Programmierbarer Taschenrechner

CASIO FX-700P

Bedienungsanleitung

Ⓑ (独)

Wir möchten uns bei dieser Gelegenheit dafür bedanken, daß Sie sich für diesen programmierbaren Rechner entschieden haben. Es handelt sich dabei um ein Präzisionsinstrument, das mit Mikro-Elektronik bestückt ist und auch extrem komplexe Rechenabläufe zuläßt. Das wichtigste Merkmal dieses programmierbaren Rechners liegt darin, daß die Programmiersprache BASIC verwendet wird, mit deren Hilfe alle Rechenprobleme einfach programmiert werden können. Die Bedienung ist einfach und sollte auch für den Anfänger keine Probleme verursachen. Auch das Programmieren zeichnet sich durch Einfachheit aus, werden doch alle Befehle über einzelne Tasten eingegeben.

Mit diesem Rechner können die folgenden Rechenverfahren eingesetzt werden:
1. Manuelles Rechnen
2. Programmrechnungen

Dieser Rechner kann nicht nur für Programmabläufe (gleich wie bei einem Computer) eingesetzt werden, sondern gewährleistet auch einfachste Bedienung als technisch/wissenschaftlicher Rechner.

Inhaltsverzeichnis

In dieser Bedienungsanleitung sind die Funktionen und Bedienungs vorgänge des Rechners beschrieben. Bitte lesen Sie daher diese Anleitung aufmerksam durch und machen Sie sich mit allen Funktionen vollständig vertraut, um durch richtige Bedienung jahrelangen und problemlosen Betrieb sicherzustellen.

Vor der Verwendung

Dieser Rechner beruht auf fortschrittlichster Technologie und ist mit elektronischen Präzisionstellen bestückt. Strikte Qualitätskontrollen und vielseitige Inspektionen stellen hohe Zuverlässigkeit sicher. Bitte halten Sie die nachfolgend aufgeführten Vorsichtsmaßnahmen ein, um Betriebsstörungen zu vermeiden.

■ Vorsichtsmaßnahmen

- Dieser Rechner besteht aus elektronischen Präzisionsteilen. Den Rechner daher niemals selbst zerlegen. Harte Stöße und Erschütterungen vermeiden. Plötzliche Temperaturschwankungen, übermäßige Feuchtigkeit, Wärme und Staub müssen ebenfalls vermieden werden. Bei zu niedrigen Temperaturen könnte sich die Anzeigegeschwindigkeit verlangsamen bzw. die Anzeige vollständig verschwinden. Sobald der Rechner aber wieder auf Normaltemperatur gebracht wurde, sollte die Anzeige wieder erscheinen.
- Mit diesem Rechner darf nur das dafür angebotene Sonderzubehör verwendet werden.
- Während der Rechner komplexe Rechenvorgänge ausführt, erscheint das Symbol "—" in der Sichtanzeige; während dieser Zeitspanne eingegebene Befehle haben keine Wirkung. Die Tasten daher immer erst dann betätigen, wenn das Ergebnis der vorhergehenden Rechnung angezeigt wird.
- Auch wenn der Rechner nicht häufig verwendet wird, sollten die Batterien alle zwei Jahre erneuert werden. Erschöpfte Batterien sofort aus dem Rechner entfernen, da dieses sonst auslaufen und den Rechner beschädigen könnten.
- Für das Reinigen des Rechners ist ein weicher und trockener Lappen bzw. ein in milder Seifenwasserlösung angefeuchtetes (nicht nasses!) Tuch zu verwenden. Niemals chemische Reinigungsmittel, Farbverdünner oder Benzin verwenden.
- Falls es zu einer Störung des Rechners kommen sollte, wenden Sie sich bitte an Ihren Fachhändler oder an einen Kundendienst.
- Bevor Sie aber den Rechner zur Wartung bringen, kontrollieren Sie bitte die Stromversorgung und die Bedienungsvorgänge, da viele der vermeintlichen Probleme auf Bedienungsfehler zurückzuführen sind.

■ Stromversorgung und Austausch der Batterien

Dieser Computer arbeitet mit zwei Lithium-Batterien (CR2032). Wenn bei vollständig aufgedrehtem Kontrastregler (siehe Seite 8 die Anzeige nur schwach erscheint, sind die Batterien erschöpft und müssen erneuert werden. Auch wenn der Computer problemlos arbeitet, die Batterien alle zwei Jahre erneuern.

- **Austausch der Batterien**

(1) Den Stromschalter abschalten (OFF), die beiden Befestigungsschrauben der Rückwand aufdrehen und die Rückwand des Computers abnehmen.

(2) An Ⓐ drücken und den Batteriefachdeckel IN Pfeilrichtung abschieben.

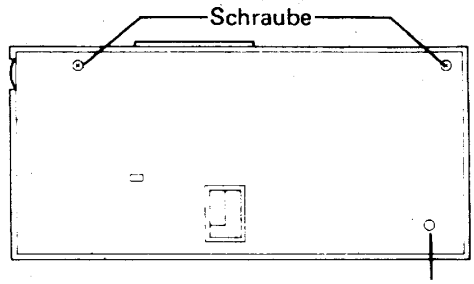
(3) Die leeren Batterien entfernen.
(Dazu den Computer mit den Batterien nach unten gerichtet halten und leicht gegen das Gehäuse schlagen.)

(4) Die neuen Batterien mit einem trockenen Lappen gründlich abreiben und die Batterien danach mit der positiven ⊕ Seite nach oben gerichtet einsetzen.

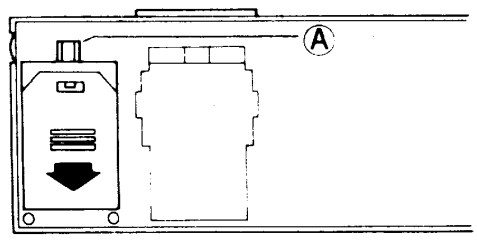
(5) Die Batterien mit dem Batteriefachdeckel hineindrücken und den Batteriefachdeckel zuschieben.

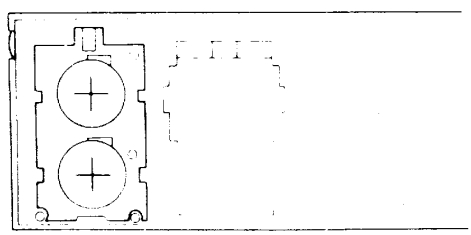
(6) Die Rückwand anbringen und mit den beiden Schrauben sichern. Danach den Stromschalter einschalten und den Initialisierungsknopf (ALL RESET) mit einem spitzen Gegenstand drücken.

* Immer beide Batterien gleichzeitig austauschen.
* Die alten Batterien niemals ins Feuer werfen, da diese ggfls. explodieren könnten.
* Beim Einsetzen der Batterien die richtige Polung (⊕ und ⊖) beachten, da bei falscher Polarität der Computer nicht arbeitet.



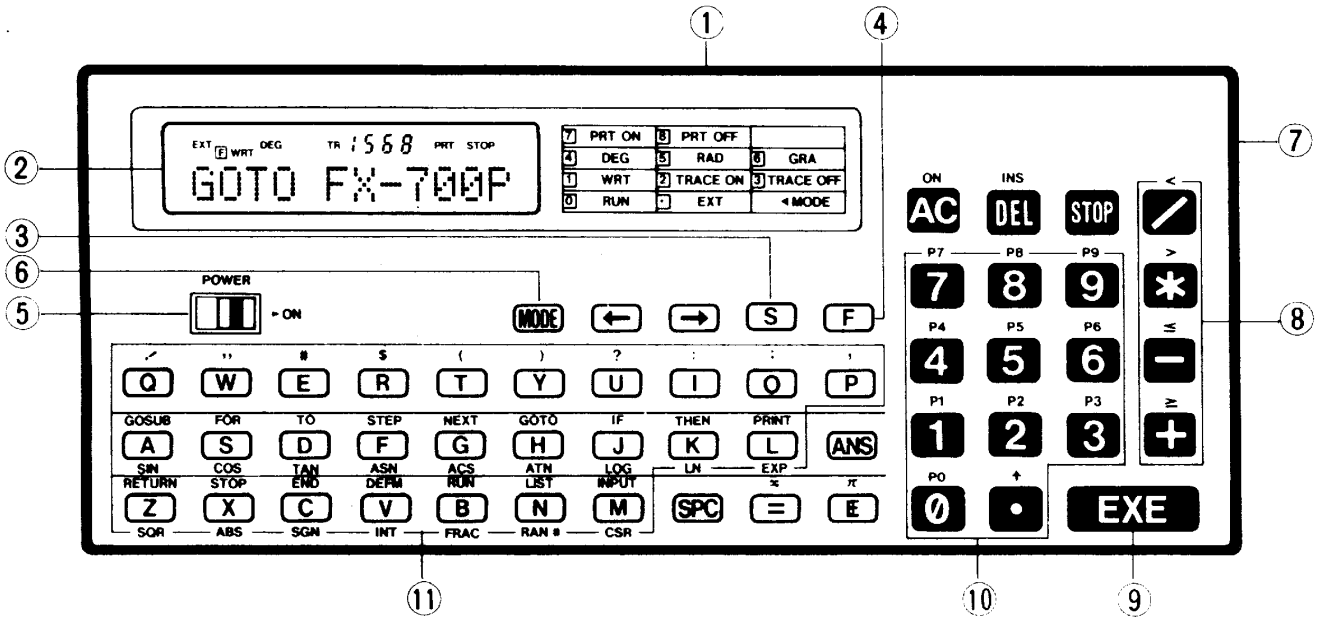
Initialisierungsknopf (ALL RESET)
(Nach dem Austauschen der Batterien mit einem spitzen Gegenstand drücken.)





Kapitel 1

Bedienungselemente und ihre Funktion



① Adapter-Anschluß
② Sichtanzeigefenster
③ Umschalttaste
④ Funktionstaste
⑤ Stromschalter
⑥ Betriebsartentaste
⑦ Kontrastregler für Sichtanzeige
⑧ Rechenbefehltasten
⑨ Ausführungstaste
⑩ Zifferneingabetasten und Dezimalpunkttaste
⑪ Buchstabentasten

1-1 Bezeichnung der einzelnen Tasten

Jede Taste weist drei verschiedene Funktionen auf.
Die entsprechende Taste direkt drücken, um die auf der Taste angegebene Funktion durchzuführen. Soll die über bzw. unter der Taste aufgeführte Funktion durchgeführt werden, dann ist die entsprechende Taste nach der [S] bzw. [F] Taste zu drücken.

Beispiel

GOSUB ..... Umschalten der Betriebsart
[A] ..... Direkte Betriebsart
SIN ..... Funktions-Betriebsart

[S] Umschalttaste (mit Symbol [S] bezeichnet)

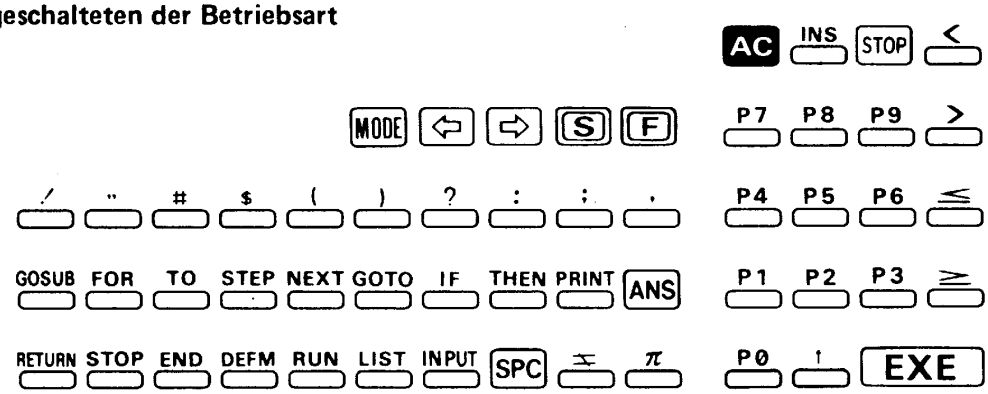
Durch Drücken dieser Taste wird die Betriebsart umgeschaltet (das Symbol "[S]" wird angezeigt), so daß die über den einzelnen Tasten aufgedruckten Funktionen verwendet werden können.

[F] Funktionstaste (mit Symbol [F]) gekennzeichnet)

Durch Drücken dieser Taste wird auf die Funktions-Betriebsart geschaltet (das Symbol "[F]" wird angezeigt), so daß die unter den einzelnen Tasten aufgedruckten Funktionen verwendet werden können.

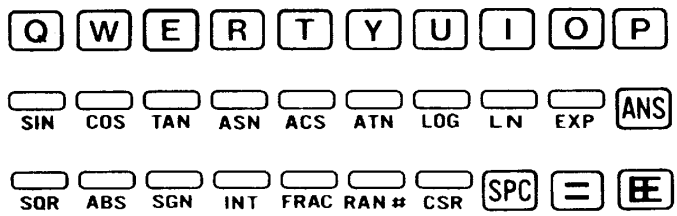
Tastenbetätigung bei Direktbetrieb

Tastenbetätigung in der umgeschalteten der Betriebsart



* In der umgeschalteten der Betriebsart dienen die Buchstabentasten als Anweisungstasten, wogegen mit Hilfe der Zifferneingabetasten der jeweilige Programmbereich (Programmspeicher) bezeichnet wird.

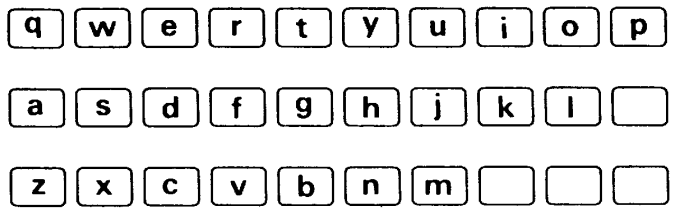
Tastenbetätigung in der Funktions-Betriebsart



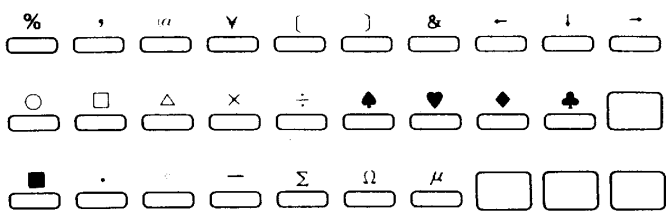
* In der Funktions-Betriebsart werden die Buchstabentasten zu Eintasten-Funktionstasten.

- In der zusätzlichen Erweiterungsfunktion (die Tastenfolge MODE • drücken, wodurch "EXT" angezeigt wird) können mit Hilfe der Buchstabentasten in der Direktbetrieb die Kleinbuchstaben und bei umgeschalteter Betriebsart verschiedene Spezialsymbole eingetastet werden.

Direktbetrieb mit erweiterter Betriebsart



Erweiterungsfunktion der umschalten der Betriebsart



* In der erweiterten Betriebsart wird durch Drücken einer Buchstabentaste und darauffolgende Betätigung der F Taste (Funktionstaste) der entsprechende Großbuchstabe angezeigt.

MODE Betriebsartentaste

Diese Taste in Verbindung mit der • bzw. der 0 bis 8 Taste drücken, um die Betriebsart des Computers bzw. das Argument einzustellen.

MODE • .... "EXT" wird angezeigt, um die Erweiterungsfunktion anzuzeigen, in der Kleinbuchstaben und Spezialsymbole eingetastet werden können. Um die Erweiterungsfunktion wieder freizugeben, die Tasten MODE • nochmals betätigen.

MODE 0 .... "RUN" wird angezeigt und manuelle Rechnungen bzw. Programme können ausgeführt werden.

MODE 1 .... "WRT" wird angezeigt und Programme können geschrieben, geprüft bzw. redigiert werden.

MODE 2 .... "TR" wird angezeigt, worauf ein Prüfprogramm durchgeführt werden kann (Einzelheiten siehe auf Seite 35).
MODE 3 .... Das "TR" Symbol wird gelöscht, wobei die Programmprüffunktion freigegeben wird.
MODE 4 .... "DEG" wird angezeigt, so daß das Argument in Altgrad eingegeben werden muß.
MODE 5 .... "RAD" wird angezeigt, so daß das Argument im Bogenmaß eingegeben werden muß.
MODE 6 .... "GRA" wird angezeigt, so daß das Argument in Neugrad eingegeben werden muß.
MODE 7 .... "PRT" wird angezeigt, so daß bei angeschlossenem Drucker alle Rechenvorgänge ausgedruckt werden.
MODE 8 .... Das "PRT" Symbol wird gelöscht, so daß keine Daten ausgedruckt werden können.

← → Kursor-Tasten

Durch Drücken dieser Tasten wird der Kursor nach links oder rechts bewegt. Mit jedem Tastendruck wird der Kursor um eine Stelle verschoben. Die Taste gedrückt halten, wenn der Kursor rasch in die gewünschte Richtung bewegt werden soll.

ON AC Gesamtlöschtaste

- Diese Taste drücken, um die gesamte Anzeige zu löschen.
- Wird diese Taste während einer Programmausführung betätigt, dann wird das Programm angehalten.
- Bei Anzeige eines Fehlersymbols wird dieses durch Drücken dieser Taste gelöscht.
- Ist die Anzeige mittels Abschaltautomatik (siehe Seite 9) abgeschaltet, dann ist diese Taste zu drücken, sobald die Stromversorgung wieder eingeschaltet werden soll.

INS DEL Lösch/Einfügetaste

- Löscht das Zeichen an der Position des blinkenden Kursors.
- In der umgeschalteten Betriebsart wird durch Drücken dieser Taste eine Leerstelle eingegeben, so daß ein Zeichen eingefügt werden kann.

STOP Stopptaste

Wird diese Taste während der Ausführung eines Programmes gedrückt, dann wird "STOP" in der Sichtanzeige angezeigt und das Programm hält am Ende der eben ausgeführten Programmzeile an.
Wird während des Ablaufes des Prüfprogrammes "STOP" in der Sichtanzeige angezeigt, dann wird durch Drücken dieser Taste die Programm-Nummer und die Zeilen-Nummer angezeigt.

EXE Ausführungstaste

- Um das Ergebnis bei manuellen Rechnungen zu erhalten, diese Taste anstelle der "=" Taste drücken.
- In der Betriebsart "WRT" ist beim Einschreiben von Programmen diese Taste zu drücken, um die eingetasteten Zeilen in den Programmspeicher zu übertragen. Wird diese Taste nicht nach jeder eingetasteten Zeile gedrückt, dann wird die entsprechende Zeile nicht in den Programmspeicher eingeschrieben.
- In der Betriebsart "RUN" ist diese Taste während der Ausführung eines Programmes für die Dateneingabe zu drücken, wenn in der Sichtanzeige "STOP" angezeigt wird. Durch das Drücken dieser Taste wird nach der Dateneingabe das Programm fortgesetzt.

ANS Antworttaste

Bei manuellen Rechnungen ist diese Taste zu drücken, um das Ergebnis (die Antwort) der vorhergehenden Rechnung abzurufen.

π EXP Exponenten/Kreiskonstantentaste

Für die Eingabe des Exponenten ist diese Taste nach dem Eintasten der Mantisse zu drücken.

Beispiel: 2.56×10^{34} → 2 . 5 6 EXP 3 4

* Der Exponent kann mit bis zu ±99 eingegeben werden; wird versucht einen höheren Exponenten einzugeben, dann kommt es zur Fehleranzeige.

In der Zweitfunktion wird durch Drücken dieser Taste die Kreiskonstante Pi (3.14 .....) abgerufen.

≠ = Gleichheits/Vergleichstaste

- Diese Taste drücken, wenn eine Austauschanweisung wie z.B. eine Vergleichsanweisung mit einem IF Befehl einzugeben (Gleichheitszeichen).
- In der umgeschalten Betriebsart wird diese Taste für einen Vergleich bei Verwendung der IF Anweisung benutzt.

P7 P8 P9
7 8 9
P4 P5 P6
4 5 6
P1 P2 P3
1 2 3
P0 ↑
0 .

Zifferneingabe/Programmnummerntasten

- Diese Tasten sind für die Eingabe der Zahlen in den Computer zu drücken. Bei Dezimalzahlen ist die [.] Taste an der logischen Stelle zu betätigen.
- In der umschalten der Betriebsart wird mit den Tasten P0 [0] bis P9 [9] die Programmnummer eingegeben bzw. bei schon eingeschriebenem Programm wird die eingetastete Programmnummer für die Ausführung aufgerufen.
- Die [↑] Taste ist in der umschalten der Betriebsart zu drücken, um eine Hochzahl (x^y) einzugeben.

≥ ≤ > <
+ − * /

Rechenbefehl/Vergleichstasten

- Für Additionen, Subtraktionen, Multiplikationen und Divisionen sind diese Tasten an der entsprechenden Stelle zu drücken.
  [*] wird für Multiplikationen verwendet (entspricht "x")
  [/] wird für Divisionen verwendet (entspricht "÷")
- In der umschalten der Betriebsart sind diese Tasten für die Beurteilung eines Vergleiches mittels IF Anweisung zu betätigen.

Q	W	E	R	T	Y	U	I	O	P
GOSUB A SIN	FOR S COS	TO D TAN	STEP F ASN	NEXT G ACS	GOTO H ATN	IF J LOG	THEN K LN	PRINT L EXP	
RETURN Z SQR	STOP X ABS	END C SGN	DEFM V INT	RUN B FRAC	LIST N RAN#	INPUT M CSR	SPC		

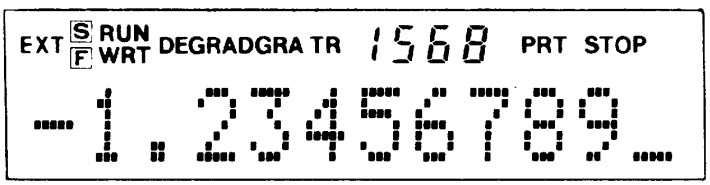
Buchstabentasten/Befehlstasten/Zeichentasten

Für das Eintasten eines Programmes oder einer Anweisung/Funktionsanweisung sind diese Tasten zu drücken, um die entsprechenden Buchstaben einzugeben. Durch Drücken der [SPC] Taste wird eine Leerstelle eingegeben.

GOSUB A ~ INPUT M Tasten: In der umgeschalten Betriebsart wird durch Drücken dieser Tasten das über der entsprechenden Taste aufgeführte Zeichen eingegeben.

A SIN ~ M CSR Tasten: In der umgeschalten Betriebsart können durch Drücken dieser Taste die entsprechenden über den Tasten aufgeführten Anweisungen eingegeben werden.

1-2 Ablesen der Sichtanzeige



Hier werden die Eingabewerte und die Ergebnisse angezeigt. Jedes Zeichen wird in einem Punktraster aus 5 (Breite) mal 7 (Höhe) Punkten angezeigt. Bis zu 12 Zeichen können gleichzeitig angezeigt werden. (Die Null wird als Ø angezeigt.) Falls eine Formel oder ein Programmadresse aus mehr als 12 Zeichen besteht, dann wird die Anzeige nach links verschoben, so daß bis zu 62 Zeichen aufeinanderfolgend eingegeben werden können.

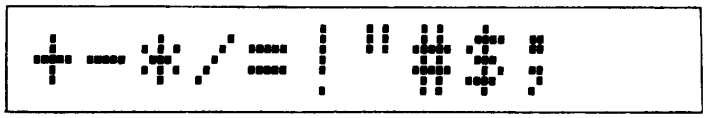
Der blinkende Kursor wird angezeigt, bis 55 Zeichen eingegeben wurden, worauf ab dem 56. Zeichen eine blinkende "■" angezeigt wird.

Die vier oben im Anzeigefeld angezeigten Stellen geben die noch zur Verfügung stehende Anzahl an Programmschritten an.

Während des Rechnung erscheint ein Minuszeichen "−" an der ersten Stelle von rechts der hochgestellten Vierstellenanzeige. Auch das Argument "DEG", "RAD" oder "GRA", "[S]" (wenn die [S] Taste gedrückt wird), "[F]" (wenn die [F] Taste gedrückt wird) bzw. die Betriebsartensymbole "RUN" (Programmausführung), "WRT" (Schreiben eines Programms), "TR" (Prüfprogramm), "PRT" (Druckerbetrieb) und "STOP" werden an den entsprechenden Stellen der hochgestellten Anzeigesymbole angezeigt.

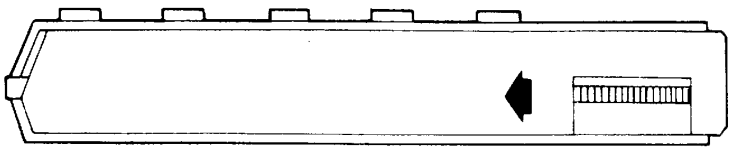
- **Beispiel für alphabetische Anzeige**

- **Beispiel für Symbolanzeige**



1-3 Kontrasteinstellung

Mit Hilfe des an der rechten Seite des Computers angebrachten Reglers kann der Kontrast der Sichtanzeige eingestellt werden.



