21

Möchten Sie sich in das interessante Gebiet der analogen und digitalen Übertragungstechnik einarbeiten?
Und haben Sie Freude, Personal zu führen?

PTT

Die Fernmeldekreisdirektion Zürich sucht einen

Elektroingenieur HTL

(Fachrichtung Fernmeldetechnik)

dem sie die folgenden Aufgaben anvertrauen möchte:

Planung, Bau und Betrieb von Übertragungsstellen: Inbetriebnahme, Unterhalt und Störungsbehebung an Übertragungsanlagen (Trägerfrequenztechnik, PCM-Systeme, Niederfrequenz).
Abklärung von technischen Problemen.
Personalführung und -ausbildung, Organisation.
Ihr Arbeitsort: Zürich-Selnau und Herdern (mit Personalrestaurant)

Wenn Sie über Organisationstalent verfügen und Sie Schweizer Bürger sind (vielleicht sogar im Idealalter von 25–35 Jahren?), so möchten wir Sie gerne kennenlernen.

Rufen Sie einfach Herrn Zünd an (Tel. 01/204 86 51), er wird Sie über die Stelle, die zeitgemässe Entlöhnung, die ausgebauten Sozialleistungen, Ihre Weiterbildungsmöglichkeiten sowie die Arbeitsbedingungen gerne orientieren.

Fernmeldekreisdirektion Zürich
Postfach
8021 Zürich

FINANZ-
BUCHHALTUNG

PFIB-4 standardmässig MIT
Budget- und Vorjahresvergleich, freiem
4-stelligem Kontoplan, Standardtexten,
Probebilanz, Mandantenfähigkeit, Passwort-
schutz, Anleitung mit Beispiel, Fenstertechnik,
Ausgaben auf Drucker oder Disk, Sortier-
möglichkeiten. OHNE Kopierschutz.
Alles inklusiv Fr.950.-, vollständige
Probier-Version Fr.50.-

ARNOLD, 061/22 17 52
Rütlistrasse 39
4051 Basel

COMPITRON AG
COMPUTERS

Effiziente Datenverarbeitung
Mehr Zeit für andere Aufgaben
Kosten senken

Wenn Ihnen diese Punkte wichtig sind, sollten Sie uns unverbindlich anrufen

GTEK 286/386 Professional Computer

Die professionelle Lösung aus Taiwan für den Einsatz in Industrie-, Dienstleistungs- und Handelsbetrieben

Generalvertretung für den deutschsprachigen Raum
COMPITRON AG, Querstrasse 8, 8105 Regensdorf (ZH/N20)
Telefon 01/841 00 11, Fax 01/841 00 24 *(24 Stunden)*
PS. Wir suchen qualifizierte Wiederverkäufer, Softwarehäuser und Verkaufsmitarbeiter

GTEK

Zu verkaufen

Ein Superbrain QD für CP/M 2.2. Ausgezeichneter Zustand, Fr. 500.-.
☎ 01/954'04'11 W. Obrist

NCR Decision Mate V Mod. 1202. Hardware: 256 KByte RAM, 2x5,25 Zoll 360 KByte Floppydisk. CRT: 25x80 resp. 640x400 Pixel in acht Farben, Grafikp. PD7220. Interface: Centronics-Parallel. CPUs: Z80, 8088 Mathprozessor 8087.
☎ 062/41'14'76

Kaypro IV mit Drucker C.Itho 1550 und viel Software. Guter Zustand. Preisidee Fr. 1'500.-. ☎ 033/23'48'74 ab 19 Uhr

HP-86B Personal Computer mit CH-Tastatur; Monitor 82913 (12 Zoll), Doppel-Floppy-Drive 3,5 Zoll 9121; Drucker Thinkjet; RS232-Interface 82939; AP- und I/O-ROM; Word/80. Preis Fr. 4'500.-.
☎ 028/23'17'78

Doppel-Floppy, 80 Track Hard-sektoriert zu Sorcerer. ☎ G: 071/22'35'39, P: 01/980'04'89

Toshiba T1100, 512 KByte, IBM Writing und Filing Ass. VB Fr. 1'000.-.
☎ 031/87'12'34

Ein Normalpapier-Kopierer **Toshiba BD 4511**, Format A6-B4, unter Vertrag gewartet, revidiert, Fr. 1'400.-.
☎ 01/954'04'11 W. Obrist

