Zeitschrift: Schweizerische Bauzeitung
Herausgeber: Verlags-AG der akademischen technischen Vereine
Band: 91 (1973)
Heft: 5: Datentechnik: Geräte und Anwendung

Artikel: Tischrechner mit problemorientierter Sprache. Erster Teil: Allgemeines und der Tektronix-Rechner
Autor: Schilt, H.
DOI: https://doi.org/10.5169/seals-71786

91. Jahrgang Heft 5 SCHWEIZERISCHE BAUZEITUNG 1. Februar 1973
HERAUSGEGEBEN VON DER VERLAGS-AKTIENGESELLSCHAFT DER AKADEMISCHEN TECHNISCHEN VEREINE, 8021 ZÜRICH, POSTFACH 630

Tischrechner mit problemorientierter Sprache

DK 681.322

Von Prof. Dr. **H. Schilt**, Biel

In einem ersten Teil werden die Vorteile einer problemorientierten Programmiersprache im allgemeinen beschrieben. Eine Darstellung der Arbeitsweise des Tektronix-Tischrechners mit einigen Beispielen zeigt eine erste Verwirklichung einer solchen Sprache.

In einem zweiten Teil folgt die Beschreibung des neuen Modells (9820 A) der Firma Hewlett-Packard, das dank einer besondern Organisation ein noch einfacheres Programmieren erlaubt. Geeignete einfache Beispiele zeigen auch hier dem Leser, was man durch die Verwendung eines Tischrechners mit beschrifteter Ausgabe der Ergebnisse und problemorientierter Sprache bei der Auswertung von angewandten mathematischen Aufgaben an Arbeitszeit gewinnen und wohl oft deshalb auch auf einen Computer-Spezialisten verzichten kann.

Erster Teil: Allgemeines und der Tektronix-Rechner

1. Über das Rechnen mit Tischgeräten im allgemeinen

Die handelsüblichen Tischrechner sind meistens folgendermassen organisiert: Sie bestehen aus einem Tastenfeld für die Eingabe der Daten, einem relativ kleinen Rechenwerk und einigen Speichern. Die Ausgabe der Resultate geschieht entweder durch eine Anzeige mit Leuchtziffern, einem Drucker oder einer Schreibmaschine. Da die Tischrechner nicht mit einem Übersetzungsprogramm (Compiler) arbeiten, ist die Organisation des Rechenwerkes entscheidend für die Art der Rechenausführung und im wesentlichen auch für die Sprache, mit der das gegebene numerische Problem in die Maschine eingegeben werden muss.

Mehr oder weniger einheitlich ist das Aufrufen von Funktionswerten; diese sind in besondern Speichern (READ ONLY MEMORY) aufbewahrt. Das Argument muss *zuerst* eingetastet und unter Umständen noch in einen besondern Arbeitsspeicher übertragen werden; das Drücken der entsprechenden Funktionstaste (oder einer Kombination von Tasten) gibt dem Rechenwerk den Befehl, den Funktionswert zu berechnen und in der Anzeige oder in einem besondern Speicher bereitzustellen.

Operationen zwischen zwei Zahlen a und b, zum Beispiel $a + b$, $a - b$, $a \cdot b$, $a : b$, werden in verschiedener Weise ausgeführt. Meistens geschieht die Anweisung nach folgendem Schema (Schema A):

1. erste Zahl (a) eintasten,
2. diese Zahl in einen besondern Speicher versorgen,
3. zweite Zahl (b) eintasten,
4. Operationsbefehl geben.

Falls man eine Kettenrechnung $a + b \cdot c$ ausführen muss, ist bei derart organisierten Rechnern eine Umstellung der Reihenfolge unumgänglich:

b eintasten,
b in richtigen Speicher übertragen,
c eintasten,
Operation «Multiplikation» befehlen,
a eintasten,
Operation «Addition» befehlen,
Resultat ausgeben.

Für das Übertragen der Zahlen in die verschiedenen Speicher sind Transferbefehle notwendig. Wir nehmen an, die Zahlen $a, b, c \ldots$ seien von einer früheren Rechnung oder durch direkte Eingabe in je einem Speicher mit den Adressen A, B, C... gespeichert. Es ist ein Befehl nötig, der, um bei unserm Beispiel zu bleiben, die Zahl b aus dem Speicher B aufruft und in den Arbeitsspeicher Y einliest; wir nennen diesen Befehl B→Y.

Ein weiterer Befehl dient dazu, die Zahl c aus dem Speicher C in das zweite Arbeitsregister X zu rufen; er heisse C→X. Die Zahl a sei im Speicher A. Die Anweisung für unsere Rechnung ($b \cdot c + a =$) heisst nun:

B→Y C→X × A→X + =.