Den Regler in Pfeilrichtung drehen, um den Kontrast zu erhöhen; in entgegengesetzter Richtung drehen, wenn ein geringerer Kontrast gewünscht wird. Mit diesem Regler kann der Kontrast an die Batteriespannung bzw. an die persönlichen Sehgewohnheiten angepaßt werden.

1-4 Erweiterung der Speicherkapazität

Normalerweise stehen 26 Speicher (Veränderliche) zur Verfügung, wobei gleichzeitig 1568 Programmschritte verwendet werden können.
Eine Erweiterung auf insgesamt 222 Speicher ist jedoch möglich. Für die Speichererweiterung werden 8 Programmschritte für jeden zusätzlichen Speicher verwendet.

Anzahl der Speicher	Anzahl der Programmschritte
26	1568
27	1560
28	1552
⋮	⋮
46	1408
⋮	⋮
94	1024
⋮	⋮
200	176
⋮	⋮
222	0

Um die Anzahl der Speicher zu erweitern, wird der DEFM Befehl verwendet.

Beispiel:

Die Anzahl der verfügbaren Speicher ist um 30 Speicher auf 56 Speicher zu erhöhen.

Bedienung:

Auf die Betriebsart RUN (Tasten [MODE] [0] drücken) oder WRT (Tasten [MODE] [1] drücken) schalten.

DEFM30 [EXE] → ***VAR:56

* Die DEFM Anweisung kann durch Drücken der Tasten [D] [E] [F] [M] oder [S] [DEFM (V)] eingegeben werden.

Eine DEFM Anweisung wird auch verwendet, um die Anzahl der zur Verfügung stehenden Speicher zu kontrollieren.

Beispiel:

Insgesamt 56 Speicher stehen zur Verfügung.

DEFM [EXE] | ***VAR:56

- Falls bereits eine hohe Anzahl von Programmschritten belegt wurde, dann kommt es zur Fehlerverriegelung, wenn versucht wird, die Anzahl der Speicher über die Anzahl der noch zur Verfügung stehenden Anzahl von Programmschritten hinaus zu erweitern. (EER1 ..... Unzureichende Anzahl von Programmschritten)
- Die exklusive Zeichenveränderliche ($) wird nicht gezählt, da sie einen speziellen Speicher verwendet.

1-5 Abschaltautomatik (AUTO POWER OFF)

Um wertvollen Batteriestrom zu sparen, wird bei eingeschaltetem Stromschalter die Stromversorgung etwa sieben Minuten nach der letzten Tastenbetätigung automatisch abgeschaltet (ausgenommen bei Programmausführung). Um die Stromversorgung wieder einzuschalten, den Stromschalter aus- und wiedereinschalten oder die [AC] Taste drücken.

* Speicherinhalt und Programme werden nicht gelöscht, wenn die Stromversorgung abgeschaltet wird. Das Argument sowie die Betriebsart ("WRT", "TR", "PRT" usw.) werden jedoch gelöscht.

Kapitel 2

Vor dem Beginn von Rechnungen

Manuelles Rechnen und Programmrechungen werden in der Betriebsart "RUN" durchgeführt (die Tasten [MODE] [0] drücken, wodurch RUN angezeigt wird).
Das jeweils eingestellte Argument "DEG", "RAD" oder "GRA" hat keinen Einfluß auf das Rechenergebnis, wenn keine Winkelberechnungen durchgeführt werden.

2-1 Hierarchien (tatsächliche Algebralogik)

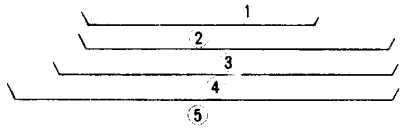
Dieser Computer rechnet mit Hierarchien, d.h. bestimmte Rechenvorgänge haben Vorrang über andere. Diese Vorrangsfolge ist:

① Funktionen (SIN, COS, TAN usw.)
② Potenzen
③ Multiplikationen und Divisionen (✱ und /)
④ Additionen und Subtraktionen (+ und −)

Bei gleicher Prioritätsfolge wird die Rechnung in der geschriebenen Form von links nach rechts durchgeführt. Bei der Berechnung von Klammerausdrücken haben die in den Klammern enthaltenen Rechnungen Vorrang.

Beispiel:

2+3✱SIN(17+13)↑2=2.75



2-2 Ein/Ausgabekapazität

Die Ein/Ausgabekapazität des Computers umfaßt 12 Stellen für die Mantisse und 2 Stellen für den Exponenten. Interne Rechenvorgänge werden ebenfalls mit 12 Stellen für die Mantisse und 2 Stellen für den Exponenten ausgeführt.
Der Bereich umfaßt dabei 1×10^{-99} bis $\pm 9.99999999999 \times 10^{+99}$.
Die Ausgabe erfolgt mit 10 Stellen für die Mantisse und 2 Stellen für den Exponenten. Wird jedoch ein Exponent verwendet, dann stehen für die Mantisse nur 8 Stellen zur Verfügung.

* Falls bei Funktionsrechnungen usw. das Rechenergebnis mehr als 12 Stellen aufweist, dann werden nur insgesamt 12 Stellen angezeigt (einschließlich 0 und Dezimalpunkt).

Beispiel:

$(1 \times 10^5) \div 7 = 14285.71429$
$(1 \times 10^5) \div 7 - 14285 = 0.7142857$

1 [EE] 5 [÷] 7 [EXE]	14285.71429
1 [EE] 5 [÷] 7 [−] 14285 [EXE]	0.7142857

Wenn das Rechenergebnis mehr als 10^{10} (10,000,000,000) oder weniger als 10^{-3} (0.001) beträgt, dann erfolgt die Anzeige automatisch in der halblogarithmischen Schreibweise.

Beispiel:

$1234567890 \times 10 = 12345678900$
$(= 1.23456789 \times 10^{10})$

1234567890 × 10 EXE → 1.2345678E10

Vorzeichen des Exponenten

* Im Anschluß an die Mantisse wird der Exponent mit Vorzeichen angezeigt.

Beispiel:

$1.234 \div 10000 = 0.0001234$
$(= 1.234 \times 10^{-4})$

1.234 ÷ 10000 EXE → 1.234E-04

2-3 Grundrechnungen

2-3-1 Rechensymbole und Funktionsbefehle

In der BASIC Programmiersprache werden für Addition und Subtraktion die gleichen Symbole (+ und −) verwandt; die Symbole * und / werden jedoch für Multiplikationen und Divisionen eingesetzt.
Beispiel:

2+3−4×5÷6

wird geschrieben als

2+3−4*5/6

Dieser Rechner enthält auch die folgenden Funktionen.

Funktionsbezeichnung		Format	
Trigonometrische Funktionen	$\sin x$	SINx	[F] [A SIN]
	$\cos x$	COSx	[F] [S COS]
	$\tan x$	TANx	[F] [D TAN]
Umgekehrte trigonometrische Funktionen	$\sin^{-1} x$	ASNx	[F] [F ASN]
	$\cos^{-1} x$	ACSx	[F] [G ACS]
	$\tan^{-1} x$	ATNx	[F] [H ATN]
Quadratwurzel	$\sqrt{x}$	SQRx	[F] [Z SQR]
Exponentialfunktion	e	EXP1*	[F] [L EXP]
Natürlicher Logarithmus	$\ln x$	LNx	[F] [K LN]
Zehnerlogarithmus (Briggsscher Logarithmus)	$\log x$	LOGx	[F] [J LOG]
Anzeige nur des ganzzahligen Teiles	INTx	INTx	[F] [V INT]
Anzeige nur der Dezimalstellen	FRACx	FRACx	[F] [B FRAC]
Anzeige des Absolutwertes	$\lvert x \rvert$	ABSx	[F] [X ABS]
Vorzeichenumkehr	Postive Zahl → 1 0 → 0 Negative Zahl → −1	SGNx	[F] [C SGN]
Rundungsfunktion	(10^y von x wird gerundet)	RND (x, y)	[—]
Zufallszahl		RAN #	[F] [N RAN#]
Statistische Rechnungen			

* Für die RND Funktion muß das Argument in Klammen eingesetzt werden.

★ EXP ist eine Anweisung für das Abrufen des Zahlenwertes aus der Exponentialtabelle.

2-3-2 Abruf des vorhergehenden Rechenergebnisses

Das Ergebnis einer manuellen oder Programmrechnung wird bis zur Durchführung der nächsten Rechnung gespeichert. Dieses Ergebnis kann durch Drücken der [ANS] Taste abgerufen werden.

Beispiel: 741+852=1593
2431−1593=838

Bedienung:

[7][4][1][+][8][5][2]	741+852
[EXE]	1593
[2][4][3][1][−][ANS]	2431−1593
[EXE]	838

Auch der nach der Rechnung angezeigte Zahlenwert kann in der darauffolgenden Rechnung verwendet werden.

Beispiel: (Anschließend an obige Rechnung)
838×2=1676

Bedienung:

[*][2]	838*2
[EXE]	1676

2-3-3 Fehleranzeige

Falls eine Formel oder ein Programmsatz nicht in der richtigen BASIC Programmsprache eingegeben oder der Rechenbereich überschritten wird, dann kommt es während des Rechnungsablaufes zu einem Fehler, so daß die Fahleranzeige in der Sichtanzeige erscheint. Bei Potenzrechnungen ($x \uparrow y$) kommt es zu keiner Fahleranzeige, wenn y eine natürliche Zahl ist und x weniger als Null (0) beträgt. Die folgenden Fehleranzeigen erscheinen bei manuellen Rechnungen:

ERR2	(Fehler im Satzaufbau)
ERR3	(Mathematischer Fehler)

Die folgenden Fehleranzeigen erscheinen bei Programmrechnungen:

ERR2 P0-10
Programmnummer Zeilennummer

(Bedeutung: In Programm-Nummer P0 liegt ein Fehler im Satzaufbau der Zeile-Nr. 10 vor.)

ERR3 P2-20

(Bedeutung: In Programm-Nummer P2 liegt ein mathematischer Fehler in Zeile-Nr. 20 vor.)

Die Bedeutung der Fehleranzeigen ist der Fehleranzeigeliste auf Seite 67 zu entnehmen.

* Bei Überschreitung des Rechenbereiches (±9.999999999E+99) kommt es zu Überlauf und das Fehlersymbol wird angezeigt. Unter 1.0×10^{-99} tritt Unterlauf ein, d.h. das Rechenergebnis lautet "0" (Null).

2-4 Tastenbedienung

Zuerst die Stromversorgung einschalten (ON). In der Sichtanzeige erscheint der Schriftzug "READY PØ" und der Rechner ist bereit für die Eingabe.

1. Tasteneingabe

- **Alphabetische Eingabe**

Beispiel: Eingabe ABC
Bedienung: [A][B][C]

```
ABC
```

Beispiel: Eingabe SIN
Bedienung: [S][I][N] (oder [F] [A/SIN])

```
SIN
```

* Entweder der Eintastenbefehl oder der alphabetische Befehl kann verwendet werden.

- **Zahleneingabe**

Beispiel: Eingabe 123
Bedienung: [1][2][3]

```
123
```

Beispiel: Eingabe 96.3
Bedienung: [9][6][.][3]

```
96.3
```

- **Symboleingabe**

Beispiel: Eingabe $#?
Bedienung: [S] [$/R] [S] [#/E] [S] [?/U]

```
$ # ?
```

Beispiel: Eingabe ¥ΣΩ
Bedienung: [MODE][.][S] [¥/R] [S] [Σ/B] [S] [Ω/N] [MODE][.]

```
¥ΣΩ
```

- **Eingabe von Zahlen mit Exponenten**

Beispiel: Eingabe 7.896×10^{15}
Bedienung: [7][.][8][9][6] [π/E] [1][5]

```
7.896E15
```

Beispiel: Eingabe -2.369×10^{-45} Exponentensymbol
Bedienung: [−][2][.][3][6][9] [π/E] [−][4][5]

```
-2.369E-45
```

2. Änderung der Eingabe (Berichtigung, Streichung und Einfügung)

- **Berichtigung**

Den Kursor an die Stelle bringen, an der die Berichtigung vorgenommen werden soll (dazu die [←] [→] Tasten verwenden). Danach die Taste des entsprechenden Buchstabens, der Zahl oder des Symboles drücken.

Beispiel: Änderung von "A$" auf "B$".

```
A$_
```

Bedienung: Den Kursor um zwei Stellen nach links bewegen.

[←][←]

```
A$
```

Die [B] Taste drücken.

```
B$
```

Beispiel: Änderung von "LIST" auf "RUN".

```
LIST_
```

Bedienung: Den Kursor um vier Stellen nach links bewegen.

[←][←][←][←]

```
LIST
```

Die Tasten [R][U][N][SPC] oder [S] [RUN/B] drücken.

```
RUN_
```

- **Streichung**

Den Kursor an die zu streichende Position bringen und die [DEL] Taste drücken. Mit jedem Druck dieser Taste wird das über dem Kursor befindliche Zeichen gelöscht, wobei gleichzeitig die rechts davon liegenden Zeichen um eine Stelle nach links bewegt werden.

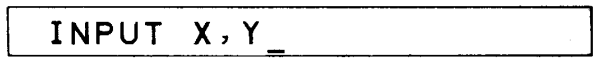
Beispiel: Streichung von "I" in "SIIN".

Bedienung: Den Kursor um zwei Stellen nach links bewegen.

[←][←]

Die [DEL] Taste drücken.

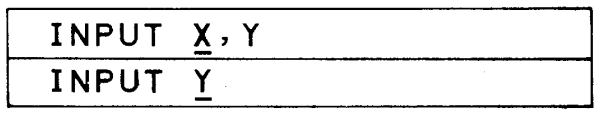
Beispiel: Streichung von "X" in "INP X, Y".



Bedienung: Den Kursor um drei Stellen nach links bewegen.

[←][←][←]

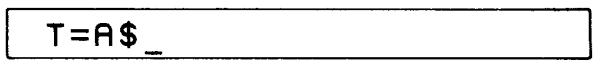
Die Tastenfolge [DEL][DEL] drücken.



- **Einfügung**

Den Kursor rechts neben die Stelle bewegen, an welcher ein neues Zeichen eingefügt werden soll. Danach die Tasten [S] [INS DEL] drücken, wodurch eine Freistelle erscheint. Danach die entsprechende Taste betätigen, um das gewünschte Zeichen an der Freistelle einzufügen.

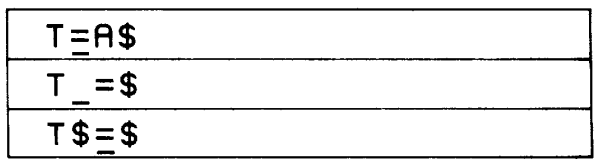
Beispiel: Änderung von "T=A$" auf "T$=A$".



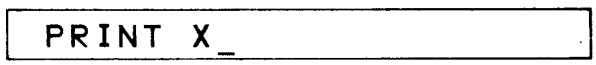
Bedienung: Den Kursor um drei Stellen nach links bewegen.

[←][←][←]

Die Tastenfolge [S] [INS DEL] drücken, wodurch eine Leerstelle über dem Kursor erscheint. Danach die Tastenfolge [S] [$ R] eingeben, um das Zeichen "$" an der genannten Stelle einzufügen.

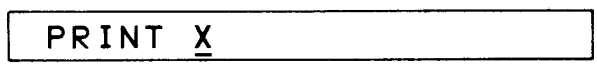


Beispiel: Änderung von "PRINT X" auf "PRINT SIN X".



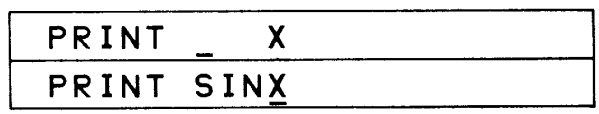
Bedienung: Den Kursor um eine Stelle nach links bewegen.

[←]



Die Tastenfolge [S] [INS DEL] [S] [INS DEL] [S] [INS DEL] verwenden, um drei Leerstellen einzugeben.

Danach die Tastenfolge [S][I][N] drücken.



Mit Hilfe der obigen Verfahren können die Eingaben berichtigt bzw. geändert werden.

Kapitel 3

Manuelles Rechnen

3-1 Erläuterung des manuellen Rechnens

Beim manuellen Rechnen werden die Rechenabläufe nicht automatisch mit den vorprogrammierten Formeln durchgeführt. Verschieben der Zahlen von rechts nach links, manuelle Eingabe und manueller Abruf von Veränderlichen werden als manuelles Rechnen bezeichnet.

3-2 Manuelle Rechenvorgänge

- Addition, Subtraktion, Multiplikation und Division werden mit tatsächlicher Algebralogik durchgeführt. Die Tasten [+], [−], [×] (X), [÷] (÷) und [EXE] (=) werden verwendet. Die [EXE] Taste wird zur Berechnung des Ergebnisses verwandt und hat die Funktion der "=" Taste.

Beispiel: 12+36−9×5÷4=36.75

Bedienung: [1][2][+][3][6][−][9][×][5][÷][4]

```
12+36-9*5/4
```

[EXE]

```
36.75
```

- Funktionsrechnungen mit Additionen, Subtraktionen, Multiplikationen und Divisionen werden ebenfalls in tatsächlicher Algebralogik durchgeführt. Die Daten sind nach dem Funktionsbefehl einzugeben.

Beispiel: log 1.23=0.0899051114

Bedienung: LOG 1[.]23

```
LOG1.23
```

[EXE]

```
0.0899051114
```

* Buchstaben und Zahlen werden in dieser Anleitung nicht umrandet dargestellt.

Beispiel: [S][I][N][S][(][1][5][+][8][S][)][EXE] → SIN[S](15+8[S])[EXE]

* Funktions- und Programmbefehle können mit Hilfe der Eintasten-Befehlstasten oder mittels alphabetischer Befehle eingegeben werden; in dieser Anleitung werden nur die alphabetischen Befehle verwendet.

- Auch Speicherrechnungen sowie Speicherung von Zahlenwerten oder Rechenergebnissen sind möglich, wobei für die Berechnung der Summen Veränderliche verwendet werden. Diese Veränderlichen werden durch alphabetische Buchstaben (A–Z), oder durch Kombination von Buchstaben und Ziffern (0–9) (wenn die Speicher als Datenfeldveränderliche verwendet werden) eingegeben. Um einen Zahlenwert oder ein Rechenergebnis als Veränderliche einzustellen, wie folgt vorgehen.

Beispiel: Den Zahlenwert 1234 als Veränderliche A speichern.

Bedienung: A[=]1234

```
A=1234
```

[EXE]

```
_
```

Beispiel: Das Ergebnis von 23 x 56 zur Veränderlichen K addieren.

Bedienung: K ⊟ K ➕ 23 ✱ 56 | K=K+23*56

EXE | _

Mit diesem Verfahren kann manuell der gleiche Vorgang wie durch einen Programmsatz in einem Programm durchgeführt werden.

- Um vor dem Drücken der EXE Taste eine Berichtigung vorzunehmen, den Kursor an die zu berichtigende Stelle verschieben und die richtige Taste drücken (siehe Kapitel 2).
- Um die gesamte Anzeige zu löschen, die AC Taste drücken.

3-3 Manuelle Rechenbeispiele

3-3-1 Grundrechenarten

■ **Addition, Subtraktion, Multiplikation und Division.**

Beispiel: 23+4.5−53=−25.5

Bedienung: 23 ➕ 4 ⊡ 5 ➖ 53 EXE | -25.5

Beispiel: 56×(−12)÷(−2.5)=268.8

Bedienung:

56 ✱ (−) 12) ◢ (−) 2 ⊡ 5) EXE | 268.8

(kann weggelassen werden)

Beispiel: 12369×7532×74103=6.9036806×10^12 (=6903680600000)

Bedienung: 12369 ✱ 7532 ✱ 74103 EXE | 6.9036806E12

Beispiel: 1.23÷90÷45.6=2.9970760×10^−4 (=0.00029970760)

Bedienung: 1 ⊡ 23 ◢ 90 ◢ 45 ⊡ 6 EXE | 2.9970760E−04

* Wenn das Ergebnis größer als 10^9 oder kleiner als 10^{-3} ist, erfolgt die halblogarithmische Anzeige mit Mantisse und Exponent.

Beispiel: 7×8+4×5=76

Bedienung: 7 ✱ 8 ➕ 4 ✱ 5 EXE | 76

Beispiel: 12+(2.4×10^5)÷42.6−78×36.9=2767.602817

Bedienung:

12 ➕ 2 ⊡ 4 EXP 5 ◢ 42 ⊡ 6 ➖ 78 ✱ 36 ⊡ 9 EXE | 2767.602817

■ Speicherrechnung

Beispiel:
$12 \times 45 = 540$
$12 \times 31 = 372$
$75 \div 12 = 6.25$

Bedienung:

A = 12 EXE	_
A × 45 EXE	540
A × 31 EXE	372
75 ÷ A EXE	6.25

Beispiel:
$23 + 9 = 32$
$53 - 6 = 47$
$-) \; 45 \times 2 = 90$
$99 \div 3 = 33$
Summe 22

Bedienung:

M = 23 + 9 EXE	
M = M + 53 − 6 EXE	
M = M − 45 × 2 EXE	
M = M + 99 ÷ 3 EXE	
M EXE	22

* Bei Verwendung dieses Verfahrens können die einzelnen Rechenergebnisse nicht unterschieden werden. Um die einzelnen Ergebnisse zu erhalten, wie folgt verfahren:

23 + 9 EXE	32
M = ANS EXE	
53 − 6 EXE	47
M = M + ANS EXE	
45 × 2 EXE	90
M = M − ANS EXE	
99 ÷ 3 EXE	33
M = M + ANS EXE	
M EXE	22

3-3-2 Funktionsrechnungen

- **Trigonometrische Funktionen (sin, cos, tan) und umgekehrte trigonometrische Funktionen ($\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$)**
- Wenn trigonometrische oder umgekehrte trigonometrische Funktionen verwendet werden, unbedingt das Winkelargument bestimmen.
  (Falls das Winkelargument nicht geändert werden muß, ist eine Neueingabe nicht erforderlich.)

Beispiel: $\sin 12.3456° = 0.2138079201$

Bedienung: MODE 4 → "DEG"

SIN 12 . 3456 EXE

(oder F A SIN 12 . 3456 gleich wie folgt)

DEG
0.2138079201

Beispiel: $2 \cdot \sin 45° \times \cos 65.1° = 0.5954345575$

Bedienung:

2 × SIN 45 × COS 65.1 EXE

DEG
0.5954345575

Beispiel: $\sin^{-1} 0.5 = 30°$ (berechne x für $\sin x° = 0{,}5$)

Bedienung: ASN 0 . 5 EXE

(oder F F ASN 0 . 5 gleich wie folgt)

DEG
30

Beispiel: $2.5 \times (\sin^{-1} 0.8 - \cos^{-1} 0.9) = 68.22042398$

Bedienung:

2 . 5 × (ASN 0 . 8 − ACS 0 . 9) EXE

DEG
68.22042398

Beispiel: $\cos(\frac{\pi}{3}\text{rad}) = 0.5$

Bedienung: MODE 5 → "RAD"

COS (π ÷ 3) EXE

RAD
0.5

Beispiel: $\cos^{-1}\frac{\sqrt{2}}{2} = 0.7853981634$

Bedienung: ACS (SQR 2 ÷ 2) EXE

RAD
0.7853981634

Beispiel: $\tan(-35\text{gra}) = -0.612800788$

Bedienung: MODE 6 → "GRA"

TAN − 35 EXE

GRA
-0.612800788

- **Logarithmische Funktionen (log, ln) und Exponentialfunktionen (e, x^y)**

EXP (e) kann nicht in einer kontinuierliche Berechnung bzw. als Mehrfachanweisung verwendet werden.

Beispiel: $\log 1.23 (= \log_{10} 1.23) = 0.0899051114$

Bedienung: LOG 1 . 23 EXE

0.0899051114

Beispiel: $\ln 90\ (=\log_e 90) = 4.49980967$

Bedienung: LN 90 EXE

4.49980967

Beispiel: $\log 456 \div \ln 456 = 0.4342944819$

Bedienung: LOG 456 ◢ LN 456 EXE

0.4342944819

Beispiel: $e = 2.718281828$

(Mit dieser Funktion wird die Anweisung eingegeben, den entsprechenden Zahlenwert aus der Exponentialtabelle abzurufen.)