HP-18C Business Consultant Menü, Meldungen und Handbuch in deutsch. Zusätzliche Anleitung für Banking Consultant. Neuzustand. NP Fr. 430.-, VP Fr. 200.-.
☎ P: 01/363'36'37

Matrix-Drucker **OKI-84ML**, komplett revidiert. Einzelblatt und Endlos-Papier. Diverse Schriften. IBM-Graphwriter-Zusatz. Preis Fr. 1'000.-. ☎ 01/361'68'68 Zefir AG

Sharp PC2500; 72 KByte ROM, 21 KByte RAM; Tabellen- und Grafiksoftware integriert, 114 mm 4-Farben-Plotter, 4x24 Display (32x150 Dots); Schreibmaschinentastatur; NP Fr. 1'200.-, VP Fr. 700.-.
☎ P: 01/810'38'50

Epson PX-4 mit 128 KByte Erweiterung, Kassettenlaufwerk, Turbo-Pascal V3.0, Basic. VP Fr. 1'200.-. ☎ 01/813'10'09

Bondwell 8, portabel, 512 KByte RAM, 3.5 Zoll Laufwerk 720 KByte, LCD-Anzeige hintergrundbeleuchtet, mit MS-DOS und diversen Programmen. Preis ca. Fr. 1'500.- oder dem Meistbietenden.
☎ 041/85'24'19 ab 18 Uhr

HP-IL-Disk-Drive 3.5 Zoll, Fr. 990.-. HP-IL-Tintenstrahldrucker, Fr. 790.-. HP-71B (inkl. Texteditor-, IL-, 2 RAM, 4 KByte-Modul), Fr. 850.-. HP-41CV (inkl. Statistik-, Math-, X-Functions-Modul), Fr. 400.-.
☎ 031/54'17'43

Inserateschluss für M+K 88-4 ist am 22. Juni

Zwei **DEC Rainbow 100+**: 384 KByte RAM und 896 KByte RAM. Dazu ein Monitor bernstein, ein Monitor farbig DEC/Hitachi VR 241 A. Zwei Printer LA 50. Ein Printer Itoh 8510. Angebot an Fritz Bieri, Jona, ☎ 055/28'36'54

Epson PX-8 Handheld mit Word-Star und Calc-Star. Diskettenstation PF10. Zusammen Fr. 1'000.-. ☎ 031/45'68'44 abends

Ein Matrix-Printer **NEC P5**, fast neu, wenig gebraucht, Fr. 2'500.-. ☎ 01/954'04'11 W. Obrist

Epson PX-8 Handheld mit Floppy PF-10; wenig gebraucht, Fr. 1'500.- (wird nur zusammen abgegeben). B. Koch,
☎ G: 033/55'22'04, P: 033/57'23'15

MS Windows 386, Präsentations-Manager Version 2,0. Benötigt DOS 3.1 und höher, Intel 80386 Prozessor! Neu, unbenützt und mit Originaldokumentation in Englisch. Preis Fr. 320.-.
☎ G: 031/67'68'21, P: 031/45'63'38

HP-110 Portable, Fr. 800.-. Epson-Drucker FX-100, Fr. 500.-. Epson-Drucker FX-1000, Fr. 800.-. P. Strauch,
☎ 01/740'76'75

DEC Rainbow 100+, mit 5 MB Harddisk und 256 KByte RAM, 2x400 KByte Disk-Drives, Drucker LA50, Zubehör. Fr. 2'000.-.
☎ 031/23'37'17

Olivetti M15, portabler PC, LCD-Dysplay, 640 KByte RAM, 2x3,5 Zoll Floppies, inkl. MS-DOS 3.27, Preis Fr. 950.-. Schreibmaschine Silver-Reed 500, Preis ca. Fr. 80.-. ☎ 041/55'84'29, 19-20 Uhr

Toshiba T3100 portable, AT-kompatibel, 8 MHz, 20 MB Harddisk, 640 KByte und 2 MB Speicher, integriertes Modem, DOS 3.3, Fr. 7'500.-. ☎ 040/60'00'20

COMPUTER-BÖRSE

IBM-PC kompatibel, 512 KByte, 2x360 KByte Laufwerke, Multifunktionskarte, 2xSer. und Uhr, Hercules-Karte, diverse Software und Handbücher, Fr. 1'050.-. Monitor grün, Fr. 150.-. ☎ P: 01/840'03'15 abends