Das Resultat ist in Y. Wir nennen das Schema dieser Art: Schema A_1. Man erkennt, dass →X oder →X + je in eine einzige Anweisung zusammengefasst werden kann; →X × in × und →X + in +, so dass damit unsere Rechenanweisung für ($b \cdot c + a$) lautet:

B→Y C × A + =.

Wir nennen dies: Schema A_2.

Oft wird der Befehl → mit →Y, Y→, →X, X→ in *vier* neue Transferbefehle aufgeteilt, wodurch beim Programmieren auch einige Schritte eingespart werden können.

Die Befehlsschemen A_1 und A_2 sind offensichtlich nicht problemorientiert; man muss bei der Ausführung der Rechnung neben den Rechenoperationen noch zusätzliche Befehle geben, die wesentlich von der internen Konstruktion der Maschine abhängen. Man beachte, dass die Rechnung oft nur in einer Reihenfolge ausführbar ist, obwohl die Operationen kommutativ sind.

Komplizierter werden die Verhältnisse dann, wenn noch Funktionen ausgerechnet werden müssen: Es sei zum Beispiel der Ausdruck $\sqrt{a} + b \cdot \ln(1 + c)$ zu bilden. Auch hier ist eine Umstellung zweckmässig: $[\ln(1 + c)] \cdot b + \sqrt{a}$. Dazu sind folgende Befehle notwendig:

C→Y	c in Arbeitsspeicher Y holen
1 +	$1 + c$ in Speicher Y bilden
ln y	$\ln(1 + c)$ in Speicher Y erzeugen
B→X	b nach X holen
×	Inhalt von X mit Inhalt von Y multiplizieren und in Y speichern: (es sei $[\ln(1 + c)] \cdot b = d$)

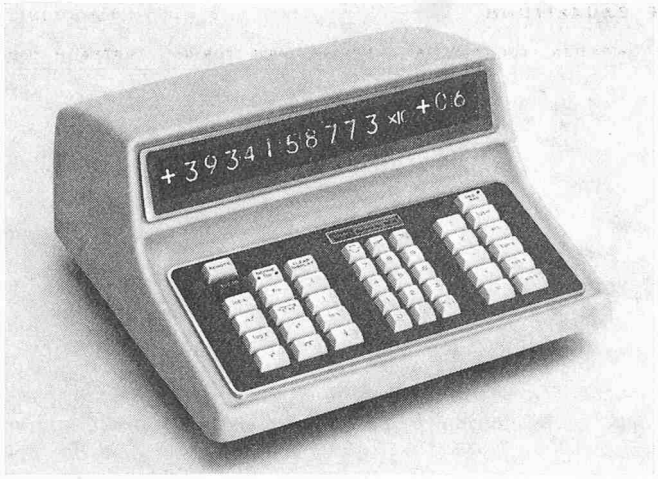

Bild 1. Das zentrale Gerät des Tischrechners Tektronix

Y→D	Zwischenresultat d in D speichern
A→Y	a nach Y holen
$\sqrt{y}$	$\sqrt{a}$ in Speicher Y erzeugen
D→X	d nach X holen
+	d zu $\sqrt{a}$ addieren
=	Resultat in Y anzeigen oder ausdrucken.

In einer Zeile geschrieben: nach Schema A_1:

C→Y ln B→X × Y→D A→ Y $\sqrt{y}$ D→X + =

nach Schema A_2:

C→Y ln B × Y→D A→Y $\sqrt{y}$ D + =

2. Die mathematische Schreibweise und der Tektronix-Rechner

Eine andere Organisation der Operationen und der Rechenspeicher wurde in dem Tischrechner *Tektronix* verwirklicht. Dafür sind zusätzliche Arbeitsspeicher notwendig, und die Transferbefehle zu diesen Speichern sind mit den Operationsbefehlen kombiniert. Für die Operationen 1. Stufe (Addition und Subtraktion) und 2. Stufe (Multiplikation und Division) sind mindestens sieben Arbeitsspeicher nötig: X, Y, Z, U_+, U_-, V., $V_:$. Diese sind für den Benützer nicht direkt zugänglich, er muss sich aber auch nicht um ihre Organisation kümmern. Für die Operationen der verschiedenen Stufen gilt nach Übereinkunft die Stufenregel: Die Operationen höherer Stufe haben vor den Operationen niederer Stufe den Vorrang,

zum Beispiel ist $a + b \cdot c - d = [a + (b \cdot c)] - d$

und $a \cdot b + c : d = (a \cdot b) + (c : d)$.

Wie die nachfolgenden Beispiele zeigen, wird beim Tektronix-Rechner diese Stufenregel auch beachtet; das heisst, die Eingabe kann so erfolgen wie die linken Seiten der soeben angeführten Beispiele, und die Maschine rechnet nach der Darstellung der rechten Seiten.