Bedienung: EXP 1 EXE

2.718281828

Beispiel: $10^{1.23} = 16.98243652$

(Berechnung des Antilogarithmus des Zehnerlogarithmus 1,23)

Bedienung: 10 S ↑ 1 . 23 EXE

16.98243652

Beispiel: $5.6^{2.3} = 52.58143837$

Bedienung: 5 . 6 S ↑ 2 . 3 EXE

52.58143837

Beispiel: $123^{\frac{1}{7}}\ (=\sqrt[7]{123}) = 1.988647795$

Bedienung: 123 S ↑ S (1 ◢ 7 S) EXE

1.988647795

Beispiel: $(78-23)^{-12} = 1.3051118 \times 10^{-21}$

Bedienung: S (78 − 23 S) S ↑ − 12 EXE

1.3051118E⁻21

Beispiel: $2^2+3^3+4^4 = 287$

Bedienung: 2 S ↑ 2 + 3 S ↑ 3 + 4 S ↑ 4 EXE

287

Beispiel: $\log \sin 40° + \log \cos 35° = -0.278567983$

Der Antilogarithmus beträgt 0,5265407845 (logarithmische Berechnung von sin 40° x cos 35°)

Beispiel: MODE 4 (Winkelargument "DEG")

Bedienung: LOG SIN 40 + LOG COS 35 EXE

-0.278567983

10 S ↑ ANS EXE

0.5265407845

* Eingabebereich für Potenzen ($x \uparrow y$) ist $x > 0$.

■ **Sonstige Funktionen ($\sqrt{\ }$, SGN, RAN#, RND, ABS, INT, FRAC)**

Beispiel: $\sqrt{2}+\sqrt{5}=3.65028154$

Bedienung: SQR 2 + SQR 5 EXE

3.65028154

Beispiel: **Es soll "1" für eine positive Zahl, "−1" für eine negative Zahl und "0" für eine Null gegeben werden.**

Bedienung:

SGN 6 EXE	1
SGN 0 EXE	0
SGN − 2 EXE	−1

Beispiel: **Erzeugung einer Zufallszahl (Pseudo-Zufallszahl von $0 < RAN\# < 1$)**

Bedienung: RAN S # EXE (oder F (N) RAN# EXE)

0.904186914

Beispiel: **Das Ergebnis von 12.3 x 4.56 ist auf 10^{-2} zu runden.** $12.3\times4.56=56.088$

Bedienung: RND S (12 · 3 + 4 · 56 S ÷ − 2 S) EXE

56.1

Beispiel: $|-78.9\div5.6|=14.08928571$

Bedienung: ABS S (− 78 · 9 / 5 · 6 S) EXE

14.08928571

Beispiel: **Der ganzzahlige Teil von $\frac{7800}{96}$ ist 81**

Bedienung: INT S (7800 / 96 S) EXE

81

* Diese Befehl übersteigt nicht den ursprünglichen Zahlenwert.

Beispiel: **Der Dezimalteil von $\frac{7800}{96}$ ist 0,25**

Bedienung: FRAC S (7800 / 96 S) EXE

0.25

■ **Bestimmung der Stellen- und Dezimalstellenzahl**

Die Stellen- und die Dezimalstellenzahl kann mit Hilfe des "SET" Befehls eingestellt werden.

Stellenzahl . . . . . SET E *n* (*n*=0 bis 9)
Dezimalstellenzahl . . . . . SET F *n*
Aufhebung der Stellenzahl . . . . . SET N

* Wird für die Stellenzahl "SET E 0" eingegeben, dann erfolgt die Anzeige mit 8 Stellen.
* Wenn eine Bezeichnung erfolgt, wird die letzte Bezeichnung gerundet angezeigt. Für interne Rechnungen wird jedoch der ursprüngliche Zahlenwert verwendet.

Beispiel: $100\div6=16.66666666.....$

Bedienung: SET E 4 EXE (Stellenzahl auf 4 Stellen)

100 / 6 EXE

1.667E01

Beispiel: $123\div7=17.57142857.....$

Bedienung: SET F 2 EXE (zwei Dezimalstellen)

123 / 7 EXE

17.57

Beispiel: $1\div3=0.3333333333.....$

Bedienung: SET N EXE (Stellenzahl aufgehoben)

1 / 3 EXE

0.3333333333

Kapitel 4

Programmrechnungen

Dieser Elektronikrechner verwendet die BASIC (Beginners All-purpose Symbolic Instruction Code) Programmiersprache. Es handelt sich dabei um eine sehr einfache problemorientierte Programmiersprache.

Merkmale der Programmiersprache BASIC

1. Sehr einfache problemorientierte Programmiersprache, die sich in einigen Stunden erlernen läßt.
2. Eine Programmiersprache, die in erster Linie für einfache Aufgaben aus dem technisch-wissenschaftlichen Bereich entwickelt wurde.
3. Verwendung der Programmiersprache im Dialog mit dem Computer, wobei jede Anweisung auf syntaktische Richtigkeit geprüft werden kann. Fehler teilt der Computer dem Anwender sofort mit.
4. Weist viele Merkmale und Eigenschaften der Programmiersprache FORTRAN auf, ist aber frei von den Regeln des FORTRAN Systems.
5. Viele eingebaute Funktionen, die den Anwendungsbereich noch weiter vergrößern.

4-1 Programmbeschreibung

Programmrechnungen **1)** Ausführung des Inhalts der Programmrechnung, **2)** Speicherung des Programms im Rechner, und **3)** Verwendung dieses Programms. Die Daten einfach eingeben, worauf automatisch die Ergebnisse erhalten werden.

■ Grundlagen des Programmierens

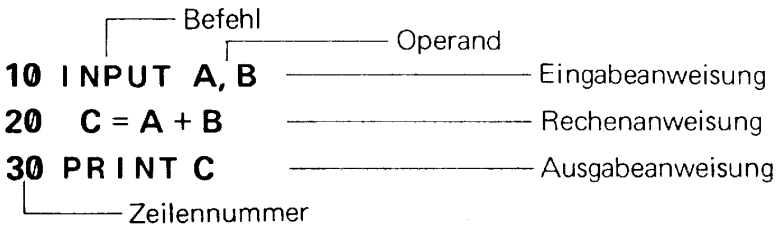
Wollen wir zuerst das Konzept der Programmiervorgänge und die für die Erstellung eines Programms erforderlichen Grundlagen betrachten.

- **Programme und Programmieren**

Wenn der Anwender einen Computer zur Lösung von Problemen verwenden möchte, dann muß er zuerst Anweisungen in einer Programmsprache eingeben, die der Computer verstehen kann. Diese Anweisungen werden in Form von Worten und Sätzen eingegeben (dieser Eingabevorgang wird Programmieren genannt), um das eigentliche Programm zu erhalten.

- **Was ist ein Programm?**

Ein Program ist der logische Ablauf von Algorithmen (Rechenabläufen), die vom Anwender zur Lösung eines Problems festgelegt werden. Dabei sind viele grammatische Regeln der verwendeten Programmiersprache zu verwenden, die später erläutert werden. Zuerst wollen wir das Format eines Programms betrachten und anhand eines Beispiels die Programmiergrundlagen kennenlernen.



Das obige Programm ist ein Grundprogramm und besteht aus der Eingabeanweisung, der Rechenanweisung, der Ausgabeanweisung und den Zeilennummern. Mit der Eingabeanweisung sind die Eingabedaten einzugeben, die dann mit Hilfe der Rechenanweisung verarbeitet werden, so daß das Ergebnis mittels Ausgabeanweisung ausgegeben werden kann. Die Rechenanweisung kann viele unterschiedliche Rechenbefehle enthalten, um z.B. komplexe Rechenvorgänge durchzuführen. Die grundlegenden Rechenanweisungen sind aber immer gleich.

Nach der jeweiligen Zeilennummer ist ein BEFEHL (Anweisung) einzugeben, der dem Rechner mitteilt, welcher Vorgang als nächstes auszuführen ist; diese Rechenanweisungen werden immer mit Hilfe der alphabetischen Buchstaben eingeschrieben. Anschließend ist der OPERAND einzutasten, der die nötigen Informationen für die Durchführung der Rechenbefehle gibt.

Der so erhaltene Rechenablauf wird als PROGRAMM in seiner grundlegendsten Form bezeichnet.

● **Anzahl der Programmschritte**

Die Programmschritte werden wie folgt gezählt:

1) Programmanweisung . . . . . 1 Schritt/1 Anweisung
2) Funktionsanweisung . . . . . 1 Schritt/1 Anweisung
3) Zeilennummer . . . . . 2 Schritte/1 Zeilennummer
4) Zeichen . . . . . 1 Schritt/1 Zeichen
5) Drücken der [EXE] Taste nach Tasteneingabe jeder Zeilennummer, um diese im Rechner zu speichern . . . . . 1 Schritt

Beispiel:

1 INPUT A [EXE] . . . . . 5 Schritte
(2) (1) (1) (1)

10 B = SIN A [EXE] . . . . . 7 Schritte
(2) (1) (1) (1) (1) (1)

100 PRINT " B = " ; B [EXE] . . . . . 10 Schritte
(2) (1) (1) (1) (1) (1) (1) (1) (1)

Gesamt 22 Schritte

■ Programmierfolge

Für das Programmieren sind die folgenden Grundregeln zu beachten:

1) Definition des Problems
2) Erstellung eines Flußdiagramms
3) Codieren (Schreiben des Programms auf einem Programmvordruck)
4) Fehlersuche

Erläuterung:

Schritt 1: Das Problem ist zu definieren, um die erforderliche Anzahl von Schritten für die Lösung des Problems zu erhalten.

Schritt 2: Erstellung des Flußdiagramms, das einen logischen Ablauf zur Lösung des Problems enhält. Das Flußdiagramm besteht aus Symbolen, die die einzelnen Elemente für Anweisungen und Berechnungen darstellen.

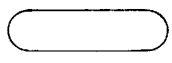
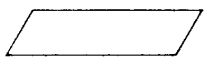
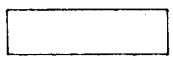
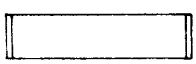
Schritt 3: Anhand des Flußdiagramms wird das Programm auf einem Programmvordruck für die BASIC Programmiersprache geschrieben.

Schritt 4: Das Programm auf Fehler prüfen.

Damit ist die Erstellung eines Programms abgeschlossen.

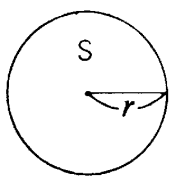
■ Flußdiagramm

Nachfolgend sind die am häufigsten verwendeten Symbole für die Erstellung eines Flußdiagramms aufgeführt.

Symbol	Benennung	Bedeutung
	Grenzstelle	Initialisierung, Abschluß usw.
	Eingabe / Ausgabe	Eingabe-/Ausgabefunktion
	Operation, allgemein	Verschiedene Operationsfunktionen
	Unterprogramm	Eine Gruppe von Befehlen, die an anderer Stelle aufgeführt sind, wie z.B. ein Unterprogramm
	Verzweigung	Entscheidung über die Wahl einer Befehlsfolge an einer Verzweigung, die mehrere Möglichkeiten offen läßt.

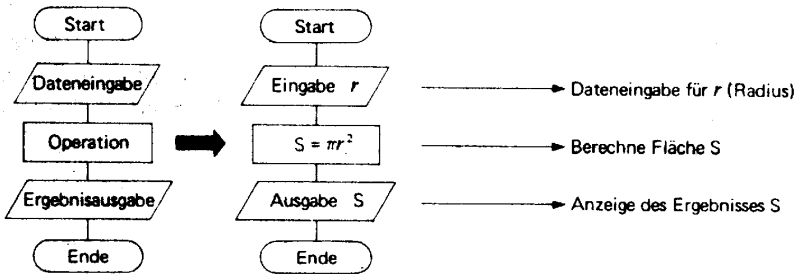
● **Beispiel eines Flußdiagramms**

Nachfolgend soll ein Programm für die Berechnung der Kreisfläche erstellt werden.



$S = \pi r^2$

Zuert sind die Anweisungen für Eingabe, Operation und Ausgabe separat zu berücksichtigen, wie es nachfolgend gezeigt ist.



Das Flußdiagramm zeigt die logischen Abläufe und gibt eine Übersicht über das vollständige Programm. Vor dem eigentlichen Programmieren sollte immer ein Flußdiagramm erstellt werden, da bei komplexen Programmen ein Flußdiagramm alle Abläufe verdeutlicht.

■ Codieren

Mit Codieren wird das Umsetzen von Informationen aus der Umgangssprache oder aus geläufigen Begriffen in einen definierten, vom Computer erkennbaren Code (Programmiersprache) bezeichnet. Dabei sind für arithmetische Operationen in einem BASIC-Programm die folgenden Symbole zu verwenden

\+ für Addition
– für Subtraktion
X für Multiplikation
/ für Division
↑ (vorangestellt) für einen Exponenten (so wird z.B. x^2 und x^3 als " $x\uparrow 2$ " bzw. " $x\uparrow 3$ " geschrieben)

Das " = " Symbol wird für Zuordnungsanweisungen benutzt, wie es später noch genauer erläutert wird. So bedeutet z.B. das " = " in $S = \pi r^2$, daß das Rechenergebnis πr^2 dem Ausdruck S zugeordnet ist (im Gegensatz zum mathematischen Gleichheitszeichen).
Wollen wird nun das Programm für die definierte Berechnung der Kreisfläche erstellen.

1) Einschreiben der Eingabeanweisung für die Eingabe des Datenwertes *r*.
Es gibt verschiedene Eingabebefehle, wobei jedoch normalerweise der Befehl "INPUT" verwendet wird, um während des Programmablaufes Daten über die Tastatur einzugeben. Die Eingabeanweisung für die Eingabe des Datenwertes *r* wird damit INPUT R. Dieser Anweisung ist die Zeilennummer voranzustellen, so daß sich der Programmsatz 10 INPUT R ergibt.
Bei der Eingabe muß also jede Anweisung numeriert (Zeilennummer) werden. Damit auch nachträglich noch Anweisungen in ein Programm eingefügt werden können, numeriert man zunächst in Zehnersprüngen. Ergänzungen erhalten dann Nummern, die zwischen den Nummern der beiden Anweisungen liegen, zwischen die sie eingefügt werden sollen. Das Eingeben solcher Ergänzungen kann unsortiert erfolgen, da der Computer selbst die richtige Reihenfolge herstellt.

2) Nun ist eine Zuordungsanweisung einzuschreiben, um das aus dem Eingabedatenwert erhaltene Ergebnis dem Ausdruck S zuzuordnen.
Da πr^2 gleich $\pi \times r^2$ ist, kann dieser Ausdruck wie folgt geschrieben werden:
S = πXR↑2 (X für Multiplikation, R↑2 für R^2)
Mit der dazugehörigen Zeilennummer ergibt sich also:
20 S = πXR↑2

3) Nun muß eine Ausgabeeinweisung eingeschrieben werden, damit das Ergebnis dieser Operation auch angezeigt wird.
Für die Anzeige des Ergebnisses ist der Befehl "PRINT" zu verwenden.
Die Ausgabeanweisung für die Ausgabe (Anzeige) des Ergebnisses S wird damit:
PRINT S
Mit der dazugehörigen Zeilennummer ergibt sich:
30 PRINT S
Das vollständige Programm für die Berechnung der Kreisfläche lautet also:

```
10 INPUT R
20 S=πXR↑2
30 PRINT S
```

Für das Codieren des Programms muß nicht unbedingt ein Programmvordruck verwendet werden, obwohl ein solcher jedoch die Definition und die Erstellung des Flußdiagramms wesentlich erleichtert.

■ Konstante und Veränderliche (Variable)

In der BASIC-Programmiersprache werden als Zeichen die Großbuchstaben (A bis Z), die Ziffern (0 bis 9) sowie einige Symbole verwendet.

● Konstante

Eine Konstante ist ein fester Wert und kann daher direkt in das Programm eingeschrieben werden.

Beispiel: In dem Ausdruck $S = \pi r^2$ stellt 2 eine Konstante dar, so daß geschriebenen werden kann S = π ✱ R ↑ 2.

● Veränderliche

Eine Veränderliche ist ein Wert, der bei der Erstellung des Programms noch nicht bekannt ist und während des Programmablaufes eingegeben werden muß, um das Ergebnis zu erhalten. Veränderliche werden mit einem einzigen Großbuchstaben (A bis Z) oder einem einzigen Großbuchstaben mit nachgestelltem $ (Zeichenveränderliche) bezeichnet. Die Veränderlichen können innerhalb des angegebenen Bereiches frei ausgewählt werden.

Beispiel: In dem Ausdruck $S = \pi r^2$, der als S = π ✱ R ↑ 2 zu schreiben ist, ist R die Veränderliche.

Beispiel:

```
Y = 2 ✱ X ↑ 2 + 3 ✱ X + 4
    |   |   |   |   |   |
V   C   V   C   C   V   C
```

V: Veränderliche

C: Konstante

Es kann also gesagt werden, daß die in der Mathematik verwandten algebraischen und Zahlensausdrücke den Veränderlichen bzw. Konstanten entsprechen. Zusätzlich zu den oben genannten Ausdrücken, gibt es auch Zeichenkonstante und Zeichenveränderliche. Zeichenkonstante werden dabei durch Zeichenfolgen dargestellt, die in Anführungszeichen direkt eingegeben werden, wie z.B. "ABC" und "END". Zeichenveränderliche stellen keinen numerischen Zahlenwert dar, sondern sind Zeichenfolgen zugeordnete Veränderliche. Mit jeder Zeichenfolge wird der Inhalt des entsprechenden Speichers geändert.
Gleichzeitige Verwendung einer Zahlenveränderlichen und einer Zeichenveränderlichen mit dem gleichen Buchstaben ist nicht möglich.
Eine Zeichenfolge besteht aus in Anführungszeichen gesetzten Zeichen, wie z.B. "123", stellt aber keinen numerischen Wert dar. Die Folge "123" ist also nur die Aneinanderreihung der Ziffern 1 und 2 und 3 und wird gleich wie der Ausdruck "ABC" in Anführungszeichen betrachtet.
Zeichenveränderliche sind allgemeine Veränderliche, die mit dem Symbol $ geschrieben werden (wie z.B. A, B, X und Y). Die Auswahl erfolgt aus diesem Bereich.

Beispiel: A$, B$, C$, X$, Y$

Zeichenveränderliche können miteinander verglichen oder addiert werden, wobei jedoch andere Operationen (wie Subtraktion, Multiplikation und Division) nicht möglich sind.

Beispiel: Wenn A$ = "123" und B$ = "456" ist
und der Ausdruck C$ = A$ + B$ gegeben ist,
ergibt sich für C$ gleich "123456"
(für C$ = B$ + A$ ergibt sich für C$ gleich "456123")

In diese Zeichenveränderliche kann eine Zeichenkette mit bis zu sieben Zeichen eingegeben werden. Neben den Zeichenveränderlichen steht auch eine exklusive Zeichenveränderliche ($) zur Verfügung, in die bis zu 30 Zeichen eingegeben werden können.

Beispiel: $ = "1234567890ABCDEFG"

Diese exklusive Zeichenveränderliche kann auch für Zeichenfunktionen (MID-Funktionen) verwendet werden, die später beschrieben sind. Die exklusive Zeichenveränderliche kann wesentlich vielseitiger als die anderen Zeichenveränderlichen verwendet werden.

■ Zuordnungsanweisung

BASIC-Zuordnungsanweisungen werden in der nachfolgend aufgeführten Form eingegeben.

Veränderliche = Numerischer Ausdruck

Bei BASIC-Zuordnungsanweisungen werden Ausdrücke mit einer arithmetischen Operation (+, −, ✱, /) an der rechten Seite als numerische Ausdrücke bezeichnet.

Beispiel: In Y = 2✱X+3 ist die rechte Seite 2✱X+3 ein numerischer Ausdruck. Das " = " bedeutet nicht "gleich", sondern "zugeordnet".

Beispiel: In Y = 2✱X+3 stellt die linke Seite die Veränderliche und die rechte Seite den numerischen Ausdruck dar. Im Gegensatz zur herkömmlichen Mathematik bedeutet dies nicht, daß Y an der linken Seite gleich 2✱X+3 an der rechten Seite ist; es bedeutet lediglich, daß das Ergebnis von 2✱X+3 der Veränderlichen Y zugeordnet ist. (Y = 2✱X+3 kann besser verstanden werden, wenn dieser Zusammenhang als Y ← 2✱X+3 angesehen wird.)

Beispiel für Zuordnungsanweisungen

A = B	Der Wert B wird A zugeordnet (der frühere Wert für A wird gelöscht).
N = 2✱M	Der doppelte Wert M wird N zugeordnet.
X = Y + Z . . .	Die Summe aus Y und Z wird X zugeordent.
I = I + 1	Der Wert von I + 1 wird I zugeordnet (d.h. der in Speicher I enthaltene Wert wird um 1 erhöht).

4-3 Einschreiben und Ausführung des Programms

■ Einschreiben des Programms

Das Speichern eines Programms im Speicher des Rechners wird als Einschreiben (oder Eintasten) des Programms bezeichnet.
Dieser Vorgang erfolgt durch Tasteneingabe auf der Tastatur.

1. Bezeichnung der Betriebsart WRT.
2. Bezeichnung des Programmbereiches.
3. Einschreiben der Programmsätze (Zeilen).
4. Der Programmbereich kann in 10 Teile unterteilt werden, d.h. es können die Programm P0 bis P9 eingegeben werden.

(1) Bezeichnung der Betriebsart WRT

Das Einschreiben des Programms erfolgt in der Betriebsart WRT.
Tastenfolge [MODE] [1] verwenden, wodurch der Schriftzug "WRT" in der Sichtanzeige erscheint.

(2) Bezeichnung des Programmbereiches

Um den Programmbereich zu bezeichnen, d.h. dem einzutastenden Programm eine Programm-Nummer zuzuordnen, die [S] Taste und danach eine der Zifferntasten [0] bis [9] drücken.

[S] [P0 0] → P0	[S] [P5 5] → P5
[S] [P1 1] → P1	[S] [P6 6] → P6
[S] [P2 2] → P2	[S] [P7 7] → P7
[S] [P3 3] → P3	[S] [P8 8] → P8
[S] [P4 4] → P4	[S] [P9 9] → P9

(3) Einschreiben der Programmsätze

Das Programm wird in Programmsätzen (Zeilen) eingegeben. Jeder Programmsatz kann bis zu 62 Zeichen enthalten (einschließlich der Zeilen-Nummer). Nach Eingabe der vollständigen Zeile ist die [EXE] Taste zu betätigen.

*** Funktion der [EXE] Taste.**

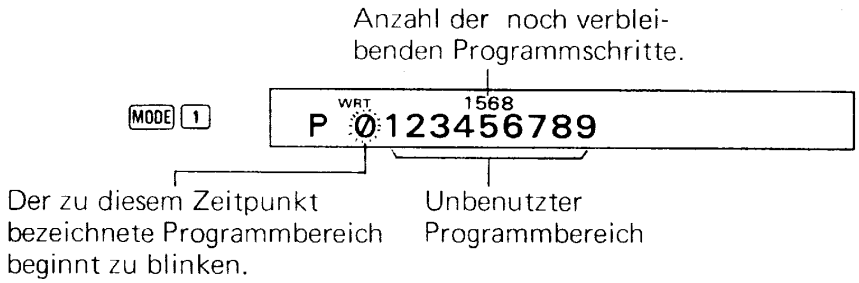
Die [EXE] Taste wird betätigt, um Programme einzutasten, Daten einzugeben oder das Ergebnis bei manuellen Rechnungen zu erhalten. Beim Programmieren ist die [EXE] Taste nach dem Eintasten jeder Programmzeile zu drücken, um die Programmzeile in den Programmspeicher des Rechners zu übertragen. Einschreiben, Änderungen, Ergänzungen bzw. Streichungen des Programms müssen jeweils mit Hilfe der [EXE] Taste abgeschlossen werden. Falls die [EXE] Taste nicht gedrückt wird, wird die eingetastete Anweisung nicht gespeichert.

Beispiel: Das folgende Programm ist mit der Programm-Nummer P0 einzugeben.

```
10 INPUT A,B
20 V=A+B
30 W=A-B
40 PRINT V,W
50 END
```

Bedienung:

1. Bezeichnung der Betriebsart WRT.



* Die Anzahl der Programmschritte ändert in Abhängigkeit von der angewählten Anzahl der Speicher und der bereits eingetasteten Anzahl der Programmschritte.

2. Bezeichnung des Programmbereiches P0.

[S] [PO Ø]

WRT 1568
P Ø123456789

3. Falls diese Programm-Nummer bereits belegt ist, das alte Programm löschen.

Im Weiteren weggelassen

CLEAR[EXE]

WRT 1568
P Ø123456789

(Ist kein Programm eingeschrieben, diesen Schritt auslassen.)