HP-75 C mit Math-ROM, Fr. 950.-. Pac-Screen-Video-Interface mit Monitor, Fr. 800.-. HP Video-Interface 82163B, Fr. 150.-. ☎ G: 052/45'18'94, P: 052/45'19'31

Olivetti M24, wegen Systemwechsel (wenig gebraucht), mit 2x360 KByte Floppy, 640 KByte RAM, monochrom Monitor, erweiterter Tastatur, 640 KByte Arbeitsspeicher, VP Fr. 2'400.-. Drucker Star SG-15, VP Fr. 500.- (A4 quer), ☎ G: 065/45'29'87, P: 065/76'26'68

Verschiedene Vorführgeräte für CAD und Desktop, z.B. 20 Zoll-Color-System für AutoCAD, HP-Darftpro A1 Plotter und anderes mehr, mit grossem Einschlag abzugeben. Schenk Data, 3304 Zuzwil, ☎ 031/96'05'06

IBM PC System/2, Modell 30, mit oder ohne Festplatte, mit Betriebssystem DOS 3.3 und Drucker, ganz neuwertig und in Garantie. VP Fr. 3'300.-, 35 % weniger als Neupreis (Fr. 5'130.-). Eventuell sind auch Programme verfügbar. ☎ 01/481'55'43 Mo bis Fr

Handheld Epson PX-8, wenig gebraucht, neuwertig, inkl. Software, NP Fr. 2'000.-, VP Fr. 700.-. ☎ 053/4'58'77

CBM 8032 System, mit Epson Drucker und CBM 8050 Floppy (8050 defekt), Fr. 300.-. Amerik. CPM Computer ohne Floppy für Selbstabholer gratis, ☎ 01/954'03'95 Russikon

Umweltschutzpapier verschiedene Endlosformate, auch mit Druck, Kopierpapier, Briefumschläge, Schul- und Büromaterial bei AP-Werkstatt, 9533 Kirchberg, ☎ 073/31'38'03

Computer-Börse

Spezialpreis für Abonnenten Fr. 20.–
Für Nichtabonnenten Fr. 70.–
In Brief oder auf PC 60-27181-0

Kleininserate werden nur gegen Vorauszahlung veröffentlicht!

Seagate Kit ST225, 20 MB, Fr. 495.-. ST 238, 30 MB, Fr. 645.-. ST251, 40 MB Fr. 845.-. Miniscr. 40 MB nur Harddisk, Fr. 595.-. Plus 10 % Luftpost. Bankcheck. Z. Egeresi, 5500 Coast Road, Santa Cruz, CA 95060 USA, Tx 550467

AT 286, 8/10/12 MHz, 5.25 Zoll Floppydisk, 20 MB Harddisk, MS-DOS 3.2, IMB RAM, 102 CH-Kys, 12 Zoll Monitor, Fr. 2'850.-. ☎ 031/92'30'78 oder 036/23'26'08

Epson PX-4, CP/M, Microkasetten und Digitalmultimetermodul, ROM Basic und Turbo-Pascal, Interface mit 2x6 programmierbaren Ein-/Ausgang, Brother HR-5 Drucker, alles für Fr. 1'300.-. ☎ G: 01/256'45'50, P: 01/954'03'95

Ein PDP 11/73 bestehend aus: BA 23 Floorstand Box, inkl. Power Supply; PDP 11/73, CPU mit 0.5 MB RAM und 1 asynchroner Datenleitung; RD 52, 31 MB Harddisk mit vermutlich RQDX-2 Controller; TQK 25-E, 60 MB Cartridge Tape; DHV 11, 8-Linien seriell Multiplexer. 4 VT 220 Terminal. 1 Rainbow mit 10 MB Harddisk und Doppelfloppy-Laufwerk. Software: Betriebssystem RT-11, Version 5.0, Multiuseraufsatz Share Plus, Version 2.2. 1 Printer LA-50, **dem Meistbietenden**. ☎ 053/8'12'22 (int. 2315)

Gesucht

Floppy-Laufwerk, 80 Track, soft-sektoriert zu Sorcerer. ☎ G: 071/22'35'39, P: 01/980'04'89