Die Rechnung $a + b \cdot c$ würde folgendermassen aussehen:

Tasten	Anzeige	
a	a	Eintasten a
+	0	Vorbereiten der Addition
b	b	Eintasten b
·	0	Vorbereiten der Multiplikation
c	c	Eintasten c
=		Ausführen der Mult. $b \cdot c$ und
	$a + b \cdot c$	*nachher* der Addition

Wir nennen dieses Rechenschema: Schema B.

Falls auf eine Operation eine solche gleicher Stufe oder eine niedrigerer Stufe folgt, so wird die vorbereitete Operation ausgeführt, damit wird erreicht, dass jeder Arbeitsspeicher U oder V in jedem Moment höchstens einfach belegt ist. Beispiel für $a + b \cdot c : d - e = r$.

Tasten	Anzeige	
a	a	a eintippen
+	0	Add. wird vorbereitet
b	b	b eintippen
·	0	Mult. wird vorbereitet
c	c	c eintippen
:	0	Mult. wird ausgeführt und Div.
d	d	vorbereitet
—		Div. wird ausgeführt und
	$a + (b \cdot c) : d$	Add. wird ausgeführt
		Subtr. wird vorbereitet
e	e	Subtr. wird ausgeführt und Resultat
=	r	angezeigt

Das Tastenfeld hat auch Klammern (und); damit können Ausdrücke der Form $(a + b) \cdot c$ ausgerechnet werden. Man kann mehrere Klammern benutzen:

...(.....)...(.....) oder ...(((.....)...)...)[1],

aber nicht verschachtelt: also nicht (...(....)...).

Man beachte, dass bei diesen Rechnungen das Programm mit der algebraischen Rechenanweisung identisch ist. Für Benützer, die nur eine elementare Kenntnis der algebraischen Symbolik besitzen, ist daher das Rechnen und das Programmieren mit dem Tektronix-Tischrechner unmittelbar möglich.

Ausser den *Speichern,* die den Rechenoperationen dienen und nicht direkt zugänglich sind, hat der Tektronix-Tischrechner noch 26 bzw. 100 Speicher. (Eine Einheit mit 26 Speichern kann leicht auf 100 Speicher ausgebaut werden.) Die Adressierung der Speicher ist sehr einfach: Taste K gefolgt von einer zweiziffrigen Zahl. z.B. K17, ruft den Inhalt des Speichers Nr. 17 in die Anzeige, das heisst in den Arbeitsspeicher X und kann dort so verwendet werden, wie eine eingetastete Zahl. Zugleich besteht eine leichtverständliche Möglichkeit für eine indirekte Adressierung. Es befinde sich z.B. im Speicher Nr. 17 die Zahl 9. Der Befehl KK17 ruft in diesem Fall den Inhalt des Speichers Nr. 9 in die Anzeige, denn mit K17 wird die Zahl 9 aufgerufen.

Die Speicher werden beim Aufrufen oder beim Drücken der Tasten CLEAR oder CLEAR DISPLAY *nicht* gelöscht. Erst das Abschalten der Maschine löscht die Speicher.

Das Einlesen einer Zahl in einen leeren oder gefüllten Speicher kann nur vom Arbeitsspeicher X (also von der angezeigten Zahl) her erfolgen, und zwar durch den einfachen Transferbefehl =; z.B. a = K05 speichert die Zahl a im Speicher Nr. 5, dabei bleibt die Zahl noch in der Anzeige, bis diese durch die Befehle CLEAR oder CLEAR DISPLAY oder durch einen Operationsbefehl gelöscht wird.

Der *Zahlenbereich* besteht aus 10 angezeigten Ziffern und 2 weiteren, nicht sichtbaren, aber ausgerechneten Ziffern mal 10 hoch einen zweistelligen Exponenten. Sowohl Zahl als auch Exponent können positiv und negativ sein; Fixkomma und auch Gleitkommadarstellung sind möglich.

Folgende *Funktionen* sind durch Tasten sofort zugänglich: x^2, e^x, sin x, cos x, tan x, Sinh x, Cosh x, Tanh x, davon alle Umkehrfunktionen, ausserdem $1/x$, log x und $[x]$ (= ganzzahliger Teil von x). Die Zahl x muss im Anzeigespeicher (X) sein, hierauf drückt man die gewünschte Funktionstaste, und in der Anzeige wird x in den zugeordneten Funktionswert

[1]) Dabei können die drei Klammern (((durch eine einzige ersetzt werden.

umgewandelt. Der Vorteil dieser Organisation geht aus der Programmierung der schon oben aufgeführten Rechnung $\sqrt{a} + b \cdot \ln(1 + c)$ deutlich hervor; dabei spielt es keine Rolle, ob die Zahlen a, b, c gespeichert sind oder nicht; man hat an Stelle einer einzutippenden Zahl nur den entsprechenden Speicher aufzurufen.