4. Zeile Nummer 10 eingeben.

10␣INPUT␣A[S][P]B[EXE]

1561
10 INPUT A,B

Bedeutet eine Leerstelle (kann weggelassen werden)

* Unbedingt dieser Taste drücken, wenn die Zeile geändert wird.

5. Zeile Nummer 20 eingeben.

20␣V[=]A[+]B[EXE]

1553
20 V=A+B

6. Zeile Nummer 30 eingeben.

30␣W[=]A[−]B[EXE]

1545
30 W=A−B

7. Zeile Nummer 40 eingeben.

40␣PRINT␣V[S][P]W[EXE]

1538
40 PRINT V,W

8. Zeile Nummer 50 eingeben.

50␣END[EXE]

1534
50 END

- Die END Anweisung verwenden, um ein Programm zu beenden. Bei einem Programm, wie es oben angegeben ist, kann auf die END Anweisung verzichtet werden; die END Anweisung ist jedoch in einem Programm erforderlich, wenn GOTO oder GOSUB Anweisungen verwendet werden.
- Zwischen der Zeilen-Nummer und den Operationsbefehlen bzw. zwischen den Operationsbefehlen und den Operanden sind Leerstellen vorgesehen, um das Ablesen der Anzeige zu erleichtern. In der BASIC-Programmiersprache haben diese Leerstellen keine Bedeutung, ausgenommen für die PRINT Anweisung.

- Die Zeilennummern sind in Zehnersprüngen im Bereich von 1 bis 9999 einzugeben, um auch nachträglich Ergänzungen problemlos einfügen zu können. Das Eingeben dieser Ergänzungen kann unsortiert erfolgen, da der Rechner selbst die richtige Reihenfolge herstellt (die Programmzeilen müssen also in der Ablaufreihenfolge numeriert sein, können aber in beliebiger Reihenfolge eingegeben werden).
- Die CLEAR Anweisung wie oben verwenden, um ein früher eingegebenes Programm zu löschen; die CLEAR A Anweisung dient dagegen für das Löschen aller früher eingegebenen Programme (PO bis P9).

■ Ausführung eines Programms

Der Ablauf eines Programms erfolgt in der Betriebsart RUN (die Tasten [MODE] [0] drücken . . . worauf "RUN" angezeigt wird).
Für die Ausführung eines Programms stehen zwei Verfahren zur Verfügung.

(1) Verfahren zur Programmausführung

1. Ausführung durch Programmbereich-Bezeichnung

Bei diesem Verfahren beginnt der Programmablauf mit der Bezeichnung des Programmbereiches, d.h. der Programm-Nummer.

[S] { [PO 0]) [P9 9] } (Zuerst die [S] Taste und danach eine der Tasten [PO 0] bis [P9 9] drücken.)

Beispiel: Das im vorhergehenden Beispiel aufgeführte Programm ist auszuführen.

Bedienung:

Betriebsart RUN (im Weiteren weggelassen)

[S] [PO 0]

?

RUN

* Das Fragezeichen "?" erscheint, da die INP Anweisung am Beginn des Programms geschrieben ist.

2. Ausführung mittels RUN Befehl

RUN [EXE] (Der Befehl RUN kann durch die Tastenfolge [R] [U] [N] oder [S] [RUN 8] eingegeben werden.)

?

* Gemäß vorhergehendem Beispiel wird nun "?" angezeigt. In dieser Eingabe-Bereitschaftsstellung ist mit Hilfe der [AC] Taste keine Löschung möglich. Nach der Tasteneingabe [MODE] [0] wird die Operation 2 durchgeführt.
Um mit der Ausführung fortzusetzen, nachdem der RUN Befehl eingegeben wurde, die Zeilen-Nummer eingeben und die [EXE] Taste drücken.

Beispiel: Ab Zeile 20 beginnen.

Bedienung: **RUN 20** [EXE]

* Bei Verfahren 1 muß die Programm-Nummer des auszuführenden Programms nicht bezeichnet werden, wogegen bei Verfahren 2 die Programm-Nummer des auszuführenden Programms eingetastet werden muß. (Falls eine falsche Programm-Nummer eingegeben wird, dann wird das im angewählten Programmspeicher enthaltene Programm durchgeführt.)

(2) Tasteneingabe während der Programmausführung

Die Tasteneingabe während der Programmablaufes erfolgt mit Hilfe der INPUT Anweisung und der KEY Funktionen. Die Tasteneingabe mittels KEY Funktion erfolgt nur jeweils mit einer Taste, wobei aber auch dann mit dem Programmablauf fortgesetzt wird, wenn keine Taste betätigt wird.
Für erforderliche Tasteneingaben mit einer INPUT Anweisung erscheint ein Fragezeichen "?" in der Sichtanzeige, worauf das Programm unterbrochen wird, um die Eingabe vornehmen zu konnen, Nach der Dateneingabe ist durch Drücken der [EXE] Taste mit dem Programmablauf fortzusetzen.

Beispiel: Das in P0 beschriebene Programm des obigen Beispiels ausführen.

Bedienung: Ausführung eines Programms

[S] [PO 0]

?

- Für dieses Programm sind 2 Veränderliche einzugeben. Zuerst wird der Zahlenwert für die Veränderliche A eingetastet.

47 [EXE]

```
?
```

- Danach ist der Zahlenwert für die Veränderliche B einzugeben.

69 [EXE]

```
 116                    STOP
```

[EXE]

```
-22                     STOP
```

Auf diese Art wird mit Hilfe der [EXE] Taste die mittels INPUT Anweisung programmierte Dateneingabe nach dem Eintasten des Zahlenwertes eingegeben.
Um in der Eingabe-Wartestellung den Programmablauf anzuhalten, die Tastenfolge [MODE] [0] verwenden, um den Rechner auf die Stoppfunktion zu schalten.

4-4 Redigieren eines Programms

- Mit Redigieren eines Programms werden die Vorgänge bezeichnet, die für logische Korrekturen des Programms, für Änderungen, Ergänzungen und Streichungen sowie für die Berichtigung von Zeilen-Nummern erforderlich sind.
- Das Redigieren eines Programms erfolgt mit Hilfe des LIST Befehls, um die einzelnen Zeilen abzurufen.
- Der LIST Befehl kann sowohl in der Betriebsart RUN als auch in der Betriebsart WRT verwendet werden; in der Betriebsart RUN wird jedoch der Programminhalt angezeigt. In der Betriebsart WRT kann das Programm redigiert werden.

(1) Anzeige des aufgelisteten Programms in der Betriebsart RUN.

(Anzeige jeweils für etwa 2 Sekunden)

Bedienung:

LIST [EXE]

(Der LIST Befehl kann durch die Tastenfolge [L] [I] [S] [T] oder [S] [LIST N] eingegeben werden.)

```
10 INPUT A,B
20 V=A+B
30 W=A-B
40 PRINT V,W
50 END
READY P0
```

Falls nicht alle Programmsätze abgerufen werden müssen, einfach die gewünschte Zeilen-Nummer eingeben.
Um das Programm ab Zeile 30 abzurufen:

Bedienung:

LIST 30 [EXE]

```
30 W=A-B
40 PRINT V,W
50 END
READY P0
```

* Während der Ausführung mittels LIST Befehl werden die einzelnen Programmsätze aufeinanderfolgend bis zum Ende des Programms angezeigt. Die [STOP] Taste drücken, um das Programm anzuhalten.
Um mit dem angehaltenen, mittels LIST Befehl ausgeführten Programm fortzusetzen, die [EXE] Taste drücken.

(2) Änderungen, Ergänzungen und Streichungen des Programms in der Betriebsart WRT

Die [MODE] [1] Tasten drücken und die Betriebsart WRT bezeichnen.

1. Änderungen

Unter Verwendung des LIST Befehls werden die einzelnen Programmsätze (Zeilen) ab der angewählten Zeilen-Nummer aufeinanderfolgend angezeigt, wenn jeweils die [EXE] Taste gedrückt wird. Falls keine Zeilen-Nummer eingetastet wird, erfolgt die Anzeige ab dem ersten Programmsatz.

a. Teilweise Änderung

Beispiel: Im vorhergehenden Beispiel ist in Zeile 20 die Addieranweisung "+" in eine Multiplikationsanweisung " ✕ " zu ändern.

Bedienung:

● Falls der Programmbereich nicht mit P0 bezeichnet wurde, die Programm-Nummer P0 eingeben.

[S] [PO] [0]

```
                    1534
P 123456789
```

● Mit Hilfe des LIST Befehls ist die Zeile Nummer 20 abzurufen.

LIST 20 [EXE]

```
                    1534
20 V=A+B_
```

● Den Kursor unter die zu ändernde Stelle bewegen (in diesem Beispiel unter "+").

[←] [←]

```
                    1534
20 V=A+B
```

* Wird die Kursortaste ([←] [→]) für länger als eine Sekunde gedrückt, dann wird der Kursor schnell in die gewünschte Richtung verschoben.

● Die Berichtigung vornehmen.

[✱] [EXE]

```
                    1534
30 W=A-B_
```

* Unbedingt die [EXE] Taste drücken, da sonst nur die Anzeige geändert, der neue Befehl jedoch nicht in das gespeicherte Programm eingeschrieben wird.

● Dadurch wurde die Zeile 30 geändert. Die [AC] Taste drücken, um die Anzeige zu löschen und die Änderung zu beenden.

[AC]

```
                    1534
_
```

* Durch Betätigung anderer Tasten in einer Zeile, in der keine Änderung erforderlich ist, werden die neuen Tastenbefehle eingeschrieben. Daher dürfen keine anderen Tasten als die [EXE] und [AC] Taste betätigt werden.

● LIST zur Änderung verwenden.

[MODE] [0]

LIST [EXE]

```
READY P0
  10 INPUT A,B
  20 V=A*B
  30 W=A-B
  40 PRINT V,W
  50 END
READY P0
```

b. Ändern einer vollständigen Zeile

Die Nummer der zu ändernden Zeile eingeben. (Dadurch wird die vorher eingegebene Zeilen-Nummer gelöscht.)

Beispiel: In Zeile 30 ist der Ausdruck "W=A−B" auf "W=V/2" zu ändern.

Bedienung: [MODE] [1]

```
                    1534
P 123456789
```

● Die neue Zeile Nummer 30 eingeben.

30 ␣ W [=] V [/] 2 [EXE]

```
                    1534
30 W=V/2
```

● Das aufgelistete Programm kontrollieren.

MODE 0
LIST EXE

```
READY P0
  10 INPUT A,B
  20 V=A*B
  30 W=V/2
  40 PRINT V,W
  50 END
READY P0
```

2. Ergänzung

Um eine Ergänzung einzugeben, eine Nummer verwenden, die zwischen den Nummern der beiden Anweisungen liegt, zwischen die sie eingefügt werden soll.

Beispiel: Die Ergänzung "U=V*2" ist zwischen Zeile 30 und Zeile 40 im obigen Beispiel einzugeben, wobei die Zeile 40 auf "PRINT V, W, U" zu ändern ist.

Bedienung: MODE 1

```
P 1534
123456789
```

● Um die Ergänzung zwischen den Zeilen 30 und 40 einzufügen, die Nummer 35 verwenden.

35 U = V * 2 EXE

```
1526
35 U=V*2
```

* Für das Einfügen einer Ergänzung zwischen den Zeilen 30 und 40 kann jede beliebige Zeilen-Nummer von 31 bis 39 verwendet werden.

● Um die Zeile 40 zu ändern, diese Zeile mit Hilfe der LIST einzufügen, die Nummer 35 verwenden.

LIST 40 EXE
→ S P U EXE
AC

```
1526
40 PRINT V,W
1524
50 END_
1524
_
```

● Das Programm kontrollieren, um sicherzustellen, daß die Ergänzung richtig eingefügt wurde.

MODE 0
LIST EXE

```
READY P0
  10 INPUT A,B
  20 V=A*B
  30 W=V/2
  35 U=V*2
  40 PRINT V,W,U
  50 END
PEADY P0
```

3. Streichung

a. Teilweise Streichung

Beispiel: Der Ausdruck "V" ist in Zeile 40 des vorhergehenden Beispiels zu streichen (löschen).

Bedienung: MODE 1

```
P 1524
123456789
```

● Gleich wie bei teilweiser Änderung ist die Zeile 40 mittels LIST Anweisung abzurufen.

LIST 40 EXE

```
1524
40 PRINT V,W
```

● Den Kursor unter den Ausdruck "V" (an die zu löschende Position) bewegen.

← ←

```
1524
40 PRINT V,W
```

● Die [DEL] Taste verwenden, um "V , " zu löschen.

[DEL][DEL]
[EXE]

```
          1524
40 PRINT W,U
          1526
50 END_
```

* Wird die [EXE] Taste nicht gedrückt, dann wird auch der Programminhalt nicht geändert.

[AC]

```
_         1526
```

* Unbedingt die [AC] Taste drücken, um den Änderungsbefehl für Zeile Nr. 50 zu löschen.

● Das aufgelistete Programm kontrollieren, um sicherzustellen, daß der gewünschte Ausdruck gelöscht wurde.

[MODE][0]
LIST[EXE]

```
READY P0
   10 INPUT A,B
   20 V=A*B
   30 W=V/2
   35 U=V*2
   40 PRINT W,U
   50 END
READY P0
```

b. Löschen einer gesamten Zeile

Die Nummer der zu löschenden Zeile eingeben.

Beispiel: Löschen von Zeile Nr. 30

Bedienung:

[MODE][1]

```
          1526
P 123456789
```

● Die Nummer der zu löschenen Zeile eingeben, z.B. 30.

30[EXE]

```
_         1534
```

● Darauf achten, daß die Zeile gelöscht wurde.

[MODE][0]
LIST[EXE]

```
READY P0
   10 INPUT A,B
   20 V=A*B
   35 U=V*2
   40 PRINT W,U
   50 END
READY P0
```

4. Berichtigung der Zeilen-Nummer

Beispiel: Das folgende Programm ist in den Programmspeicher P2 eingegeben.

```
10 INPUT N
20 M=N*N
30 L=SQR N
40 PRINT M,L
50 END
```

Die Zeile-Nummer 20 ist zwischen den Zeilen Nr. 30 und 40 einzufügen.

Bedienung:

[MODE][1][S][P2]

```
          1500
P _1 3456789
```

● Die Zeilen-Nummer 20 mittels LIST Befehl abrufen.

LIST 20[EXE]

```
          1500
20 M=N*N_
```

● Den Kursor unter die Ziffer 2 der Zeile 20 bewegen.

←←←←←←←←

```
                1500
20 M=N*N
```

● Die Nummer 20 auf 35 ändern.

35 EXE

```
                1495
30 L=SQR N
```

● Um die Änderung abzuschließen, die AC Taste drücken, um den Änderungsbefehl zu löschen.

AC

```
                1495
_
```

● Den LIST Befehl verwenden, um die Programmänderung zu kontrollieren

MODE 0

LIST EXE

```
READY P2
   10 INPUT N
   20 M=N*N
   30 L=SQR N
   35 M=N*N
   40 PRINT M,L
   50 END
READY P2
```

● Der Inhalt der Zeile 20 wurde als Zeile 35 zwischen den Zeilen 30 und 40 eingefügt, wobei jedoch die Zeile 20 noch vorhanden ist. Diese Zeile wird nicht benötigt und muß daher gelöscht werden.

MODE 0

20 EXE

```
                1495
P _1...3456789
                1503
_
```

● Damit ist die Änderung beendet. Mittels LIST Befehl nochmals den Programminhalt kontrollieren.

MODE 0

LIST EXE

```
READY P2
   10 INPUT N
   30 L=SQR N
   35 M=N*N
   40 PRINT M,L
   50 END
READY P2
```

■ Fehlerbeseitigung

(1) Programmfehlerbeseitigung

Das Fehlerbeseitigungssystem dieses Rechners läßt sich grob in die Fehlerbeseitigung mittles Rechner und die Fehlerbeseitigung im Dialogbetrieb mit dem Sichtgerät unterteilen.

Fehlerbeseitigungssysteme

- **I. Fehlerbeseitigung mittels Rechner**
  - **a. Vollständige Fehlerbeseitigung**
    Programmlogikprüfung.
  - **b. Teilweise Fehlerbeseitigung**
    Programmzeilen-Einheitsprüfung.
- **II. Fehlerbeseitigung im Dialogbetrieb mittles Sichtgerät**
  Mittels Sichtgerät ist der Programmablauf auf Fehler zu prüfen, indem das automatische Programmüberprüfungsprogramm der BASIC Programmiersprache verwendet wird.

Die Fehlerbeseitigung mittels Rechner wird während des Programmierens ausgeführt.
Nachfolgend ist die Fehlerbeseitigung im Dialogbetrieb mit dem Sichtgerät beschrieben.

(2) Fehlerbeseitigung im Dialogbetrieb

Jeder während des Programmablaufes festgestellte Fehler wird in der Sichtanzeige mittels Fehlerinformation angezeigt. Diese Fehler werden in Zeileneinheiten angezeigt und zwar in Form von BASIC Programmiersprachenfehlern in der Sichtanzeige. Anhand der im Sichtgerät angezeigten Fehlerinformationen kann die Fehlerbeseitigung im Dialogbetrieb unter Verwendung dieser Anleitung durchgeführt werden.
Die Bedeutung der Fehlerinformationen ist in der Fehleranzeigeliste auf Seite 67 aufgeführt.

Beispiel:

```
10 INPUT X
20 IF X≤0:PRINT"X≤0":GOTO 10
30 Y=X↑2+3*X+15
40 PRINT Y
50 END
```

Adresse 20 des obigen Programms beurteilt den Eingabebereich für das Potenzieren ($x \uparrow y$). Wenn $x < 0$ ist, kehrt das Programm an die Eingabeanweisung in Adresse 10 zurück.

Der Ausdruck **Y = X↑2+3X+15** wurde aus Versehen in Zeile 30 dieses Programms eingegeben.

Bedienung: Wenn dieses Programm ausgeführt wird, dann wird mit der Zeilen 10 INPUT Anweisung "?" angezeigt.

MODE 0 RUN EXE	?

- Nun ist die Zahl 45 einzugeben.

45 EXE	ERR2 P0-30

- Diese Fehlerinformation zeigt an, daß ein Anweisungsfehler in Zeile 20 vorliegt und der Programminhalt überprüft werden muß.

AC MODE 1	P 1540 123456789
LIST 30 EXE	30 1540 Y=X↑2+3X±

- Da die Anweisung "*" zwischen "3" und "X" in Zeile 30 fehlt, wird die Berichtigung durch Einfügen vorgenommen.

← S INS DEL	30 1540 Y=X↑2+3_X
* EXE	40 1539 PRINT Y

(3) Fehlerbeseitigung während der Programmausführung

Die Fehlerbeseitigung im Dialogbetrieb erfolgt anhand der Fehlerinformationen, die vom Rechner ausgegeben werden. Falls jedoch kein Fehler angezeigt wird, das Ergebnis aber nicht den Erwartungen entspricht, das Programm nochmals ausführen und die einzelnen Rechenvorgänge während des Programmablaufes kontrollieren.
Bei diesem Verfahren wird die STOP Anweisung für die Programmunterbrechung und die TRACE Betriebsart für die zeilenweise Fehlerbeseitigung verwendet.

• Fehlerbeseitigung mittels STOP Befehl

Beispiel: Das folgende Programm wird eingetastet.

```
10 Y=0
20 INPUT N,X
30 FOR I=1 TO N
40 Y=Y+X*X
50 NEXT I
60 PRINT Y
70 END
```

Um den Wert Y in der FOR•NEXT Schleife abzulesen, das Ergebnis jeder Programmschleife mittels STOP Anweisung kontrollieren.

Bedienung: Die STOP Anweisung sollte unmittelbar nach der Berechnungsformel eingegeben werden, d.h. die STOP Anweisung ist zwischen Zeile 40 und Zeile 50 einzutasten.

MODE 1	P 1522 123456789
45 STOP EXE	1518 45 STOP

• Dadurch wird der Programmablauf nach Beendigung der Berechnung in Zeile 40 angehalten, worauf die Fehlerbeseitigung durchgeführt werden kann.

MODE 0	READY P0
RUN EXE	?
4 EXE	?
87 EXE	STOP _

Kursor blinkt

• Welchen Wert hat Y bei diesem Programmhalt?

Y EXE	7569 STOP

• Wenn das Programm fortgesetzt wird, dann tritt der nächste Programmhalt bei der nächsten STOP Anweisung ein, worauf der Wert für Y wiederum abgerufen werden kann.

EXE	STOP _
Y EXE	15138

• Durch Wiederholung dieser Vorgänge kann der Rechnungsablauf kontrolliert werden.
In diesem Beispiel wird ein sehr einfaches Programm verwendet; bei einem komplexen Programm ist es äußerst schwierig, den Rechnungsvorgang mittels Fehlerbeseitigung zu kontrollieren. Wenn daher die Veränderlichen mit Hilfe der STOP Anweisung kontrolliert werden, können Programmfehler einfach festgestellt und berichtigt werden.

● Fehlerbeseitigung in der (TRACE) Betriebsart

Die Programmausführung erfolgt in der TR Betriebsart (MODE 2 drücken), wobei jede Zeile einzeln ausgeführt wird und bei einem Programmhalt die Fehlerbeseitigung einfach durchgeführt werden kann.
Mit Hilfe des vorhergehenden STOP Befehls wollen wir nun in der TR Betriebsart eine Fehlerbeseitigung durchführen.

Bedienung:

Bezeichnung der RUN Betriebsart ..... MODE 0	READY P0
Bezeichnung der TR Betriebsart ..... MODE 2	READY TR P0
RUN EXE	P0-10 TR STOP
EXE	? TR
Kontrolle des Programmablaufes ..... STOP	? TR STOP
STOP	P0-20 TR STOP
Mit der Programmausführung fortsetzen ..... EXE	? "TR" und "STOP" werden nachfolgend weggelassen.
4 EXE	?
87 EXE	_
STOP	P0-20
EXE	P0-30
EXE	P0-40
Wert für Y ..... Y EXE	7569
EXE	P0-45
⋮	⋮

Diese Schritte wiederholen

Die Fehlerbeseitigung in der Betriebsart TR ist das optimale Verfahren zur Kontrolle des Programmablaufes, wobei einfach und bequem die Stellen aufgefunden werden können, an welchen Fehler vorhanden sind.

4-5-1. Programmsprünge und Programmschleifen

■ Sprungbefehle

Sprungbefehle können grob in zwei Befehlsarten eingeteilt werden. Eine wird als GOTO Anweisung bezeichnet und stellt einen unbedingten Sprungbefehl dar, wogegen die andere als Verzweigung oder bedingte Sprunganweisung bezeichnet und in Verbindung mit der IF Anweisung verwendet wird.

● GOTO Anweisung

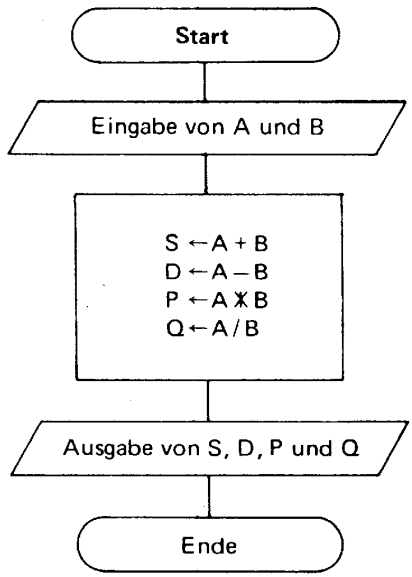
Eine GOTO Anweisung wird als unbedingter Sprungbefehl bezeichnet, da das Programm an der im Sprungbefehl angegebenen Sprungadresse (Zeilen-Nummer) fortgesetzt wird.

Beispiel 1: Eine GOTO Anweisung ist in das Programm einzufügen, um Summe, Differenz, Produkt und Quotient der Daten A und B zu berechnen.

Das grundlegende Programm ist nachfolgend aufgeführt.

```
10 INPUT A,B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRINT S,D,P,Q
70 END
```

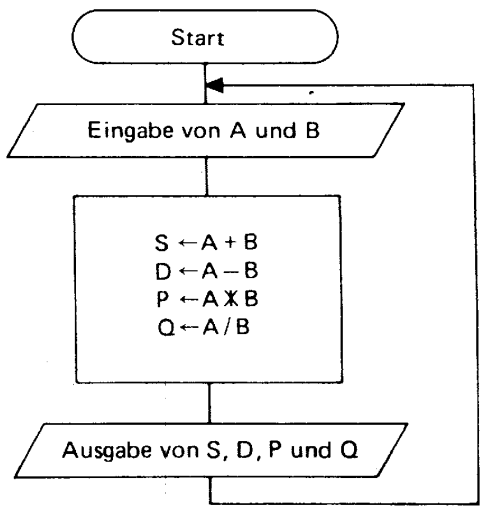
Flußdiagramm



Um dieses Programm auszuführen, müssen die Daten wiederholt in der Betriebsart RUN eingegeben werden. Die GOTO Anweisung ist daher in Zeile 70 und nicht in der Zeile END einzugeben, so daß das Programm an jene Zeile zurückkehrt, an der die Daten eingegeben werden müssen (Zeile 10 in diesem Programm).