Unser Programm lautet entweder $a\ \sqrt{x} + b \cdot (1 + c)\ \ln x =$ oder mit Speicher K01 $\sqrt{x}$ + K02 · (1 + K03) $\ln x$ =; dabei wurde vorausgesetzt, dass die Zahlen a in K01, b in K02 und c in K03 gespeichert seien.

3. Das Arbeiten mit dem Tektronix-Rechner

3.1. Beispiele für das zentrale Gerät allein

Das Äussere des Tektronix-Tischrechners besteht aus einem zentralen Rechengerät, an dem ein Kartenleser, ein Programmer und ein Drucker anschliessbar sind.

Das *zentrale Gerät* (siehe Bild 1) besitzt neben den besprochenen Operationen 1. und 2. Stufe auch zwei Operationen 3. Stufe, nämlich mit den Tasten

$$|x|^y \text{ und } \sqrt{x^2 + y^2}$$

Ihre Verwendung geht nach dem gleichen Schema wie bei den Operationen 1. und 2. Stufe vor sich, nämlich:

$$a\ |x|^y\ b = |a|^b; \quad a\ \sqrt{x^2 + y^2}\ b = \sqrt{a^2 + b^2}$$

Beide Operationen können sinnvoll iteriert werden.

$$a\ |x|^y\ b\ |x|^y\ c = (|a|^b)^c \text{ und } a\ \sqrt{x^2 + y^2}\ b\ \sqrt{x^2 + y^2}\ c = \sqrt{a^2 + b^2 + c^2}$$

(Also eine einfache Art, die Norm eines mehrdimensionalen Vektors zu erhalten.)

Die Operationen 3. Stufe verlangen ausser den besprochenen weitere Arbeitsspeicher, auf die hier nicht eingegangen werden soll. Ebenfalls gilt für die Operation 3. Stufe auch die Stufenregel: Wenn keine Klammern es anders verlangen, werden mit dem Eingeben eines neuen Operationsbefehls nur diejenigen vorbereiteten Operationen ausgeführt, die von gleicher oder höherer Stufe sind; zum Beispiel:

$$a \cdot b\ |x|^y\ c = a \cdot (b^c)$$

$$a\ \sqrt{x^2 + y^2}\ b\ |x|^y\ c = (\sqrt{a^2 + b^2})^c$$

$$a\ |x|^y\ b\ \sqrt{x^2 + y^2}\ c = \sqrt{(a^b)^2 + c^2}$$

$$a + b\ |x|^y\ c = a + |b|^c$$

Das zentrale Gerät ist ohne Programmer einfach, aber trotzdem wirksam programmierbar, nämlich mit Hilfe der Tasten DEF f(x) und f(x). Es stehen dafür 256 Programmschritte zur Verfügung. Falls die Taste DEF f(x) gedrückt ist (kleine Lämpchen markieren diesen Zustand), so wird die nachfolgende eingegebene Rechnung ausgeführt *und* programmiert. Das nochmalige Drücken der Taste DEF f(x) schliesst das Programm ab. Werden die entsprechenden Speicher neu gefüllt und die Taste f(x) gedrückt, wird die Rechnung nochmals mit den neuen Werten ausgeführt.

Interessiert man sich auch für Zwischenwerte der Rechnung, so kann man diese beim Programmieren zusätzlichen Speichern zuordnen. Nach dem Berechnen des Resultates kann man diese Speicher aufrufen, mit den Werten Rechnungsoperationen ausführen, ohne dass das Programm verändert wird. Es besteht die Möglichkeit, in solchen Programmen auch *Entscheidungen* und indirekte Adressierungen einzubauen, und zwar auf folgende Art:

a^0 ist bekanntlich 1, wenn $a \neq 0$, und unbestimmt, wenn $a = 0$ ist.

$0\ |x|^y\ 0$ ist jedoch im Tektronix-Tischrechner als 0 definiert; ferner ist $a\ |x|^y\ 1 = |a|$.

Somit ist z. B. $(a\ |x|^y\ 1 - a)\ |x|^y\ 0 = \begin{cases} 0, \text{ wenn } a \geqslant 0 \\ 1, \text{ wenn } a < 0. \end{cases}$

Das Tektronix-Rechengerät besitzt keine Taste TO POLAR. Durch folgendes Programm, das überall auch als Unterprogramm eingebaut werden kann, ist der Übergang von rechtwinkligen zu Polarkoordinaten möglich.

Es sei x in K01 und y in K02 gespeichert.

DEF f(x) K01 $\sqrt{x^2 + y^2}$ K02 = K03 (K02 : K01) arc tan = $\pi \cdot$ (K01 $|x|^y$ 1 − K01) $|x|^y$ 0) = K04 DEF f(x)

In K03 findet man den Betrag des Radius-Vektors und in K04 den Bogen im Intervall von $-\pi/2$ bis $3\pi/2$. Umstellung auf 360°-Teilung ist durch die Taste RAD DEG möglich.

Unerlaubte Operationen (Wurzel aus einer neg. Zahl, Dividieren durch null, arc sin von einer Zahl > 1 oder < −1) werden durch Blinken angezeigt. Dabei wird weder das Programm noch eine Rechnung gestört. Zum Beispiel wird $\sqrt{-4}$ durch 2 mit Blinken angezeigt, und wenn im Programm für TO POLAR K01 = 0 ist, blinkt die Anzeige, aber der Winkelwert wird richtig und mit richtigem Vorzeichen angegeben ($\pi/2$ bzw. $-\pi/2$). Das Blinken kann durch Drücken der roten Taste CLEAR gelöscht werden, dabei wird, ausser den Arbeitsspeichern, weder ein Programm verändert noch ein Speicher K... gelöscht.

Die zwei letzten Beispiele: $\sqrt{a} + b\ \ln(1 + c) = d$ und $r = \sqrt{x^2 + y^2}$ und $\varphi = \arctan y/x + \pi\ (|x| - x)^0$ eignen sich gut für die Benützung einer *indirekten Adressierung*.

Die drei Speicher K01, K02 und K03 sollen nacheinander mit den Zahlen a, b, und c gefüllt werden. Das kann direkt geschehen: a = K01, b = K02 usw. Diese Art der Eingabe hat zwei Nachteile:

a) Nach dem Eintippen der Zahl muss man noch vier Tasten drücken = K 0 1;
b) wenn viele Konstanten einzugeben sind (Koeffizienten von Matrizen), kann man sich leicht verzählen.

Man kann das Programm nun so ergänzen, dass nach jeder Eingabe derjenige Speicher zur Aufnahme bereit ist, in den die nächste Zahl gespeichert werden soll. Ausserdem kann man es so einrichten, dass nach der Eingabe jeder Konstanten der nächste Speicher gelöscht, also zur Aufnahme der nächsten Zahl vorbereitet ist. Dieser Speicher wird am Schluss des Programms nochmals aufgerufen; solange noch Zahlen einzugeben sind, ergibt der Befehl f(x) daher null. Das eigentliche Rechenprogramm wird nun so geplant, dass das Resultat in denjenigen Speicher eingelesen wird, der auf den letzten, mit Konstanten gefüllten Speicher folgt. Nachdem die letzte Konstante eingetippt ist und f(x) gedrückt wird, erscheint das Resultat. Interessiert man sich noch für ein zweites Resultat aus der gleichen Rechnung (Beispiel TO POLAR), so speichert man dieses in den darauffolgenden Speicher und erhält das zweite Resultat nach nochmaligem Drücken der Taste f(x). Das Programm würde folgendermassen lauten: Man beginne mit 0 = K00 und definiere DEF f(x) = KK00 K00 + 1 = K00 CD = KK00 folgt Programm wie oben, dann KK00 DEF f(x). Nach dem Programmieren steht die Zahl 1 in K00. KK00 heisst jetzt also K01, und der eingetippte Wert a wird durch Drücken der Taste f(x) gemäss dem programmierten Befehl = KK00 in den Speicher K01 eingelesen. Ferner wird die Zahl 2 in K00 erzeugt (durch die Befehle K00 + 1 = K00), dazu wird durch den Befehl CD (Clear Display) = KK00 der Speicher K02 gelöscht. Hierauf soll die Zahl b eingetippt werden, mit f(x) wird sie in den Speicher K02 gelesen usw. Die Rechnung des Programms wird jedes-

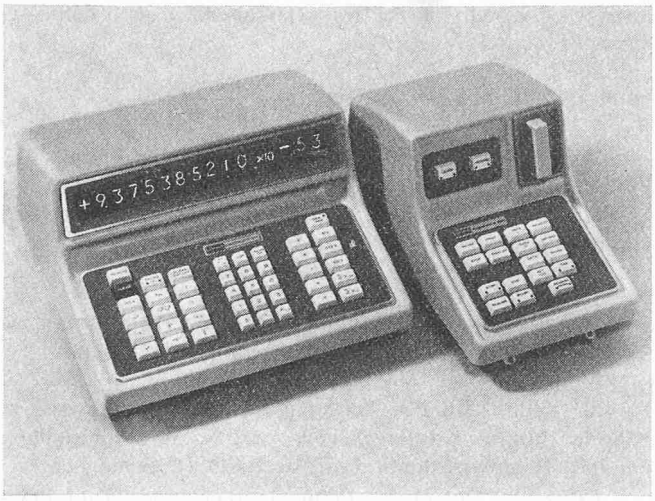

Bild 2. Der Tektronix-Tischrechner mit dem zugehörigen Programmer

mal durchgerechnet; da noch nicht alle Konstanten in der Maschine sind, sind die Resultate unbrauchbar; diese werden aber auch nicht angezeigt. Erst nach Eingabe der letzten Konstanten erscheint der (erste) Wert des Resultates. Kurz zusammengefasst sieht die Ein- und Ausgabe für das erste Beispiel so wie auf Tabelle 1 links angegeben aus.