Diese GOTO Anweisung (GOTO 10) stellt einen unbedingten Sprungbefehl auf Zeile 10 dar. Das Programm ist nachfolgend gezeigt.

```
10 INPUT A,B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRINT S,D,P,Q
70 GOTO 10
```



Wenn bei diesem Programm der erste Datenwert eingegeben wurde, dann wird der Rechner auf die Eingabewartefunktion (angezeigt durch ?) geschaltet, so daß der nächste Datenwert sofort eingegeben werden kann.

Ausführung: Eingabe der Datenwerte 15 und 3 sowie 903 und 43.

Bedienung:

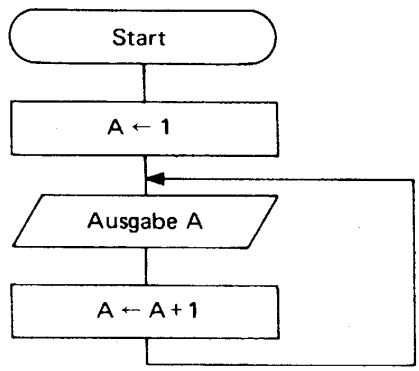
RUN EXE	?
15 EXE	?
3 EXE	18
EXE	12
EXE	45
EXE	5
EXE	?
903 EXE	?
43 EXE	946
EXE	860
EXE	38829
EXE	21

Bei diesem Programm wird durch die am Ende eingefügte GOTO Anweisung auf Zeile 10 "INPUT A, B" zurückgesprungen. Wenn in diesem Beispiel eine Zeilen-Nummer nach der GOTO Anweisung eingetastet wird, dann erfolgt der Programmsprung zu dieser Zeilen-Nummer. Falls jedoch anstelle der Zeilen-Nummer die Bezeichnung "#" und "0" bis "9" eingetastet wird, dann erfolgt der Programmsprung zur angegebenen Programm-Nummer.

Beispiel:
GOTO 10 (Sprunganweisung an Zeile 10 und Ausführung ab Zeile 10)
GOTO #5 (Sprunganweisung an Programm P5 und Ausführung von Programm P5)

Beispiel 2: Erstelle ein Programm, das den Zahlenwert für A in Schritten von 1 erhöht.

```
10 A=1
20 PRINT A
30 A=A+1
40 GOTO 20
```



Erläuterung:
Da A sequentiell im Zahlenwert in Schritten von 1 erhöht wird, muß A ein Anfangswert von 1 zugeordnet werden, d.h. "A=1" in Zeile 10.
Danach folgt die Anweisung PRINT A.
In Zeile 30 wird A das Ergebnis zugeordnet, daß sich ergibt, wenn 1 zu A addiert wird. Dieser Programmsatz lautet "A=A+1". Als nächstes folgt die Sprunganweisung GOTO, und zwar an Zeile 20 und nicht zum Programmbeginn. Diese Anweisung lautet GOTO 20.
Die GOTO Anweisung stellt daher einen unbedingten Sprungbefehl zur angegebenen Adresse dar.

Hinweis: Bei Verwendung einer GOTO Anweisung muß die richtige Adresse (Zeilen-Nummer) eingegeben werden. Falls die Zeilen-Nummer nicht der GOTO Anweisung angefügt wird, kommt es zu einem Programmfehler.

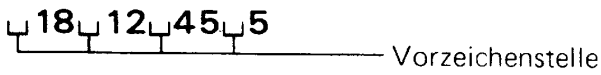
PRINT Anweisung
Die PRINT Anweisung in diesem Programm ändert die Anzeige in Abhängigkeit vom Begrenzungszeichen, das dem Operanden folgt.
Wenn z.B. 1, S, D, P und Q durch ein " , " (Komma) getrennt werden, dann stoppt die Anzeige nach S. Um nun D anzeigen zu können, muß die Taste EXE gedrückt werden. Allgemein kann gesagt werden, daß bei Trennung durch ein Komma die Ergebnisse einzeln angezeigt werden. Welche Anzeige ergibt sich, wenn

anstelle des " , " ein " ; " verwendet wird?
Das Ergebnis ist nachfolgend dargestellt.

Bedienung:

RUN[EXE]	?
15[EXE]	?
3[EXE]	18 12 45 5 STOP
[EXE]	?
903[EXE]	?
43[EXE]	946 860 388
	860 38829 21

In diesem Fall sorgt der " ; " (Strichpunkt) dafür, daß bei mehreren Ergebnissen diese sequentiell angezeigt werden.
Weiters erscheint in der Sichtanzeige nach jedem Ergebnis eine Leerstelle. Das "+" erscheint nicht an der Vorzeichenstelle, wird aber in der Rechnung berücksichtigt.



Wird das Beispiel 2 wie oben ausgeführt, dann wird das folgende Ergebnis erhalten.

Bedienung:

RUN[EXE]	1
[EXE]	2
[EXE]	3
[EXE]	4
⋮	⋮

Wird jedoch " ; " anschließend an "PRINT A" in Zeile 20 eingegeben,
20 PRINT A:
dann erhält man das folgende Ergebnis.

RUN[EXE]	1 2 3 4 5 6
	1 2 3 4 5 6
	2 3 4 5 6 7
	2 3 4 5 6 7
	3 4 5 6 7 8
	⋮

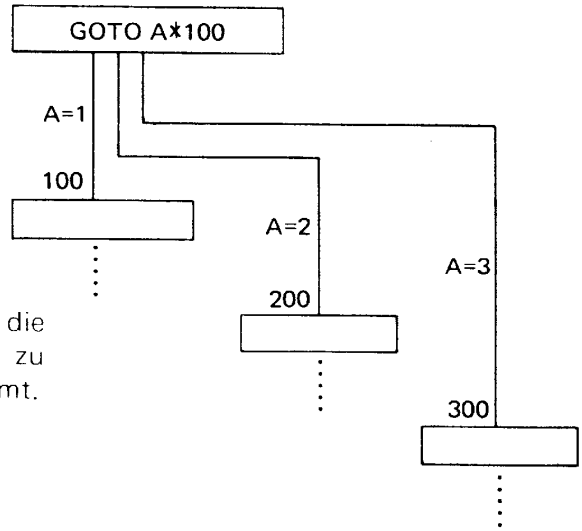
Mit Hilfe des Begrenzungszeichens " ; " werden daher die Ergebnisse aufeinanderfolgen angezeigt.

• Indirekter Sprungbefehl mittels GOTO Anweisung
Die vorhergehend beschriebene GOTO Anweisung diente zur Bezeichnung eines unbedingten Sprungbefehls an eine bestimmte Adresse. Bei einem indirekten oder bedingten Sprungbefehl wird die Ausführung eines Programmsprunges von der Erfüllung bestimmter Bedingungen abhängig gemacht (wie z.B. dem Wert einer Veränderlichen). Der bedingte Sprungbefehl wird verwendet, wenn der Sprungbefehl anhand der gegebenen Problemstellung nicht am Beginn des Programm geschrieben werden kann.
Der bedingte Sprungbefehl besteht aus:
GOTO Veränderliche oder numerischer Ausdruck
GOTO # Veränderliche oder numerischer Ausdruck

Dabei wird der Wert der Veränderlichen (A, B, X, Y usw.) oder der numerische Ausdruck (A+B, X+10 usw.) verwendet, um die Adresse des Sprungbefehls (Zeilen-Nummer oder Programm-Nummer) zu bezeichnen.

Beispiel 1: GOTO A✳100

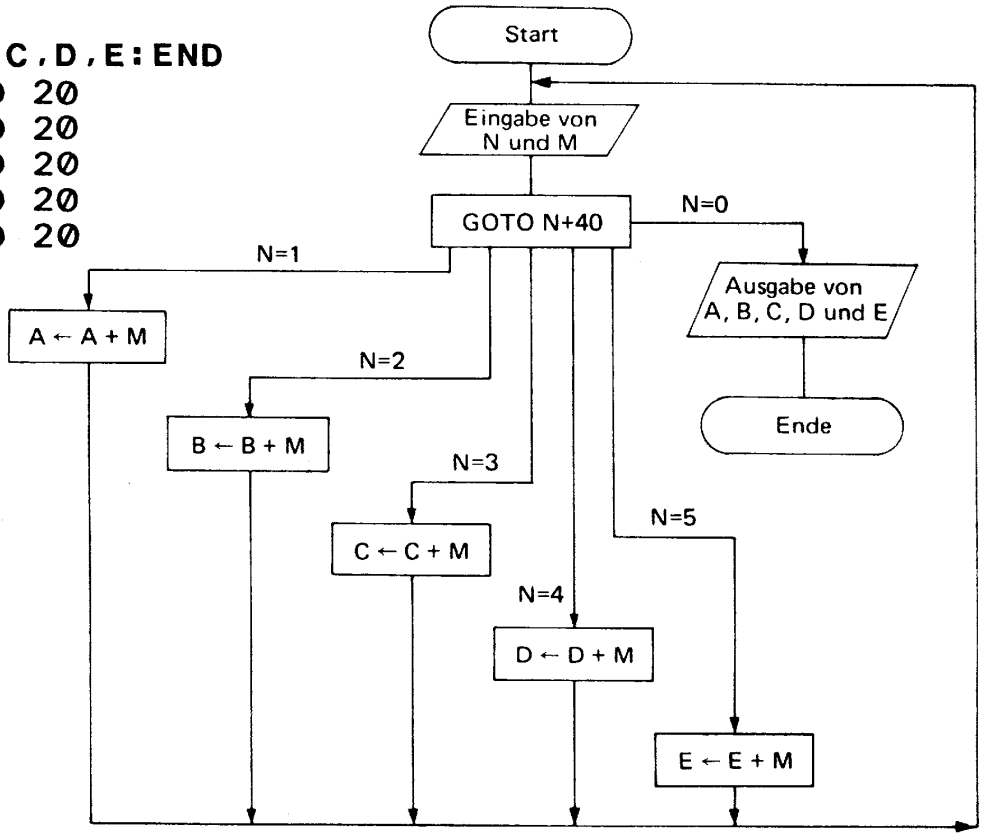
In diesem Programm, wenn A gleich 1, 2 oder 3 ist
Für A=1 bedeutet GOTO 100 einen Programmsprung an Zeile 100
Für A=2 bedeutet GOTO 200 einen Programmsprung an Zeile 200
Für A=3 bedeutet GOTO 300 einen Programmsprung an Zeile 300



Falls A einen anderen Wert als 1, 2 oder 3 aufweist, dann ist die Sprungadresse unbekannt, so daß es zu einem Fehler bzw. zu einem Programmsprung zu einer anderen Zeilen-Nummer kommt.

Beispiel 2: Erstelle ein Sortierprogramm mit bedingtem Sprungbefehl (sortiert nach 5 Gruppen). Das Summensortierprogramm lautet wie folgt:

```
10 VAC
20 INPUT N,M
30 GOTO N+40
40 PRINT A,B,C,D,E:END
41 A=A+M:GOTO 20
42 B=B+M:GOTO 20
43 C=C+M:GOTO 20
44 D=D+M:GOTO 20
45 E=E+M:GOTO 20
```



Der "VAC" Befehl in Zeile 10 dient zum Löschen der im Speicher gespeicherten Daten (Dateninhalt wird 0). In diesem Programm ist ein vorhergehendes Löschen erforderlich, um die Eingabedaten der Zeilen 41 bis 45 summieren zu können. Mit der nächsten INPUT Anweisung wird die Code-Nummer (N) und der Gewinnbetrag eingegeben.
Unter Verwendung der Zeilen 30 GOTO Anweisung wird mit der Code-Nummer (N) 1 der Programmsprung an Zeile 41 durchgeführt, d.h. den Gesamtgewinnen der Code-Number 1.

Im Falle der Code-Nummer 2 erfolgt der Programmsprung an Zeile 42. Auf diese Art werden die Gewinne (M) in die Code-Nummern 1 bis 5 unterteilt.

```
N=1 : GOTO 41 →41    A=A+M : GOTO 20 Ausführung
N=2 : GOTO 42 →42    B=B+M : GOTO 20 Ausführung
N=3 : GOTO 43 →43    C=C+M : GOTO 20 Ausführung
N=4 : GOTO 44 →44    D=D+M : GOTO 20 Ausführung
N=5 : GOTO 45 →45    E=E+M : GOTO 20 Ausführung
```

Ist daher N gleich 0, dann erfolgt der Sprung an Zeile 40. Die sortierten Ergebnisse A, B, C, D und E werden angezeigt und abgeschlossen. In diesem Programm muß N mit 0 bis 5 eingegeben werden; jede andere Zahl führt zu Fehler.
Wollen wir das obige Programm mit tastächlichen Zahlenwerten durchrechnen.
Die Rechnungen sind aufeinanderfolgend einzugeben, wobei die Summen sortiert werden sollen (innerhalb von 5 Gruppen).

Code	Gewinn
3	2870
2	1960
5	3850
5	1250
1	2500
2	2310
3	1850
5	4370
3	5360
1	2220
2	1450
4	6120
1	3100

Code	Gewinn
1	7820
2	5720
3	10080
4	6120
5	9470

Bedienung:

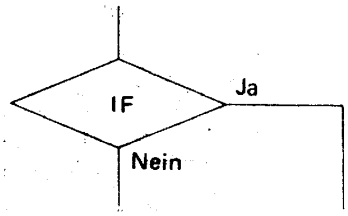
RUN EXE	?
3 EXE	?
2870 EXE	?
2 EXE	?
1960 EXE	?
⋮	⋮
1 EXE	?
3100 EXE	?
0 EXE	?
0 EXE	7820
EXE	5720
EXE	10080
EXE	6120
EXE	9470

* Der in den Zeilen 40 bis 45 dieses Programms verwandte " : " wird als "Mehrfachanweisung" bezeichnet. Er ermöglicht das Eintasten von verschiedenen Befehlen in einer Zeile. Wenn die Befehle aufeinanderfolgend ausgeführt werden, können die Zeilen-Nummern entfallen, so daß Programmschritte eingespart werden können. Mit einer Mehrfachanweisung können viele Befehle kombiniert werden; die Anzahl der in eine Zeile einzutastenden Zeichen (einschließlich der Zeilen-Nummer) ist jedoch auf 62 Zeichen begrenzt.

Hinweis: In einer Mehrfachanweisung kann der VAC Befehl (Speicherlöschbefehl) nicht verwendet werden.

• IF Anweisung (Wenn-Anweisung)

Eine IF Anweisung stellt eine Verzweigung mit einem bedingten Sprungbefehl dar, dessen Ausführung von der Erfüllung bestimmter Bedingungen abhängt.
Diese Verzweigung wird wie folgt im Flußdiagramm dargestellt.

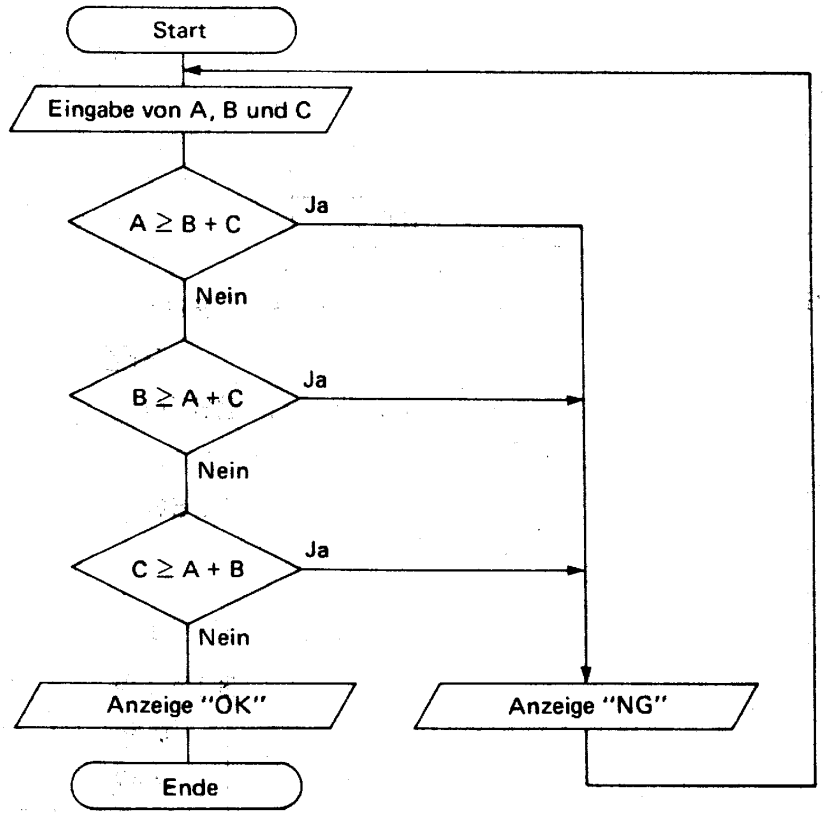


Diese Verzweigung bedeutet, daß bei Erfüllung der IF (Wenn) Bedingung der "JA" Weg, bei Nichterfüllung der "NEIN" Weg eingeschlagen wird. Die IF Anweisung ist daher eine Verzweigung, an der eine Entscheidung getroffen werden muß, um anhand des bisherigen Ergebnisses mit dem richtigen Programmzweig fortzusetzen.
Diese IF Anweisung wird für den Abschluß einer Programmschleife verwendet, wenn die Anzahl der Daten nicht bekannt ist, oder wenn die nächste Operation in Abhängigkeit von dem vorhergehenden Ergebnis geändert wird usw.

Beispiel 1: Eingabe der Längen von drei Seiten eines Dreiecks und Bestimmung, ob mit diesen drei Seiten ein Dreieck gebildet werden kann.

Das Programm ist nachfolgend aufgeführt.

```
10 INPUT A,B,C
20 IF A≥B+C THEN 70
30 IF B≥A+C THEN 70
40 IF C≥A+B THEN 70
50 PRINT "OK"
60 END
70 PRINT "NG"
80 GOTO 10
```



Für dieses Programm müssen drei Datenwerte eingegeben werden. Bei einem Dreieck muß die Summe von zwei Seiten immer größer als die dritte Seite sein. Die Seiten werden miteinander verglichen; wenn kein Dreieck gebildet werden kann, erscheint "NG" in der Sichtanzeige und das Programm kehrt in die Dateineingabe-Wartestellung zurück. Kann ein Dreieck gebildet werden, dann erscheint "OK" in der Sichtanzeige und das Programm wird abgeschlossen.

Die IF Anweisung wird aus den nachfolgenden Ausdrücken gebildet.

IF Vergleichsausdruck THEN Zeilen-Nummer (numerischer Ausdruck) oder #*n* (*n*=0 bis 9).

oder

IF Vergleichsausdruck; Befehls- oder Zuordnungsanweisung

Der nach der IF Anweisung eingegebene Vergleichsausdruck vergleicht die rechte Seite mit der linken Seite eines durch Gleichheitszeichen (oder Ungleichheitszeichen) getrennten Ausdruckes; wird die Bedingung

erfüllt (JA), dann wird mit "THEN" oder " ;" fortgesetzt. Wird die Bedingung nicht erfüllt (NEIN), dann erfolgt die Fortsetzung mit der nächsten Zeile. Wenn z.B. in Zeile 20 A gleich oder größer als die Summe von B und C ist, dann kann kein Dreieck gebildet werden, so daß "THEN 70", d.h. der Sprungbefehl an Zeile 70 durchgeführt wird. Das "THEN" enthält die Funktion der GOTO Anweisung.

Ist nach dem "THEN" eine Zeilen-Nummer angegeben, dann erfolgt der bedingte Sprung an diese Zeilen-Nummer. Bei eingetastetem " #" und 0 bis 9 erfolgt der Sprung an Programm (P0 bis P9).
Wenn die Vergleichsbedingung mit JA beantwortet und anstelle des Sprungbefehls eine Befehls- oder Zuordungsanweisung gewünscht wird, dann wird " ;" anstelle von "THEN" verwendet.
In diesen Vergleichsausdrücken können Konstante, Veränderliche, numerische Ausdrücke, Zeichen-Konstante und Zeichen-Veränderliche verwendet werden.

A > 10	Veränderliche und Konstante (JA, wenn A größer als 10 ist)
X ≥ Y	Veränderliche und Veränderliche (JA, wenn X gleich oder größer als Y ist)
N = L + 3	Veränderliche und numerischer Ausdruck (JA, wenn N gleich oder größer als die Summe aus L und 3 ist)
A$ = "XYZ"	Zeichen-Veränderliche und Zeichen-Konstante (JA, wenn die Zeichenfolge in A$ gleich "XYZ" ist)
P$ ≠ Q$	Zeichen-Veränderliche und Zeichen-Veränderliche (JA, wenn die Zeichenfolge in P$ ungleich zur Zeichenfolge in Q$ ist)

* Ein Vergleich von Veränderlichen mit Zeichen-Veränderlichen ist nicht möglich.
* Der Vergleich der Zeichenfolgen erfolgt nach dem ASCII Code (USACII).

"THEN" oder " ;" können unterschiedlich verwendet werden, abhängig von den nachfolgenden Anweisungen.

```
THEN 150        (Zeilen-Nummer)
THEN #9         (Programm-Nummer)
; PRINT A
; Z = X + Y
```

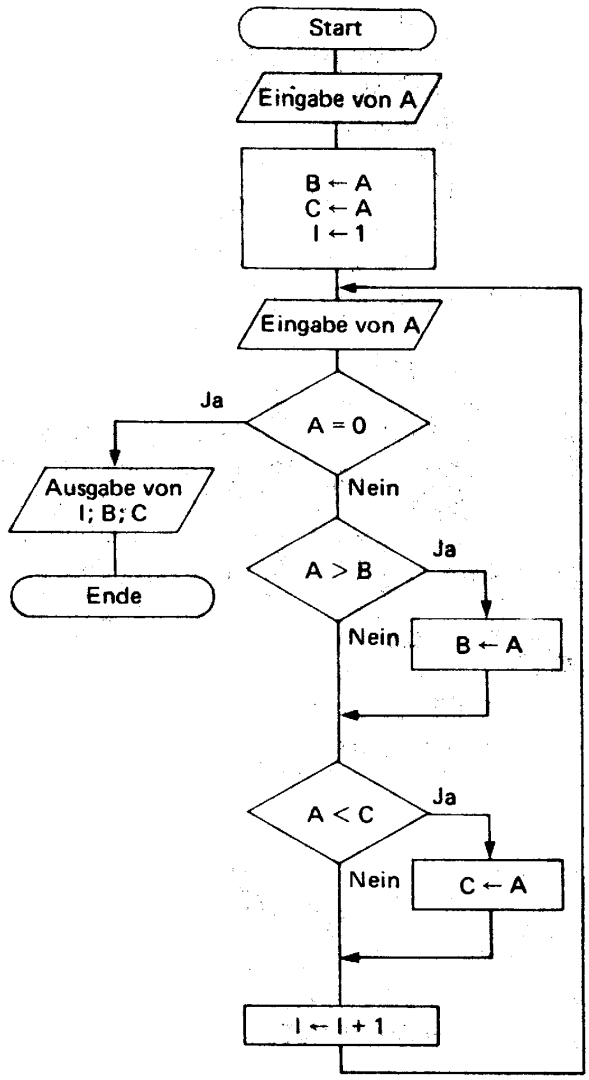
Anwendung der PRINT Anweisung
Im vorhergehenden Programm werden "OK" und "NG" verwendet, um das Ergebnis des Vergleiches anzuzeigen. Dies ist die Anwendung der PRINT Anweisung.
Falls PRINT A eingetastet wird, wird der numerische Wert der Veränderlichen A angezeigt.
Wird PRINT "A" eingetastet, dann wird nur der Buchstabe A angezeigt.
Mit anderen Worten, in Anführungszeichen aufgeführte Ausdrücke werden als Zeichen behandelt und angezeigt.