Im zweiten Beispiel (TO POLAR) sind zwei Resultate gefragt. Die Bedienung und die Anzeige der Maschine erfolgen wie auf Tabelle 1 rechts angegeben.

An die Zentraleinheit kann ein *Lochkartenleser* angeschlossen werden. Die Lochkarten sind im Oktal-Code nach einem bestimmten Schlüssel für die einzelnen Maschinenbefehle zu lochen, was man sehr leicht von Hand mit einem einfachen, von der IBM herausgegebenen Gerät tun kann. Ohne umständliches Eintippen kann man jedes Programm für DEF f(x) in kurzer Zeit einlesen. Wie beim Eintippen von Hand, so wird auch beim Einlesen mit dem Kartenleser das Programm einmal durchgerechnet und kann somit durch vorheriges Eingeben geeigneter Zahlen kontrolliert werden.

3.2. Beispiele für das zentrale Gerät mit Programmer

Der Tektronix-Tischrechner kann durch einen *Programmer* (Bild 2) ergänzt werden. Damit können Rechenprogramme bis zu 512 Schritte ohne Unterbruch programmiert werden, ausserdem stehen noch die 256 Programmschritte in DEF f(x) zur Verfügung. 10 Programme dieser Länge können in einer Bandkassette gespeichert werden.

Das Programmieren ist ebenso einfach, das heisst problemorientiert, wie mit Hilfe der Taste DEF f(x). Bevor man die zu programmierenden Schritte eingibt, hat man nur die

Tabelle 1

Erstes Beispiel		Zweites Beispiel	
Eintippen	Anzeige	Eintippen	Anzeige
a	a	1 = K00	1
f(x)	0	x	x
b	b	f(x)	0
f(x)	0	y	y
c	c	f(x)	r
f(x)	Resultat	f(x)	φ
1 = K00	1	1 = K00	1
CD	0	CD	0
Die Maschine ist bereit für die Eingabe neuer Werte a, b, c usw.		Die Maschine ist bereit für neue Eingabe.	

Taste LEARN zu drücken. Das Programm wird durch END LEARN abgeschlossen. Es gibt vier nicht programmierbare Befehle: DISPLAY, LIST, STEP, EDIT. Diese können während des Programmierens benutzt werden, z.B. DISPLAY schaltet die Anzeige von den Rechenoperationen um auf Zifferngruppen und umgekehrt; die vordersten 3 Ziffern geben die Nummer des nächsten Programmschrittes an, die mittleren Ziffern den Code, der zu diesem Schritt gehört, und die letzten Ziffern die Nummer des Blockes auf dem Band. LIST befiehlt das Ausdrucken des Programmes, und mit STEP kann man jeden Schritt einzeln ausführen, und zwar nach Wunsch, gerechnet oder als Programmschritt angezeigt. LEARN EDIT schiebt den Rest des Programmes um einen Schritt nach hinten, so dass man Korrekturen im Programm einfügen kann, ohne das ganze Programm nochmals neu einzugeben.

Unbedingte Sprünge sind durch GOTO, gefolgt von 3 Ziffern, zu befehlen; die drei Ziffern geben jenen Programmschritt an, bei dem das Programm fortgesetzt werden soll.

Bedingte Sprungbefehle gibt es fünf (bzw. sechs): IF DATA +, IF DATA 0, IF DATA –, IF RANGE, IF FLAG, jeweils gefolgt von 3 Ziffern. IF DATA + bedeutet z.B., dass das Programm nur dann nach dem durch die 3 Ziffern angegebenen Programmschritt springen soll, wenn die Anzeige positiv ist. Ähnlich verhalten sich die andern Sprungbefehle. Bei IF FLAG wird das Programm verzweigt und SET FLAG gelöscht, wenn vorher SET FLAG gesetzt wurde.

Neu ist IF RANGE: Dieser bedingte Sprung wird ausgeführt, wenn die Anzeige blinkt. Mit diesem Befehl können z.B. sowohl reelle als auch komplexe Lösungen im Programm sinngemäss weiterverfolgt werden. Der sechste bedingte Sprungbefehl INDIRECT bezieht sich nicht auf die nachfolgenden 3 Ziffern im Programm, sondern auf die 3 letzten Ziffern in der Anzeige, die dem Befehl INDIRECT vorangeht. Diese 3 Ziffern geben hier die Sprungadresse an; damit können beliebig viele, voneinander abhängige Verzweigungen befohlen werden.

Auf einfache Weise lassen sich Zahlentabellen in den Programmer einlesen, auf das Band speichern und bei Bedarf in die Speicher der Zentraleinheit übertragen. Dabei kann man nach folgendem Programm vorgehen:

DEF f(x) CD DEF f(x)
LEARN a_1 f(x) a_2 f(x) .....a_n f(x) LEARN

Damit sind die Zahlen a_1 a_2 ....a_n in den Programmer eingelesen.