Beispiel:			
	PRINT A	PRINT "A"	PRINT "ANTWORT"
	⇓	⇓	⇓
	10 (wenn A = 10)	A	ANTWORT
	PRINT X	PRINT "X"	PRINT "N=" ;N
	⇓	⇓	⇓
	23 (wenn X = 23)	X	N =␣15 (wenn N = 15)

Beispiel 2: Nachfolgend ist ein Programm zur Berechnung des Maximal- und Minimalwertes aufgeführt.

```
10 INPUT A
20 B=A
30 C=A
40 I=1
50 INPUT A
60 IF A=0 THEN 110
70 IF A>B;B=A
80 IF A<C;C=A
90 I=I+1
100 GOTO 50
110 PRINT I;B;C
120 END
```

Die INPUT Anweisung in Zeile 10 wird für die Eingabe der ursprünglichen Daten verwendet. Die ursprünglichen Daten bestehen aus dem Maximal- und dem Minimalwert. Diese werden daher gemeinsam mit den Daten der Zeile 20 und der Zeile 30 verarbeitet; so daß die ursprünglichen Daten dem Maximalwert B und dem Minimalwert C zugeordnet werden.
In Zeile 40 wird eine Veränderliche I für die Zählung der Datenzahl verwendet. Daher wird "1" als ursprünglicher Datenwert eingegeben. INPUT Anweisung in Zeile 50 wird verwendet, um die zweiten und weitere Daten einzugeben. Der Vorgang wird daher durch die GOTO Anweisung in Zeile 100 wiederholt. Wird der Datenwert "0" mit Hilfe der IF Anweisung in Zeile 60 eingegeben, dann wird damit der Programmablauf abgeschlossen.
Mit anderen Worten, wenn "0" mit der INPUT Anweisung in Zeile 50 eingegeben wird, dann erfolgt der Sprung an die PRINT Anweisung zur Ausgabe des Ergebnisses der Zeile 110.
Die IF Anweisungen in den Zeilen 70 und 80 beurteilen, ob die Eingabedaten größer oder kleiner als der Maximalwert bzw. Minimalwert der bereits eingegebenen Daten sind, und ersetzen diese gegebenenfalls.
Nach Eingabe alle Daten wird mit Hilfe der Zeile 110 die Datenzahl sowie der Maximal- und der Minimalwert angezeigt.

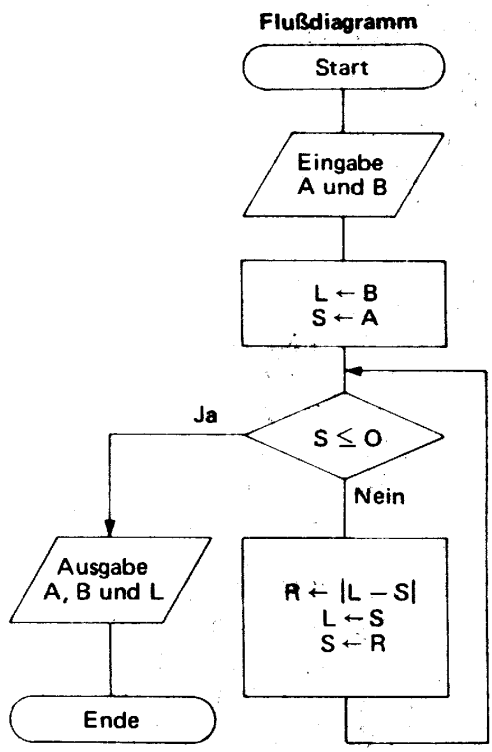


• Verschiedene GOTO und IF Anweisungen

Hier möchten wir einige Beispiele für die Anwendung der GOTO und IF Anweisung betrachten.

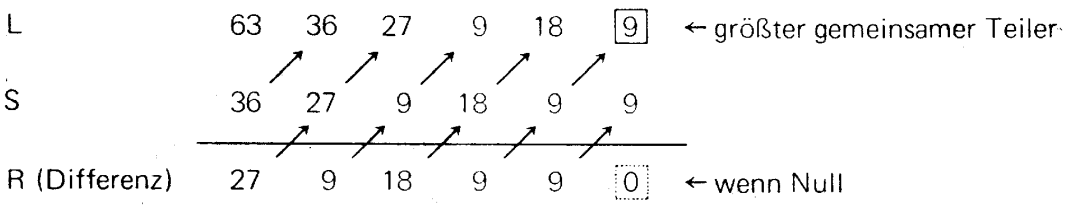
Beispiel 1: Programm zur Bestimmung des größten gemeinsamen Teilers nach dem euklidschen Algorithmus.

```
 10 INPUT A,B
 20 L=A
 30 S=B
 40 IF S≤0 THEN 90
 50 R=ABS(L-S)
 60 L=S
 70 S=R
 80 GOTO 40
 90 PRINT A;B;L
100 END
```



Nach dem euklidschen Algorithmus wird der zweite Datenblock vom ersten Datenblock subtrahiert, worauf das Ergebnis (Differenz) vom zweiten Datenblock subtrahiert wird. Danach ist diese Differenz vom vorhergehenden Ergebnis zu subtrahieren. Dieser Vorgang ist zu wiederholen, bis die Differenz Null beträgt. Sobald dieser Fall eintritt, entspricht die vorhergehende Differenz dem größten gemeinsamen Teiler. Der größte gemeinsame Teiler für die Zahlen 63 und 36 ist nachfolgend gezeigt.

Veränderliche



In diesem Programm werden die Veränderlichen zu Bestimmung der Differenz mit L bzw. S bezeichnet. Die Differenz wird vorübergehend R zugeordnet; danach werden L und S durch S bzw. R ersetzt, worauf der gleiche Vorgang nochmals durchgeführt wird (Programmschleife).
Die IF Anweisung in Zeile 40 bestimmt, ob S (d.h. die Differenz nach dem Austausch) Null ist oder nicht. Falls die Differenz Null ist, dann stellt L den größten gemeinsamen Teiler dar, so daß die Daten A und B und ihr größter gemeinsamer Teiler angezeigt werden. Beträgt die Differenz nicht Null, dann wird aufgrund der GOTO Anweisung in Zeile 80 wiederholt die Differenz gebildet. Die IF Anweisung wird dabei für die Entscheidung verwendet, welcher Befehl (welche Operation) anhand des Rechenergebnisses durchzuführen ist.
Auch die Austauschoperationen in den Zeilen 50 bis 70 müssen besonders beachtet werden. Die Daten A und B sind ursprünglich L und S zugeordnet, wobei ihre Differenz vorübergehend R zugeordnet ist. Danach wird die Differenz zwischen L uns S nochmals bestimmt, wobei L und S durch S bzw. R ersetzt werden. Dieser Vorgang wird wiederholt, bis eine Differenz von Null enhalten wird.

Der Zusammenhang zwischen L, S und R ist nachfolgend dargestellt.

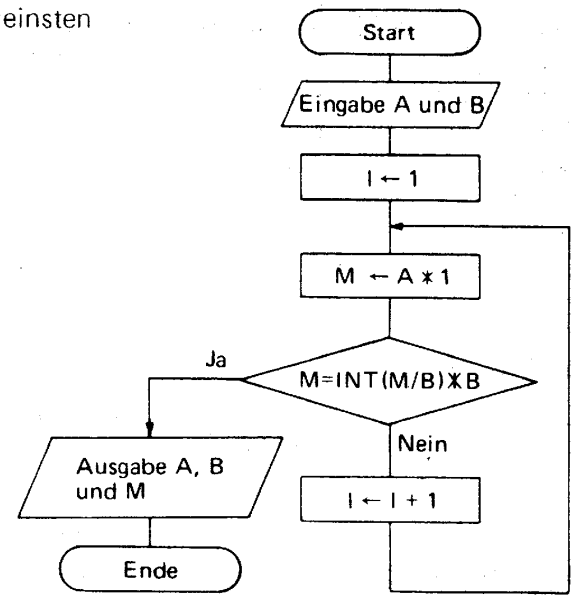
	L	S	R	
Ursprünglich	63	36	0	
Subtraktion (erstes Mal)	63	36	27	R = ABS (L–S)
Austausch (erstes Mal)	36	27	27	L = S, S=R
Subtraktion (zweites Mal)	36	27	9	R = ABS (L–S)
Austausch (zweites Mal)	27	9	9	L = S, S=R
Subtraktion (drittes Mal)	27	9	18	R = ABS (L–S)
Austausch (drittes Mal)	9	18	18	L = S, S=R
⋮		⋮		⋮
Subtraktion (sechstes Mal)	9	9	0	R = ABS (L–S)
Austausch (sechstes Mal)	9	0	0	L = S, S=R

↑
größter gemeinsamer Teiler

Hinweis: Falls die Reihenfolge des obigen Vorganges geändert wird, dann geht die Bedeutung der Austauschvorgänge verloren.

Beispiel 2: Erstellung eines Programms zur Bestimmung des kleinsten gemeinsamen Vielfachen.

```
10 INPUT A,B
20 I=1
30 M=A*I
40 IF M=INT(M/B)*B THEN 70
50 I=I+1
60 GOTO 30
70 PRINT A,B,M
80 END
```



Bei diesem Programm sind zuerst die Daten A und B einzugeben, wobei der ursprüngliche Wert für A mit eins, zwei, drei usw. multipliziert wird, bis das Ergebnis dieser Multiplikation dem kleinsten Vielfachen des Wertes B entspricht, wie es durch die IF Anweisung geprüft wird.
Die IF Anweisung in Zeile 40 vergleicht die Veränderliche "M" mit dem numerischen Ausdruck "INT (M/B) * B". Mit anderen Worten, die Veränderliche "M" wird mit dem Ergebnis des numerischen Ausdrucks "INT (M/B) * B" verglichen. Dieses Verfahren ist gleich wie:

```
N=INT(M/B)*B
IF M=N THEN 70
```

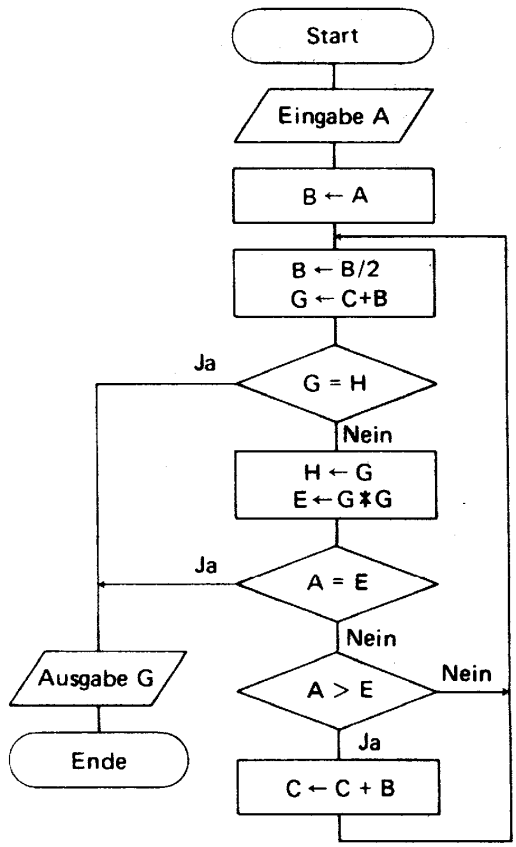
[INT (M/B) ist ein Funktionsbefehl, bei dem die Dezimalstellen von dem Ergebnis von (M/B) abgeschnitten werden, um eine ganze Zahl zu erhalten.]

Das hier verwandte Verfahren spart aber einige Programmzeilen und damit Programmschritte im Speicher ein.

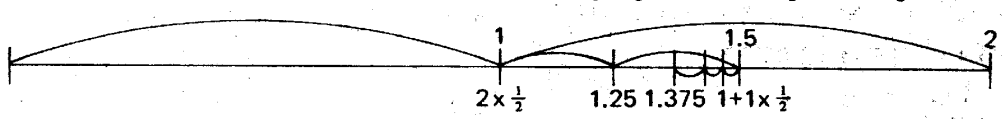
Danach wird der Multiplikator I jeweils um eins erhöht, mit dem Datenwert multipliziert und anhand der in Zeile 60 aufgeführten GOTO Anweisung verglichen.
Die verwendeten Veränderlichen sind A, B, M und I. A und B sind die beiden Eingabedaten. Da M ein Mehrfaches von A ist, kann der Wert von M leichter verstanden werden.
Der Multiplikator I wird jeweils um eins erhöht, um ein neues Mehrfaches zu erzeugen. Er wird häufig als Veränderliche verwendet, die mit jedem Durchlaufen der Programmschleife um eins erhöht wird.

Beispiel 3: Erstellung eines Programms zur Berechnung der Quadratwurzel gemäß Dichotomie. Das Programm ist nachfolgend aufgeführt.

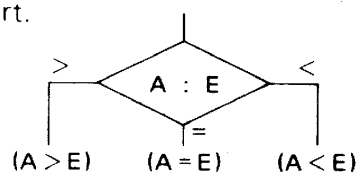
```
10 VAC
20 INPUT A
30 B=A
40 B=B/2
50 G=C+B
60 IF G=H THEN 120
70 H=G
80 E=G*G
90 IF A=E THEN 120
100 IF A>E;C=C+B
110 GOTO 40
120 PRINT G
130 END
```



Gemäß Dichotomie wird zuerst die Hälfte des Datenwertes A bestimmt, wonach das Quadrat des halben Datenwertes gebildet und mit dem Wert von A verglichen wird. Stimmen die beiden Werte nicht überein, dann wird die Hälfte des halben Datenwertes gebildet, wiederum quadriert und mit dem Wert A verglichen, bis ungefähre Übereinstimmung erzielt ist. Dieser Rechenvorgang ist nachfolgend dargestellt.



In diesem Programm wird der "VAC" Befehl in Zeile 10 zum Löschen des Speichers verwendet. Danach erfolgt die Eingabe des Datenwertes A. Dieser Datenwert wird später in dem Vergleichsvorgang der IF Anweisung verwendet. Direkte Änderungen sind nicht möglich. Änderungen werden daher durchgeführt, in dem eine Zuordnung zu B erfolgt und B danach geändert wird. Die IF Anweisung wird hier verwendet, um den ungefähren Wert in Zeile 60 zu bestimmen.
Der obere Grenzwert der berechneten Stellen wird bestimmt. Danach werden G und H verwendet, um den Punkt zu bestimmen, an dem der höchste Grenzwert erreicht wird; wenn das Ergebnis gleich dem vorhergehenden Ergebnis ist, wird die Operation abgeschlossen. Die IF Anweisung in Zeile 90 ist gleich wie die IF Anweisung in Beispiel 2. Falls die Quadratwurzel geteilt werden kann, dann ist das Quadrat des hier bestimmten Ergebnisses gleich dem Datenwert A, so daß das Ergebnis angezeigt wird.
Die IF Anweisung in Zeile 100 bestimmt, welches der beiden gleich geteilten Teile des Ergebnisses der Lösung (ungefährer Wert) entspricht.
Die IF Anweisungen in diesem Programm werden anders als früher verwendet, die Funktion der einzelnen IF Anweisungen ist aber gleich wie früher.
Das Programm ist etwas kompliziert, so daß eine Kombination von IF-Anweisungen erforderlich ist.
Die IF Anweisungen in den Zeilen 90 und 100 vergleichen A und E miteinander. Das Flußdiagramm für diesen Vorgang ist nachfolgend aufgeführt.



Die BASIC Programmiersprache weist aber keine IF Anweisung mit dreifacher Verzweigung auf. Es muß daher eine Kombination von IF Anweisungen mit doppelter Verzweigung verwendet werden.
Die IF Anweisungen haben unterschiedliche Auswirkungen, wenn sie an verschiedenen Stellen im Programm verwendet werden, wie es in den Beispielen 1 und 2 gezeigt wurde.
In Beispiel 1 wird eine IF Anweisung am Beginn der Programmschleife verwendet, so daß ein Sprung zum ENDE des Programms vor der Ausführung der Operation erfolgen kann.
In Beispiel 2 werden jedoch die Operationen ausgeführt und erst danach auf Erfüllung bestimmter Bedingungen kontrolliert.
Diese beiden Verfahren zeigen, daß in Abhängigkeit von den Daten die IF Anweisung entweder an Beginn einer Programmschleife (Beispiel 1) oder am Ende der Operation (Beispiel 2) geschrieben werden kann.

■ FOR·NEXT Anweisung

Die FOR·NEXT Anweisung wird verwendet, wenn die Anzahl der zu durchlaufenden Schleifen bekannt ist.

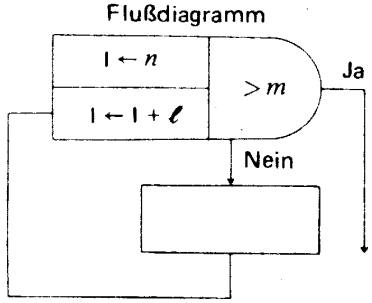
■ Funktion der FOR·NEXT Anweisung

Die FOR·NEXT Anweisung setzt sich wie folgt zusammen..

FOR Veränderliche = *n* TO *m* STEP *ℓ*

Ursprünglicher Wert — *n*
Endwert — *m*
Schritt ⇒ Veränderliche oder numerischer Ausdruck — *ℓ*

Flußdiagramm



Operationsbefehl des numerischen Ausdruckes usw.

NEXT Veränderliche

Diese Anweisung befehlt den zwischen "FOR" und NEXT" geschriebenen Vorgang zu wiederholen, wobei die Veränderliche in Schritten von ℓ von n bis m zu ändern ist. Wenn die Veränderliche auf m geändert wurde, setzt das Programm mit der nach der "NEXT" Anweisung folgenden Zeile fort.
Das nachfolgende Programm zeigt ein Beispiel, wie ein bestimmter Vorgang auszuführen ist, während die Veränderliche I in Schritten von 2 von 1 auf 10 erhöht wird.

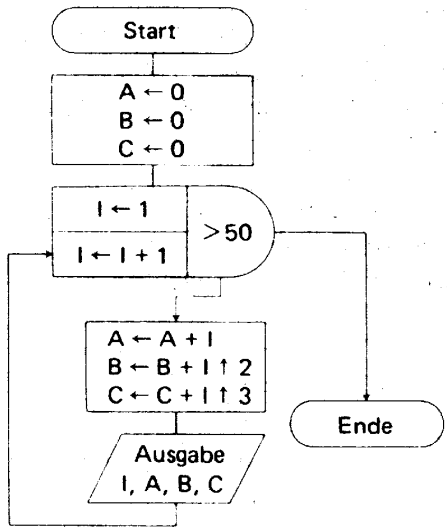
```
FOR I = 1 TO 10 STEP 2
.....
NEXT I
```

Die STEP Anweisung kann weggelassen werden, wenn die Veränderliche in Einerschritten erhöht werden soll.

FOR I = 1 TO 10 STEP 1 ist daher gleich wie
FOR I = 1 TO 10

* Die FOR•NEXT Anweisung erhöht die Veränderliche nur in Schritten mit einem vorbestimmten Wert und kann die Veränderliche nicht vermindern. Falls eine Veränderliche von 10 auf 1 zu vermindern ist, dann muß eine negative Schrittbezeichnung wie "STEP−1" erfolgen.

Beispiel: Erstelle eine Tabelle für $\sum_{i=1}^{n} i$, $\sum_{i=1}^{n} i^2$ und $\sum_{i=1}^{n} i^3$ mit n im Bereich von 1 bis 50.



Dieses Programm ist nachfolgend aufgeführt.

```
10 VAC
20 FOR I=1 TO 50 STEP 1
30 A=A+I
40 B=B+I↑2
50 C=C+I↑3
60 PRINT I,A,B,C
70 NEXT I
80 END
```

Dieses Beispiel kann nur mit Hilfe der FOR•NEXT Anweisung durchgeführt werden. Die Summen von i, i^2 und i^3 ($\sum_{i=1}^{n}$) werden angezeigt, wobei n in Einerschritten von 1 bis 50 erhöht wird.
Die Vorgänge zwischen "FOR ....." in Zeile 20 und "NEXT ....." in Zeile 70 werden 50 Mal wiederholt, wenn I in Einerschritten von 1 auf 50 erhöht wird.
Dieses Beispiel kann mit Hilfe einer IF Anweisung durchgeführt werden.

```
10 VAC
20 I=1
30 A=A+I
40 B=B+I↑2
50 C=C+I↑3
60 PRINT I,A,B,C
70 I=I+1
80 IF I=51:END
90 GOTO 30
```

Vergleichen wir die Beispiele mit der FOR•NEXT Anweisung mit den Beispielen mit der IF Anweisung, dann kann die Funktion der FOR•NEXT Anweisung klar verstanden werden.

Mit anderen Worten, die FOR•NEXT Anweisung kombiniert die Prüffunktion einer IF Anweisung mit der GOTO Anweisung und der Schrittfunktion.

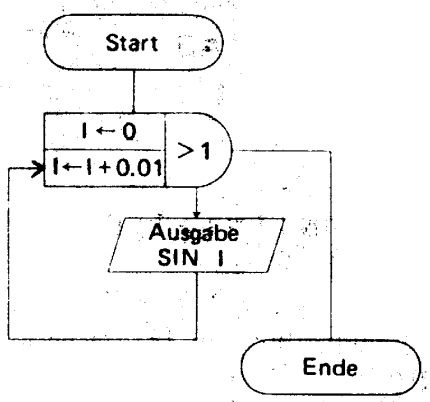
```
20 FOR I=1 TO 50 STEP 1 ⇨ { 20 I=1
                           { 70 I=I+1

80 NEXT I ⇨ { 80 IF I=51:END
            { 90 GOTO 30
```

Die FOR•NEXT Anweisung enthält also eine Schritt- und eine Prüffunktion, so daß sie besonders vorteihaft eingesetzt werden kann, wenn die Anzahl der zu durchlaufenden Schleifen bekannt ist.

Beispiel 2: Erstellung eines Programms zur Auflistung der Sinusfunktion von 0 bis 1 in Schritten von 0.01.

Dieses Programm ist nachfolgend aufgeführt.

```
10 MODE 4
20 FOR I=0 TO 1 STEP 0.01
30 PRINT SIN I
40 NEXT I
50 END
```



In diesem Beispiel wird die Sinusfunktion von 0 bis 1 in Schritten von 0.01 berechnet. Für die FOR•NEXT Anweisung können 0.01 Schritte verwendet werden, da dieser Rechner intern ein Dezimalsystem verwendet. SIN I kann jeweils für den Datenwert berechnet werden, der mit Hilfe der Eingabeanweisung in Zeile 30 eingegeben wird.
Mit "MODE 4" in Zeile 10 wird "Altgrad" als Winkelargument bezeichnet. "MODE 5" und "MODE 6" dienen für die Bezeichnung des Winkelarguments im Bogenmaß bzw. in Neugraden.

- **Verschiedene Programmschleifen**

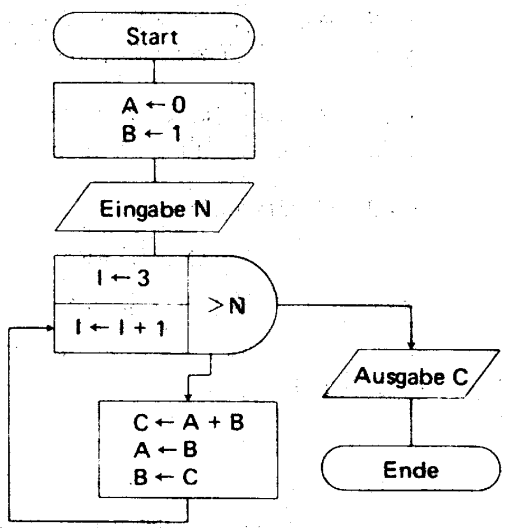
Beispiel 1: Bestimme die *n*-te Zahl einer Zahlenreihe nach Fibonacci.

Eine Fibonacci Serie ist eine Zahlenreihe, die aus ganzzahligen Zahlen besteht, wobei jede Zahl jeweils der Summe der beiden vorhergehenden Zahlen entspricht. Die Summe aus erster und zweiter Zahl enstpricht also der dritten Zahl, die Summe aus zweiter und dritten Zahl ist gleich der vierten Zahl usw.

$$\underbrace{0\ ,\ 1\ ,\ 1\ ,\ 2\ ,\ 3\ ,\ 5\ ,\ 8\ ,\ 13,\ 21,\ 34}_{0+1\quad 1+1\quad 1+2\quad 2+3},$$

Dieses Programm ist nachfolgend aufgeführt.

```
10 A=0:B=1
20 INPUT N
30 FOR I=3 TO N
40 C=A+B
50 A=B
60 B=C
70 NEXT I
80 PRINT C
90 END
```



Die Veränderlichen A, B und C stellen die Veränderlichen der Zahlenreihe dar. A ist der Wert der vorhergehenden Zahl, B ist der Wert der nachfolgenden Zahl und C stellt die Summe der beiden Zahlen dar (die der Zahl B nachfolgende Zahl).
Ein ursprünglicher Wert von 0 ist A, ein solcher von 1 ist B zugeordnet. Dann beginnt die Operation mit der Berechnung der dritten Zahl.
Mit der FOR•NEXT Anweisung wird mit der Programmschleife bei 3 begonnen, bis der gewünschte Punkt (N-te Zahl) erreicht ist. Der Ausgangswert für die FOR•NEXT Anweisung muß nicht notwendigerweise bei 1 beginnen.