Der Befehl CD (= CLEAR DISPLAY) in f(x) löscht jeweils die Anzeige, so dass die Maschine für die Eingabe der nächsten Zahl aufnahmebereit ist. Die Taste RECORD 3 liest das Programm mit den Zahlen auf das Band in den Block Nr. 3.

Zum späteren Benutzen der Zahlen dienen folgende Programmschritte: SEARCH PROGR. 3. Mit diesem Befehl werden die Zahlen vom Band wieder in den Programmer gelesen. Falls man sie nun z.B. in die Speicher Nr. 11 bis Nr. (11 + n) eingeben will, befiehlt man 10 = K00 und DEF f(x) = KK00 K00 + 1 = K00 CD DEF f(x). START bewirkt, dank dem gespeicherten f(x) und der DEF f(x), das Einlesen der Zahlen a_1 in K11, a_2 in K12 usw. bis a_n in K(11 + n). Das Einlesen der Konstanten in die Speicher kann schon beim Programmieren erfolgen, wenn man die dort definierte Funktion f(x) durch die eben beschriebene ersetzt.

Tasten, deren Befehle vom vorhergehenden Zustand der Maschine abhängen, sind mit kleinen Lämpchen versehen, die bei einem bestimmten Zustand aufleuchten und wieder löschen, wenn der Zustand geändert wird. Solche Tasten sind: LEARN, RECORD, SEARCH PROGRAMM, START, STOP, SET FLAG, DEF f(x). Die Umschalttaste DEG RAD

hat zwei Lämpchen, von denen immer nur eines leuchtet, nämlich dasjenige, das den jeweiligen Zustand der Taste angibt. Das Drücken der Taste DEG RAD wirkt nur auf die in der Anzeige befindliche Zahl, das heisst, die Anzeige wird von Radianten in Grad umgerechnet oder umgekehrt. Wenn die Taste vor dem Ausrechnen eines Programmes in der Stellung DEG ist, bewirkt das zusammen mit den Befehlen der Tasten sin$x$, cos$x$ oder tan$x$ (bzw. arc$x$) die Umrechnung der Argumente von Grad in Radianten (bzw. der Funktionswerte in Grad). CLEAR befiehlt ausser dem Löschen der Anzeige immer auch das Umschalten auf RAD.

Falls eine Adresse bei den bedingten Sprungbefehlen oder bei SEARCH PROGR. oder RECORD nicht vollständig ist, leuchtet auch eine Lampe auf. Alle diese Lichter sind sehr angenehm, und ihre Beachtung erspart dem Benützer manche Fehlmanipulation.

Folgendes kleine Beispiel möge die Eleganz der Programmierung mit dem Tektronix-Tischrechner demonstrieren. Es sei die quadratische Gleichung

$$x^2 + rx + r^2 \cos\alpha = 0$$

zu lösen. Der Gang der Rechnung ist auf Bild 3 angegeben.

Nach Eingabe der Konstanten in K01 und in K02 und Drücken der Taste START rechnet die Maschine unmittelbar das Resultat x_1, das in der Anzeige erscheint, CONTINUE befiehlt die nächste Rechnung mit x_2 in der Anzeige und Rückkehr zum Anfang. Falls die Anzeige blinkt, bedeutet das erste Resultat absoluter Betrag von x_1 und das zweite Resultat gibt den Polarwinkel der komplexen Zahl x_1 an. CLEAR löscht das Blinken, und die Maschine steht für eine neue Eingabe der Konstanten bereit.