Die Austauschvorgänge in den Zeilen 40 bis 60 erfordern besonderer Aufmerksamkeit. Die Reihenfolge kann nicht geändert werden. Sobald die Summe aus A und B in C eingegeben ist, wird B in A und C in B übertragen.
Wenn diese Reihenfolge nicht eingehalten wird, kann der Austausch nicht richtig durchgeführt werden.
Die Verwendung dieser FOR•NEXT Anweisung unterscheidet sich von dem vorhergehenden Beispiel. Daher wird die Veränderliche N verwendet, da der Endwert durch eine spätere Eingänge geändert wird.

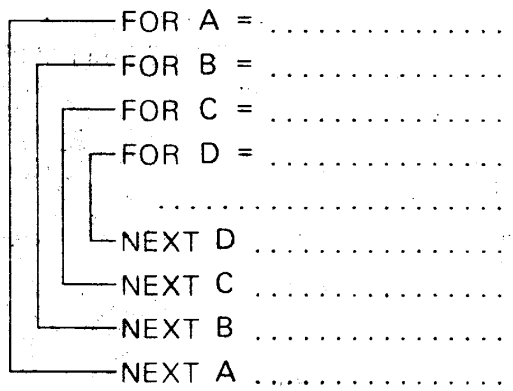
Dieses Beispiel kann mit einer IF Anweisung durchgeführt werden. Ein Beispiel für ein Programm mit einer IF Anweisung ist nachfolgend aufgeführt.

```
10 A=0:B=1:I=3
20 INPUT N
30 C=A+B
40 A=B
50 B=C
60 IF I≠N;I=I+1:GOTO 30
70 PRINT C
80 END
```

In diesem Programm erhöht die IF Anweisung in Zeile 60 die Veränderliche I und beeinflußt die Funktion der GOTO Anweisung. Die Ausgangswerte werden mit Hilfe der Mehrfachanweisung in Zeile 10 eingestellt.

• Verschachtelung

Die FOR•NEXT Schleifen können in bis zu 4 Ebenen verschachtelt werden.

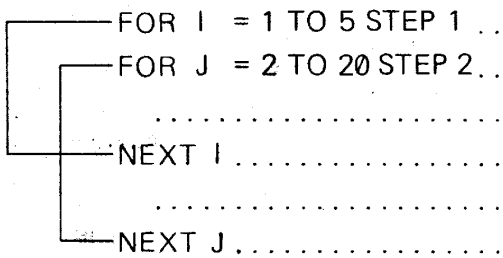


Dies ist ein Beispiel für eine Verschachtelung auf vier Ebenen.
Eine FOR•NEXT Anweisung wird mit einer anderen FOR•NEXT Anweisung wie gezeigt eingegeben.

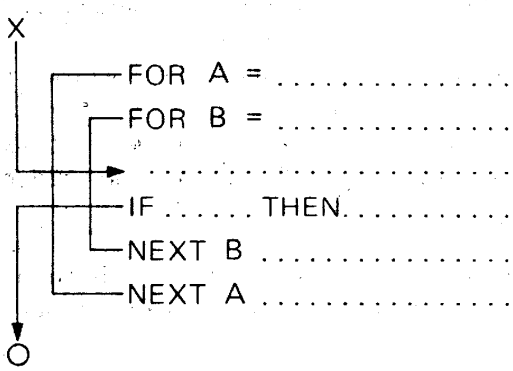
Beim Verschachteln von mehreren Schleifen ist darauf zu achten, daß für jede FOR Anweisung für eine Veränderliche auch eine NEXT Anweisung eingegeben wird.

Die Verschachtelung ist dabei so auszuführen, daß die Schleife jeweils geschlossen sind. Ein Überlappen der FOR•NEXT Schleifen, wie es im nachfolgenden Beispiel aufgeführt ist, ist nicht möglich.

Solche FOR•NEXT Schleifen können nicht verwendet werden.

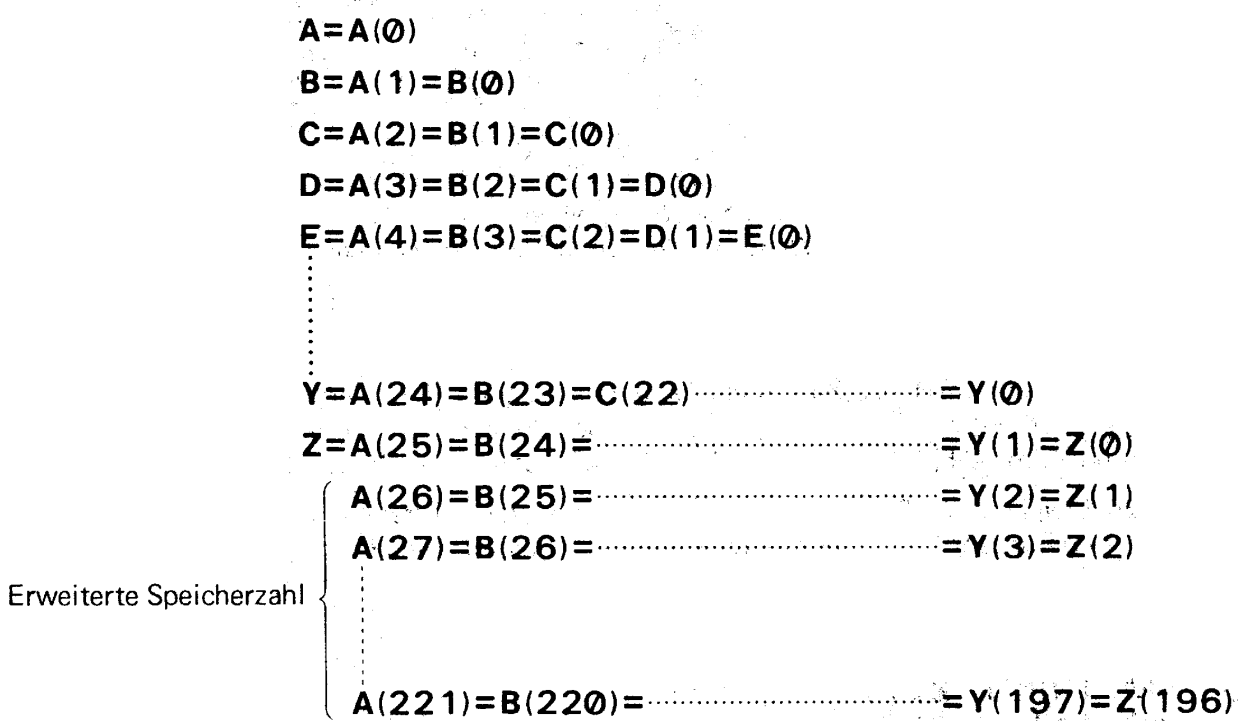


Weiters können Befehle aus einer FOR•NEXT Schleife ausgegeben, nicht aber in eine FOR•NEXT Schleife eingegeben werden.



4-5-2 Datenfelder

Für eindimensionale Datenfelder (Listen) werden die Buchstaben A(*i*), B(*i*) usw. verwendet. Da diese Datenfelder sowohl mit den normalen 26 Speichern als auch bei erweiterter Speicherzahl verwendet werden können, sind die folgenden Datenfelder-Kompositionen zu beachten.



Wenn Datenfelder auf diese Weise verwendet werden, dann kann auch der gleiche Speicher in Abhängigkeit vom Datenfeldargument mehrmals verwendet werden; es ist daher darauf zu achten, daß der gleiche Speicher in einem Programm nur einmal eingesetzt wird.

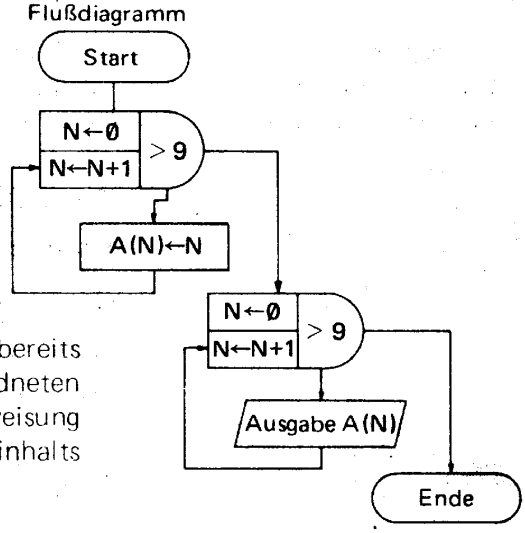
Beispiel:

Kann gleichzeitig verwendet werden ..... A, B, C, F(Ø), F(9)
Kann nicht gleichzeitig verwendet werden..... F, G, A(5), A(6)

Abhängig von der Größe des Datenfeldes ist die Erweiterung der Speicherzahl richtig auszuführen.

Beispiel 1: Erstelle ein Programm für die Anzeige des Wertes i in einem eindimensionalen Datenfeld A(i), wenn dieser von 0 auf 9 ändert.

Flußdiagramm



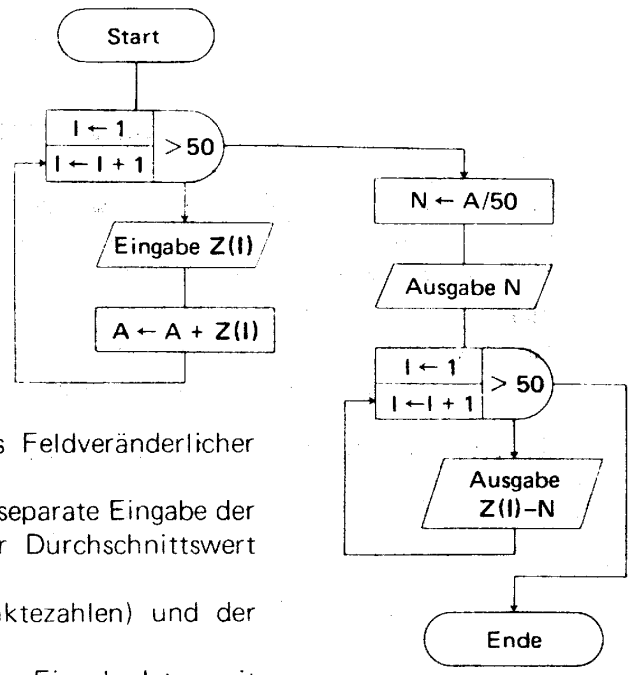
Dieses Programm ist nachfolgend aufgeführt:

```
10 FOR N=0 TO 9
20 A(N)=N
30 NEXT N
40 FOR N=0 TO 9
50 PRINT A(N)
60 NEXT N
70 END
```

Die FOR·NEXT Anweisungen in den Zeilen 10 bis 30 wurden bereits früher erläutert. In Zeile 20 wird jedoch A(N) mit einem zugeordneten Wert N verwendet. Auf ähnliche Weise ist die FOR·NEXT Anweisung der Zeilen 40 bis 60 zu verwenden, um den Wert des Datenfeldinhalts mittels PRINT Anweisung in Zeile 50 anzuzeigen.

Auf diese Weise speichert das Zahlenfeld Daten als separate Veränderliche, indem einfach die Elemente ausgetauscht werden, ohne daß die Bezeichnung der Veränderlichen geändert wird. Diese Funktion kann daher besonders vorteilhaft mit der FOR·NEXT Anweisung verwendet werden.

Beispiel 2: Erstelle ein Programm für die Anzeige der Differenz zwischen der durchschnittlichen Punktezahl und der individuellen Punktezahl, indem die von 50 Studenten erzielten Punktezahlen eingegeben werden.




```
10 A=0
20 FOR I=1 TO 50
30 INPUT Z(I)
40 A=A+Z(I)
50 NEXT I
60 N=A/50
70 PRINT N
80 FOR I=1 TO 50
90 PRINT Z(I)-N
100 NEXT I
110 END
```

Für dieses Programm werden die Punktezahlen mittels Feldveränderlicher Z(I) in Zeile 30 eingegeben.
Die Summe wird mit Hilfe der Zeile 40 erhalten. Für die separate Eingabe der Punktezahl wird ein Datenfeld verwendet, worauf der Durchschnittswert berechnet wird.
Die Differenz zwischen den gespeicherten Daten (Punktezahlen) und der durchschnittlichen Punktezahl wird angezeigt.
Auf diese Art kann ein Programm mit einer Vielzahl von Eingabedaten mit Hilfe eines Datenfeldes auf sehr wenige Programmschritte begrenzt werden.
Wird kein Datenfeld verwendet, dann ist das Programm wie folgt einzutasten:

```
Z(1) ←——→ A
Z(2) ←——→ B
Z(3) ←——→ C
  :        :
```

Die Eingabeanweisung lautet dabei:

```
INPUT  A,B,C,.....
```

Es ist daraus ersichtlich, daß eine Trennung der Daten einen großen Zeitaufwand erfordert.
Da in diesem Programm die Datenfelder Z(1) bis Z(50) verwendet werden, muß die Anzahl der Speicher auf 50 erweitert werden.

4-5-3 Ein-/Ausgabebefehle

In diesem Kapitel werden die mit diesem Rechner verwendeten Ein-/Ausgabebefehle beschrieben.

■ Eingabebefehle

Ein Eingabebefehl ist ein Befehl, der für die Eingabe von Daten während des Programmablaufes dient und mit Hilfe der INPUT Anweisung oder den KEY Funktionen eingegeben wird.

• INPUT Anweisung

Die INPUT Anweisung wird verwendet, um manuell Daten für die Veränderlichen eines Programmes einzugeben, wenn während des Programmablaufes in der Eingabewartebedingung "? " in der Sichtanzeige angezeigt wird. .

Die INPUT Anweisung besteht aus

INPUT ["Zeichenfolge"], Veränderliche, ["Zeichenfolge"], Veränderliche .....

Diese Zeichenfolgen können entfallen; falls sie jedoch einmal eingegeben wurden, dann werden sie jeweils für der Eingabewartebedingung angezeigt. Diese Zeichenfolgen können also auch als Hinweis benutzt werden. Die Veränderlichen sind numerische Veränderliche, Zeichen-Veränderliche oder exklusive Zeichen-Veränderliche ($), denen die Eingabedaten zugeordnet werden.

Beispiel:

Bei
```
INPUT A
```
?

Bei
```
INPUT"A=",A
```
A=?

Durch Eingabe der Daten und Drücken der [EXE] Taste während der durch die INPUT Anweisung abgerufenen Eingabewartebedingung wird auf den nächsten Programmschritt weitergeschaltet.

Beispiel: Ein Hinweis ist der INPUT Anweisung für das Programm zur Bestimmung der Summe, der Differenz, des Produktes und des Quotientens von zwei Veränderlichen hinzuzufügen.

Dieses Programm ist nachfolgend gezeigt.

```
10 INPUT "A=",A,"B=",B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRINT S,D,P,Q
70 END
```

Nachfolgend ist der Programmablauf dargestellt.

Bedienung:

RUN[EXE]	A=?
45[EXE]	B=?
23[EXE]	68
[EXE]	22
[EXE]	1035
[EXE]	1.956521739

Auf dieser Art und Weise werden die in der INPUT Anweisung eingegebenen Zeichenfolgen als Hinweise angezeigt, um die Dateneingabe zu erleichtern.

Falls bei dieser INPUT Anweisung Zeichendaten für die numerischen Veränderlichen eingegeben werden, tritt Fehler ein. Wird der Fehler mittels AC Taste gelöscht, dann wird wiederum "? " angezeigt und der Rechner auf die Dateneingabebedingung geschaltet.
Auch die Eingabe für numerische Veränderliche kann als numerischer Ausdruck eingegeben werden.
Aufgrund dieser INPUT Anweisung wird " ? " angezeigt und der Computer nimmt die Eingabewartestellung ein. Nach Eingabe des Datenwertes und dem Drücken der EXE Taste wird der Programmablauf fortgesetzt. Die Eingabewartestellung kann durch Drücken der AC Taste nicht freigegeben werden. Falls Sie daher das Programm an beliebiger Stelle unterbrechen möchten, die Tastenfolge MODE 0 verwenden.

* Mittels INPUT Anweisung einzugebende Daten sind numerische Werte oder die Ergebnisse numerischer Ausdrücke (für numerische Veränderliche) sowie Zeichenketten (für Zeichenveränderliche).

Im Falle von INPUT A

Numerischer Wert .....123 EXE → A=123

Ergebnis eines numerischen Ausdrucks .....14 * 25 EXE → A=350

Im Falle von INPUT B$

Zeichenkette .....ABC EXE → B$=ABC
789 EXE → B$=789

Andere numerische Veränderliche können ebenfalls als Eingang für numerische Veränderliche verwendet werden.

Im Falle von INPUT A (mit X = 987654)

Veränderliche .....X EXE → A=X
=987654

• KEY Funktionen

Mit dieser Funktion wird während des Programmablaufes mit jedem Druck einer Taste ein Zeichen eingelesen. Im Gegensatz zu der INPUT Anweisung kann diese Funktion nicht während der Eingabewartebedingung durchgeführt werden. Das Programm läuft normal ab, so daß nichts eintritt, wenn keine Taste betätigt wird.

Zeichen-Veränderliche = KEY

Das mittels KEY Funktion eingelesene Zeichen wird der angewählten Zeichen-Veränderlichen zugeordnet.

Beispiel:

```
10 A$=KEY: IF A$=" " THEN 10
20 IF A$="A" THEN 100
30 IF A$="B" THEN 200
40 IF A$="C" THEN 300
50 GOTO 10
    :
    :
```

Dieses Programm dient nur für die Dateneingabe, wobei mit Hilfe der KEY Funktion in Zeile 10 und unter Verwendung der nächsten IF Anweisung geprüft wird, ob die Zeichendatenanzeige eingetastet wurde.
Auch wenn die EXE Taste nicht verwendet wird, kann mit dieser KEY Funktion nur die erste Tasteneingabe gelesen werden. Dabei wird aber nicht wie bei der INPUT Anweisung das Programm angehalten. In Verbindung mit der nächsten IF Anweisung wird daher die Eingabewartebedingung erhalten.
In den Zeilen 20 bis 40 wird der Inhalt der Zeichen-Veränderlichen verglichen und die Adresse des Sprungbefehls bestimmt. Auf diese Weise kann mittels KEY Funktion nur das Zeichen einer Taste eingelesen werden.

■ Ausgabebefehle

In dem Ausgabebefehl ist eine PRINT Anweisung enthalten, um die Rechenergebnisse anzuzeigen.

• PRINT Anweisung

Eine PRT Anweisung setzt sich wie folgt zusammen.

PRINT [**Ausgaberegelfunktion** / **CSR**] { **Numerischer Ausdruck** / **Zeichen-Ausdruck** } [{ , / ; } .....]

{ } Jeder der in diesen Klammern geschriebenen Inhalte kann verwendet werden.
[ ] Der in diesen Klammern geschriebene Inhalt kann weggelassen werden.

Diese PRINT Anweisung zeigt den Wert des numerischen Ausdrucks oder die numerische Veränderliche bzw. die zwischen " " gestellte Zeichenfolge oder den Inhalt der Zeichenveränderlichen an.

Beispiel:

```
10 INPUT "A=",A,"B=",B
20 C=INT (A/B)
30 D=A-B*C
40 PRINT C;"...";D
50 GOTO 10
```

Dieses Programm berechnet den Rest neben dem Quotienten, wenn A durch B geteilt wird.
Die Werte der Veränderlichen werden gleich wie in Zeile 40 angezeigt.
Die zwischen " " aufgeführten Posten werden als Zeichen angezeigt.
Weiters ist zu erwähnen, daß das Verfahren der Anzeige von " ," und " ;" zwischen den Veränderlichen und Zeichenfolgen unterschiedlich ist.
Im Falle von " ;" wird der nächste Datenwert nach der vorhergehenden Anzeige angezeigt.
Im Falle von " ," wird der nächste Datenwert angezeigt, nachdem die vorhergehende Anzeige gelöscht wurde.
Um die Anzeige im Falle von "," fortzusetzen, die EXE Taste drücken.

■ Ausgabekontrollfunktionen

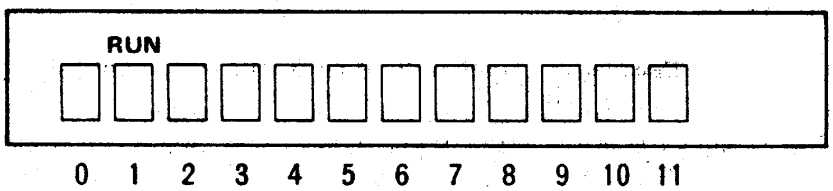
Die Ausgabekontrollfunktionen bestimmten die Position.

• CSR Funktion

Diese Kontrollfunktion wird in einer PRINT Anweisung verwendet. Sie bestimmt die Position der Ausgabe in der Sichtanzeige (12 Positionen) unter Verwendung der PRINT Anweisung. Das Format der CSR Funktion ist nachfolgend dargestellt.

PRINT CSR Numerischer Ausdruck (Der Dezimalteil des numerischen Ausdrucks wird abgeschnitten und der Wert ist 0 bis 11.)

Dabei wird der Wert des numerischen Ausdrucks verwandt, um zu bestimmen, an welcher Stelle (gerechnet von links in der Sichtanzeige) mit der Ausgabe von Daten begonnen wird.
Nachfolgend ist dargestellt, wie die Einheiten in der Sichtanzeige gezählt werden.



Beispiel:

```
10 INPUT X
20 PRINT X;CSR 5;X↑2
30 GOTO 10
```

Mit diesem Programm wird das Quadrat von bereits eingegebenen Daten erhalten, wobei sowohl das Ergebnis als auch die Daten angezeigt werden. Die CSR Funktion wird in der PRINT Anweisung in Zeile 20 verwendet.
Der anfängliche Datenwert "X" wird von links angezeigt, worauf der nächste Befehl "X↑2" mit Hilfe der CSR Funktion an der fünften Stelle von links angezeigt wird.

Wird dieses Programm ausgeführt, ergibt sich der folgende Ablauf.

Bedienung:

RUN EXE	?	
7 EXE	7	49
EXE	?	
45 EXE	45	2025
EXE	?	
852 EXE	852	725904

Da die beiden unterschiedlichen Ausgaben kontinuierlich an bestimmten Stellen angezeigt werden, wird das Ablesen erleichtert.

* Falls " ;" nach "CSR" im Programm geschrieben steht, dann wird dies von der vorbestimmten Position aus angezeigt. Wird dagegen " ," verwendet, dann erfolgt keine Anzeige, wenn nicht vorher die Anzeige einmal gelöscht wird. Die Anzeige erfolgt also nicht kontinuierlich.

• SET Anweisung

Die SET Anweisung bestimmt die Anzahl der Stellen für manuelle Rechnungen; hier möchten wir aber erläutern, wie die Bestimmung beim Erstellen des Programms durchgeführt wird.
In dem nachfolgend gezeigten Beispiel für die SET Anweisung, wird mit E die Anzahl der effektiven Stellen und mit F die Anzahl der Dezimalstelle bestimmt; mit N wird diese Bestimmung gelöscht.

SET { E *n* / F *n* / N } (*n* ist 0 bis 9)

Beispiel 1:

```
10 SET E4
20 MODE 4
30 FOR X=0 TO 180 STEP 5
40 PRINT SIN X
50 NEXT X
60 SET N
70 END
```

Dieses Programm berechnet den sin x in Schritten von jeweils 5 Grad für x gleich 0 bis 180 Grad.
Um eine Zahl mit 4 effektiven Stellen in diesem Beispiel zu erhalten, wird sin x mit Hilfe von "SET E4" berechnet.
Wird die SET Anweisung zu Beginn des Programms geschrieben, dann werden alle folgenden Anzeigen (Ergebnisausgaben) mit 4 effektiven Stellen erhalten.
Wird die Rechnung fortgesetzt, dann kann das Ergebnis jeweils nur mit 4 effektiven Stellen erhalten werden. Daher wird "SET N" in Zeile 60 verwendet, um diese Stellenbestimmung zu löschen.

Beispiel 2: Für das in Beispiel 1 gezeigt Programm ist die Dezimalstellenzahl auf 4 Stellen einzustellen.

```
10 SET F4
20 MODE 4
30 FOR X=0 TO 180 STEP 5
40 PRINT SIN X
50 NEXT X
60 SET N
70 END
```

Auf diese Art kann die Anzahl der Ausgabestellen mit Hilfe der SET Anweisung ähnlich wie bei manuellem Rechnen kontrolliert werden.