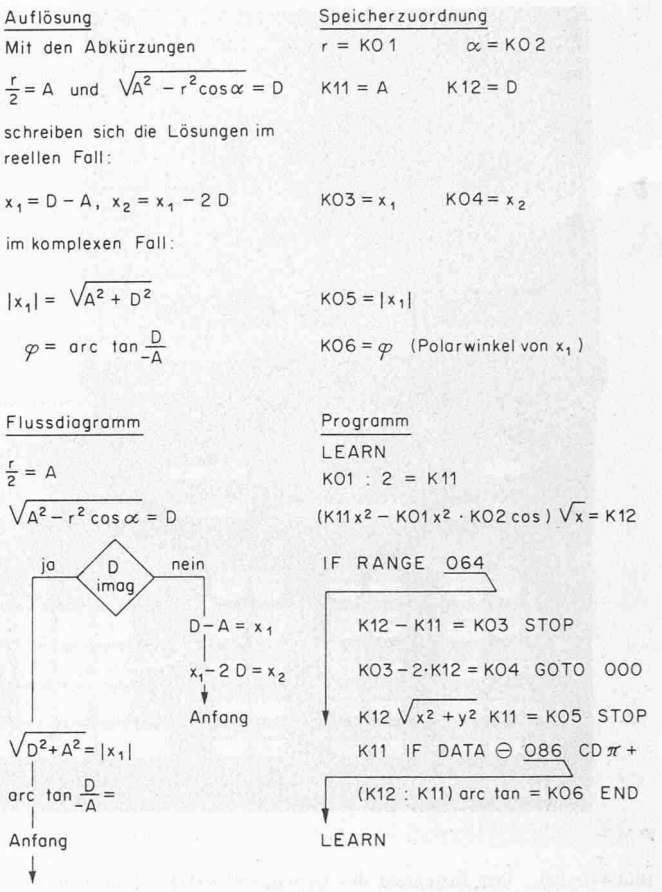

Bild 3. Rechnungsgang zum Beispiel im Text

Zweiter Teil: Der Rechner Hewlett-Packard 9820

4. Beschreibung des Gerätes

Von der Firma Hewlett-Packard wir nun auch mit dem Modell 9820 (kurz HP 20 genannt) ein Tischrechner angeboten, der nach einer problemorientierten Sprache organisiert ist. Eine Beschreibung dieses Modells ist daher hier angebracht. Die Grundausrüstung der HP 20 enthält eine Tastatur, eine Anzeige, ein Druckwerk, einen Magnetkartenleser, einen Speicher und zusätzlichen Platz für drei einschiebbare «Read Only Memories» (ROM genannt).

4.1. Die Tastatur

Das Tastenfeld ist in fünf Blöcke aufgeteilt, Bild 4. Die linken drei enthalten Buchstaben und Sonderzeichen; sie dienen zugleich dem Aufrufen der ROM. Der vierte Block enthält Ziffern, Operationsbefehle und Satzzeichen; im fünften Block sind die Tasten für logische Operatoren, für Sprunganweisungen und zum Aufrufen der Register sowie drei Befehlstasten für die Arbeitsart des Rechners zusammengefasst. Oberhalb dieser Blöcke gibt es eine Reihe von Tasten, die sich auf die Anzeige, die Ein- und Ausgabe und den Magnetkartenleser beziehen und dafür eine klare Beschriftung tragen.

4.2. Die Anzeige

In der Anzeige können auf einmal höchstens 16 Zeichen erscheinen. Ein Zeichen ist aus Lichtpunkten zusammengesetzt, die einer Matrix von 5 mal 7 Licht emittierenden Dioden (LED) entnommen sind. Die Schrift ist 7,5 mm hoch und sehr gut lesbar. Beim einmaligen Drücken der Taste BACK wird immer das letzte sichtbare Zeichen gelöscht; durch Bedienen der Taste FORWARD kommt das Zeichen wieder in die Anzeige. Wird eine Zeile mit mehr als 16 Zeichen eingegeben, so verschwinden die vordersten Zeichen aus der Anzeige, ohne dass deswegen im Innern etwas von der Information verloren geht; im übrigen können die nicht mehr sichtbaren Zeichen mit Hilfe der Taste BACK wieder in die Anzeige geholt werden.

4.3. Der Drucker

Die Maschine druckt auf einen 57 mm breiten Papierstreifen mit Buchstaben und Ziffern. Diese bestehen ebenfalls aus Punkten, die einer 5 mal 7 Matrix entnommen sind. Die Schrift ist 2,7 mm hoch und kann bequem und ohne Missverständnisse gelesen werden. Der Drucker druckt alles, was der Benützer wünscht. Es seien z.B. die Register A, B, R7 mit gegebenen Werten gefüllt; mit den Befehlen PRINT A, B, R7; EXECUTE werden diese Werte auf je einer Zeile gedruckt. Da das ganze Alphabet zur Verfügung steht, kann auch ein kleiner Text gedruckt werden; z.B. wird durch PRINT «ERGEBNIS»; EXECUTE das Wort ERGEBNIS gedruckt. Wenn man das Resultat einer Rechnung im Register X gespeichert hat, ihm aber den Namen Φ geben möchte, so wird durch PRINT «PHI =», X; EXECUTE der Name PHI = am Anfang einer Zeile und der Wert im Register X am Ende der nächsten Zeile gedruckt. Bild 5 zeigt den Streifen für das unter Abschnitt 6 angeführte Beispiel.

Das Anzeigen oder Drucken von Zahlen kann in «Fixed Point n» (FXD n) oder in «Floating Point n» (FLT n) befohlen werden. Z.B. würde die Zahl 100 π in FXD 5 als 314.15927 und in FLT 4 als 3.1416 E 02 erscheinen. Die Zahlen werden in beiden Fällen gerundet, für die Rechnungen jedoch werden immer 12 Ziffern benutzt und so gespeichert.

4.4 Die Magnetkarten

Es gibt zwei verschiedene Magnetkarten, die sich nur in der Länge und damit auch in der Kapazität unterscheiden; sie