4-5-4 Zeichenfunktionen

Die Zeichenfunktionen beziehen sich auf die Zeichenveränderlichen (A$, B$, $ usw.). Diese Funktionen sind jedoch anders zu behandeln, da diese für die Eingabe von Zeichen anstelle von numerischen Werten dienen. Daher bestimmten diese Zeichenfunktionen die Größe der Zeichenveränderlichen, rufen einige Zeichen aus den Zeichenketten der Zeichenveränderlichen ab und übertragen die in den Zeichenveränderlichen gespeicherten Zahlen in Zahlenwerte.

LEN und MID

Eine LEN Funktion ist ein Befehl für die Zählung der Zeichenzahl in einer Zeichenveränderlichen. Damit kann die Größe der Zeichenveränderlichen ermittelt werden.
Eine MID Funktion ist ein Befehl zum Abruf einiger Zeichen aus der exklusiven Zeichenveränderlichen ($).
Sie wird für eine Neuordnung der Zeichenveränderlichen verwendet.
Da mittels LEN Funktion die Größe der Zeichenveränderlichen ermittelt werden kann, kann diese für die Zuteilung von Ausgabestellen für die Zeichenveränderlichen oder für eine Teilung bei nicht bestimmter Größe verwendet werden.

Beispiel:

```
10 A$="123"
20 B$="456"
30 C$=A$+B$
40 D=LEN(C$)
50 PRINT D
60 END
```

Dieses Programm zeigt die Größe (Anzahl der Zeichen) der Zeichenveränderlichen an, indem die Zeichenveränderlichen A$ und B$ addiert werden.
Mit Zeichenveränderlichen können nur Additionen durchgeführt werden. Da aber die Größe der Zeichenveränderliche im Ergebnis nicht bekannt ist, wird die LEN Funktion zur Bestimmung der Größe dieser Addition verwendet.
Für die LEN Funktion können nur Zeichenveränderliche verwendet werden.
Das Format der MID Funktion ist nachfolgend dargestellt.

MID (m[,n]) (m, n: numerische Ausdrücke)

Diese Funktion ruft den n Zeichenteil vom m-ten Zeichen der in der exklusiven Zeichenveränderlichen ($) gespeicherten Zeichenfolge ab.
Wird n weggelassen, dann werden alle Zeichen nach dem m-ten Zeichen abgerufen.
Da der Dezimalteil von m und n abgeschnitten wird, haben diese einen Wert von 1 bis 30. Und m muß kleiner sein, als die gespeicherte Anzahl der Zeichen. Ist die Summe von m und n größer als die gespeicherte Anzahl der Zeichen plus 1, dann tritt Fehler ein.

Beispiel: Wenn $ = "ABCDEFGHIJ"

Um vier Zeichen ab dem dritten Zeichen abzurufen → MID (3,4) . . . CDEF
Um zwei Zeichen ab dem ersten Zeichen abzurufen → MID (1,2) . . . AB
Um alle Zeichen ab dem fünften Zeichen abzurufen → MID (5) . . . EFGHIJ

Beispiel:

```
10 INPUT $
20 N=LEN($)/2
30 X$=MID(1,N):Y$=MID(N+1)
40 PRINT X$,Y$
50 END
```

Dieses Programm teilt die Eingabezeichenfolgen in zwei, indem die LEN Funktion und die MID Funktion verwendet werden.
Damit wird die Größe der exklusiven Zeichenveränderlichen $ in Zeile 20 bestimmt. Diese Hälften werden N zugeordnet, wobei die erste Hälfte und die zweite Hälfte mit Hilfe der MID Funktion geteilt werden.
Wenn daher die Größe der Eingabezeichenfolge nicht bestimmt ist, können die Hälften oder Drittel nicht abgerufen werden, so daß die Größe mit Hilfe der LEN Funktion bestimmt werden muß.
In diesem Falle ist daher die in $ einzugebende Zeichenfolge auf 30 Zeichen begrenzt. Da aber die Anzahl der Zeichen in X$ und Y$ auf 7 Zeichen begrenzt ist, wird die in einzugebenden Anzahl an Zeichen auf 14 Zeichen begrenzt.

- **VAL Funktion**

Die VAL Funktion dient für die Umwandlung der Zahlen einer Zeichenveränderlichen in einen numerischen Wert.

Format: VAL (Zeichenveränderliche)

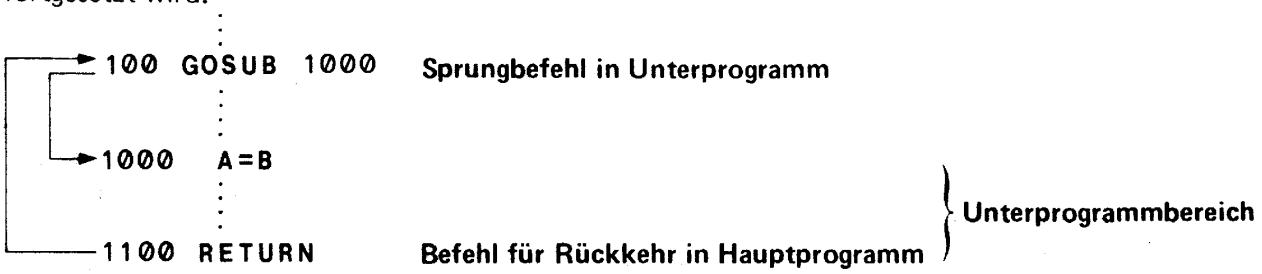
Da diese Funktion, die in einer Zeichenveränderlichen gespeicherten Zahlen in einen numerischen Wert umwandelt, kommt es zu einem Fehler, wenn keine Zahlen in der Zeichenveränderlichen gespeichert sind (zum Beispiel "ABC").

Beispiel: Wenn Z$ = "78963" ist, ergibt sich VAL (Z$) = 78963

Hinweis: Wird diese Funktion in einem Programm verwendet und kommt es zu einem Fehler (wenn die angewählte Zeichenveränderliche keine Zahlen enthält), dann erscheint das Fehlersymbol "ERR 2", wobei Programmbereich und Zeilen-Nummer nicht angezeigt werden.

4-5-5 Unterprogramme

Wenn z.B. ein bestimmter Programmteil mehr als einmal im Hauptprogramm vorkommt, ist es zweckdienlich, diesen in einem Unterprogramm zu codieren und dann beliebig oft in das Hauptprogramm abzurufen.
Mit Hilfe des Befehls (GOSUB) kann aus dem Hauptprogramm das Unterprogramm abgerufen werden; nachdem das Unterprogramm durchgeführt wurde, wird wieder an die Stelle des Hauptprogramms zurückgekehrt, an der das Unterprogramm abgerufen wurde (GOSUB Befehl), wonach das Hauptprogramm fortgesetzt wird.



Die für den Ablauf eines Unterprogrammes erforderlichen Befehle sind die "GOSUB" und "RETURN" Anweisungen. Der Sprungbefehl (Abruf) zum Unterprogramm ist die GOSUB Anweisung, wogegen mit der RETURN Anweisung wieder in das Hauptprogramm zurückgekehrt wird.
Die GOSUB Anweisung wird in den folgenden Formaten verwendet.

(1) **GOSUB Numerische Ausdruck** (der numerische Ausdruck wird am Dezimalpunkt abgeschnitten und verwendet die Zeilen-Nummern 1 bis 9999)

(2) **GOSUB # Numerischer Ausdruck** (der numerische Ausdruck wird am Dezimalpunkt abgeschnitten und verwendet $0 \leq n < 10$)

Im ersten Verfahren wird die Zeilen-Nummer direkt nach der "GOSUB" Anweisung geschrieben, wobei das Unterprogramm mit Hilfe des in der Veränderliche enthaltenen Wertes indirekt adressiert werden kann. Im zweiten Verfahren wird ein Unterprogramm eines anderen Programmbereiches (P0 bis P9) für die direkte oder indirekte Adressierung verwendet. In beiden Fällen wird nach Beendigung des Unterprogrammes jedoch nur dann wieder in das Hauptprogramm zurückgekehrt, wenn die RETURN Anweisung am Ende des Unterprogrammes geschrieben ist.

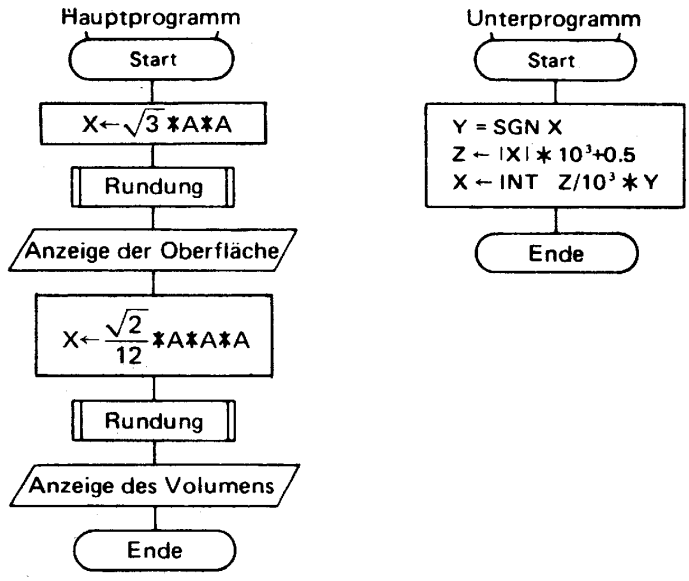
■ Grundlagen für Unterprogramme

In diesem Abschnitt möchten wir die Verwendung von Unterprogrammen anhand tatsächlicher Programmbeispiele erläutern.

Beispiel 1: Erstelle ein Programm für die Berechnung der Oberfläche und des Volumens eines regelmäßigen Tetraeders mit der Seitenlänge A mit einer Genauigkeit von drei Dezimalstellen. Dabei ist das Programm für die Rundung in ein Unterprogramm einzuarbeiten.

$$\left[\begin{array}{l} S = \sqrt{3}a^2 \\ V = \frac{\sqrt{2}}{12}a^3 \end{array}\right]$$

```
  10 INPUT A
  20 X=SQR 3*A*A
  30 GOSUB 1000
  40 PRINT X
  50 X=SQR 2/12*A*A*A
  60 GOSUB 1000
  70 PRINT X
  80 END
1000 Y=SGN X
1010 Z=ABS X*10↑3+0.5
1020 X=INT Z/10↑3*Y
1030 RETURN
```



Das Hauptprogramm für die Berechnung der Oberfläche und des Volumens ist einfach zu verstehen, da dieses ähnlich zu den bereits früher erstellten Programmen ist.
Der Unterschied besteht lediglich in dem Rundungsprogramm, das ab Zeile 1000 als Unterprogramm im Hauptprogramm enthalten ist. Dieses Unterprogramm hat die Aufgabe, die Oberfläche und das Volumen (wie in den Zeilen 20 und 50 bestimmt) zu berechnen, und zwar durch Rundung mit drei Dezimalstellen.

Dieses Unterprogramm kann beliebig oft verwendet werden.
In dem vorliegenden Programm wird dieses Unterprogramm zweimal verwendet; das erste Mal für das Runden des berechneten Oberflächenwertes und das zweite Mal für die Rundung des erhaltenen Volumens. Wird kein Unterprogramm verwendet, d.h. das Hauptprogramm voll ausgeschrieben, dann ist eine größere Anzahl an Programmschritten erforderlich.

```
  10 INPUT A
  20 X=SQR 3*A*A
  30 Y=SGN X
  40 Z=ABS X*10↑3+0.5
  50 X=INT Z/10↑3*Y
  60 PRINT X
  70 X=SQR 2/12*A*A*A
  80 Y=SGN X
  90 Z=ABS X*10↑3+0.5
 100 X=INT Z/10↑3*Y
 110 PRINT X
 120 END
```

Das zuletzt beschriebene Programm und das früher beschriebene Programm mit Unterprogramm führen die gleichen Vorgänge aus, wobei jedoch das mit Unterprogramm versehene Programm kürzer und einfacher ist. Mit anderen Worten, ein Unterprogramm wird verwendet, um ein Hauptprogramm zu vereinfachen und Programmschritte zu sparen.

* Da bei diesen Beispiel das Rundungsprogramm in einem Unterprogramm enthalten ist, "PRINT X" nicht im Unterprogramm enthalten. In diesem Programm sind die Zeilen 30 bis 60 und die Zeilen 80 bis 110 gleich, so daß diese in einem Unterprogramm zusammengefaßt werden.

Dieses Unterprogramm ist auch Bestandteil des Hauptprogramms, so daß es nicht im Hauptprogramm enthalten ist, sondern am Ende ("END") geschrieben und mit der RETURN Anweisung versehen wird. Wird es nämlich in das Hauptprogramm eingeschrieben, dann wird es auch als Hauptprogramm gelesen und wiederholt ausgeführt, so daß es bei Ausführung der RETURN Anweisung zu einem Fehler kommen würde. In diesem Beispiel muß das Unterprogramm daher getrennt geschrieben werden, indem an Zeile 1000 oder

5000 begonnen wird.
Ein Unterprogramm ist ähnlich zu einem GOTO Befehl (GOTO Anweisung) und wird als Sprungbefehl angesehen; der Unterschied besteht darin, daß das Unterprogramm immer mit einer RETURN Anweisung endet, so daß nach Ablauf des Unterprogramms zur GOSUB Anweisung des Hauptprogramms zurückgekehrt wird.

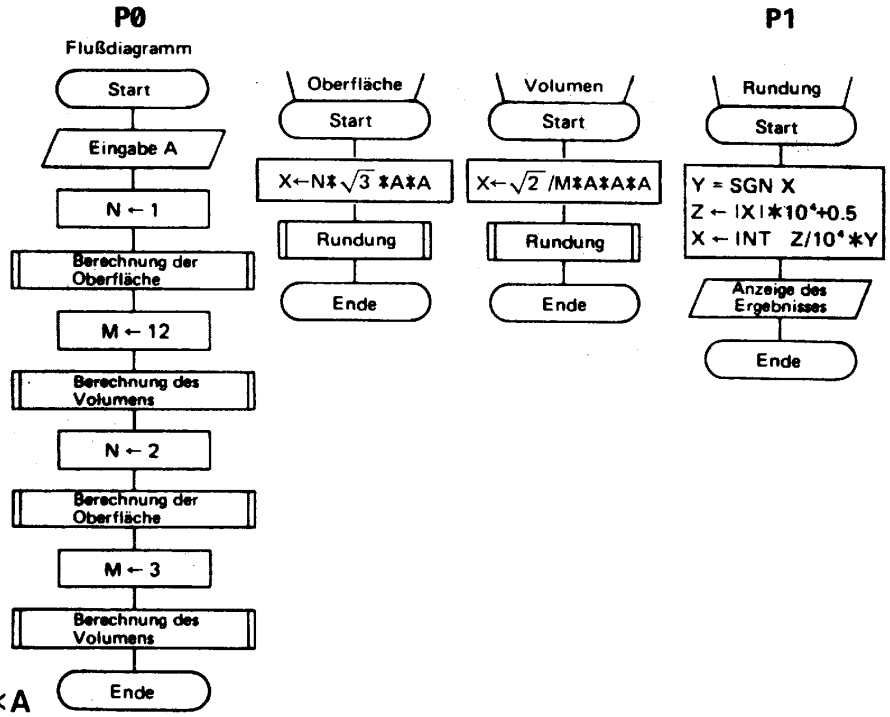
Beispiel 2: Erstelle ein Programm für die Berechnung der Oberfläche und des Volumens eines regelmäßigen Tetraeders und eines regelmäßigen Oktaeders mit der Seitenlänge A, und zwar mit einer Genauigkeit von vier Dezimalstellen. Dabei sind das Haupt- und das Unterprogramm für die Berechnung der Oberfläche und des Volumens in P0 und das Rundungs-Unterprogramm in P1 einzugeben.

Regelmäßiger Tetraeder

$S = \sqrt{3}a^2$, $V = \frac{\sqrt{2}}{12}a^3$

Regelmäßiger Oktaeder

$S = 2\sqrt{3}a^2$, $V = \frac{\sqrt{2}}{3}a^3$




```
P0  10  INPUT A
    20  N=1
    30  GOSUB 1000
    40  M=12
    50  GOSUB 2000
    60  N=2
    70  GOSUB 1000
    80  M=3
    90  GOSUB 2000
   100  END
  1000  X=N*SQR 3*A*A
  1010  GOSUB #1
  1020  RETURN
  2000  X=SQR 2/M*A*A*A
  2010  GOSUB #1
  2020  RETURN

P1  10  Y=SGN X
    20  Z=ABS X*10↑4+0.5
    30  X=INT Z/10↑4*Y
    40  PRINT X
    50  RETURN
```

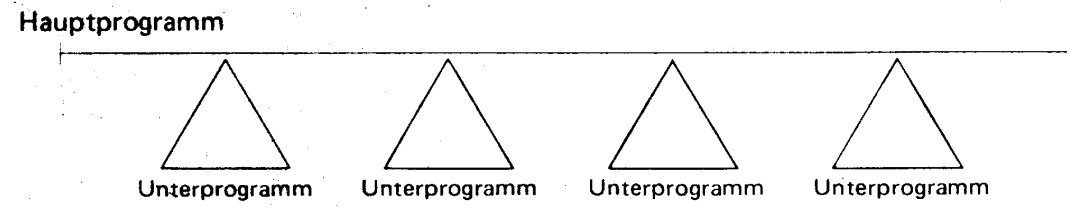
In diesem Programm werden drei Unterprogramme verwendet; zwei dieser Unterprogramme stellen den gleichen Programmteil für die Berechnung der Oberfläche und des Volumens dar, wogegen das dritte Unterprogramm das Rundungsprogramm ist.

Nach diesem Verfahren können gemeinsame Rechnungsvorgänge in einem Unterprogramm zusammengefaßt werden; danach ist wiederholte Ausführung durch Änderung der Werte für N und M möglich. Die gemeinsamen Programmteile können also in Unterprogrammen zusammengefaßt werden, was zu einer Einsparung an Programmschritten und zu einem wesentlich übersichtlicheren Programm führt.

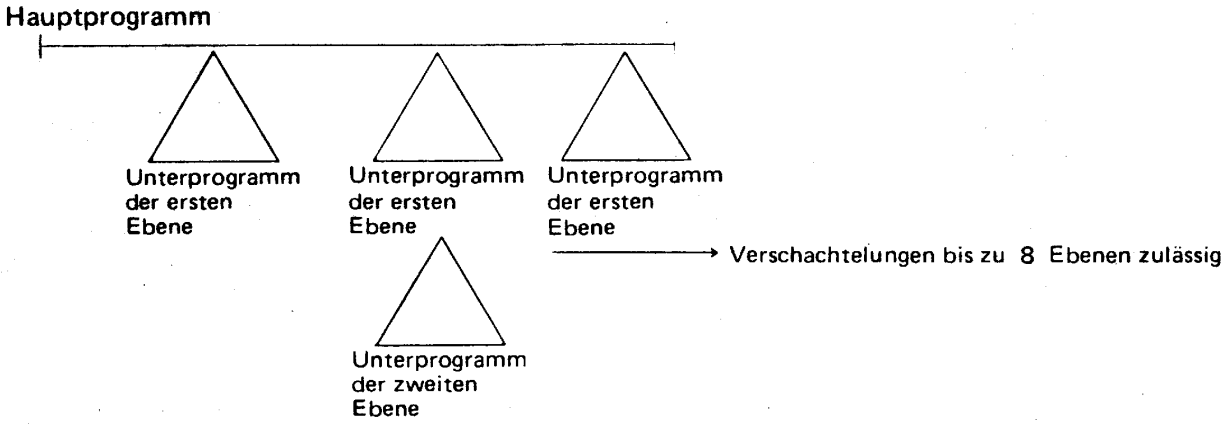
In den Zeilen 1010 und 2010 dieses Programms wird ein Unterprogramm von einem Unterprogramm aufgerufen. Dies wird als Verschachtelung bezeichnet. Ein Unterprogramm kann also ähnlich wie von einem Hauptprogramm auch von einem anderen Unterprogramm abgerufen werden.

Eine Verschachtelung von Unterprogrammen ist auf bis zu 8 Ebenen (8 Mal) möglich; darüberhinausgehende Verschachtelungen führen zu Fehler. Vom Hauptprogramm wird also zuerst das Unterprogramm der ersten Ebene aufgerufen, worauf dieses Unterprogramm die Aufgabe des Hauptprogramms übernimmt und das nächeste Unterprogramm (zweite Ebene) abruft. Trotzdem muß aber am Ende jedes Unterprogrammes die RETURN Anweisung geschrieben werden, und zwar für das jeweils übergeordnete Unterprogramm.
Ähnlich wie das Rundungs-Unterprogramm kann dieses Unterprogramm auch in einen anderen Programmbereich eingeschlossen werden (in diesem Fall P1). Damit können also auch Unterprogramme gemeinsam für verschiedene Hauptprogramme verwendet werden.

Auch die Verwendung von vielen Unterprogrammen in einem Hauptprogramm ist möglich.
Ähnlich wie vei Verschachtelungen können bis zu 8 Ebenen verwendet werden.



Nach diesem Verfahren können viele Unterprogramme verwendet werden.



Gemeinsame Programmteile können also in einem Unterprogramm zusammengefaßt werden. Bei komplexen Programmen können Unterprogramme auch verwendet werden, um bestimmte Programmteile in Gruppen aufzuteilen.
Der GOSUB Befehl in einem Unterprogramm ist ähnlich einer GOTO Anweisung, da er das Unterprogramm bezeichnet, zu welchem der Programmsprung erfolgt und von dem dann wieder zur Ausgangsposition zurückgekehrt wird.

Wenn das Unterprogramm in das gleiche Hauptprogramm eingeschrieben wird, dann muß die END Anweisung am Ende des Hauptprogrammes geschrieben werden; bei einer Wiederholung ist die GOTO Anweisung erforderlich.

■ Indirekte Adressierung eines Unterprogrammes

Ähnlich wie bereits für die GOTO Anweisung beschrieben, ist auch indirekte Adressierung eines Unterprogramms möglich.
Das Verfahren ist ähnlich wie das für die GOTO Anweisung. Es erfolgt ein Programmsprung zum Unterprogramm, nach dessen Ausführung wieder in die Ausgangsposition zurückgekehrt wird.

Beispiel 1:

```
 10 INPUT N
 20 GOSUB N+100
 30 PRINT X*6
 40 END
101 X=5:RETURN
102 X=10:RETURN
103 X=15:RETURN
104 X=20:RETURN
105 X=25:RETURN
106 X=30:RETURN
107 X=35:RETURN
108 X=40:RETURN
109 X=45:RETURN
110 X=50:RETURN
```

In diesem Programm ist das Unterprogramm indirekt mit Hilfe des Eingabewertes N adressiert. Wenn N gleich 1 bis 10 ist, wird ein Unterprogramm zwischen 101 und 110 aufgerufen, wobei der Wert der Veränderlichen X bestimmt und das Rechenergebnis von X *6 angezeigt wird.

Da die indirekte Adressierung der GSB Anweisung den Programmsprung (Unterprogramm) bestimmt, wird bei einem Wert von eins der Veränderlichen das erste Unterprogramm (Zeile 101) aufgerufen. Beträgt der Wert der Veränderlichen zwei, dann erfolgt der Sprung zum zweiten Unterprogramm (Zeile 102) usw. Die indirekte Adressierung erfolgt also in Abhängigkeit vom Wert der Veränderlichen.

Beispiel 2: Unter Verwendung der indirekten Adressierung der GSB Anweisung ist das Programm zur Ermittlung der sortierten Summen neu zu erstellen, das bereits in einem früheren Beispiel aufgeführt wurde und bei dem die indirekte Adressierung mittels GOTO Anweisung verwendet wurde (5 Abteilungen).

```
 10 VAC
 20 INPUT I
 30 GOSUB I+100
 40 GOTO 20
100 PRINT A,B,C,D,E:END
101 INPUT J:A=A+J:RETURN
102 INPUT J:B=B+J:RETURN
103 INPUT J:C=C+J:RETURN
104 INPUT J:D=D+J:RETURN
105 INPUT J:E=E+J:RETURN
```

Dieses Programm ist fast gleich wie in dem früheren Beispiel einer indirekten Adressierung mittels GOTO Anweisung. Die GOTO Anweisung in Zeile 30 wurde jedoch durch eine GOSUB Anweisung ersetzt.
Nach diesem Verfahren werden auch die Code-Nummern 1 bis 5 in Zeile 20 eingegeben und die Zeile 30 indirekt adressiert sowie die Unterprogramme unter Verwendung der Zeilen 101 und 105 unterteilt. Jede Abteilung wird in einem Unterprogramm summiert. Auch für dieses Programm wird jedes Unterprogramm in einer Zeile als Mehrfachanweisung geschrieben. Jede Zeile wird also zu einem Unterprogramm, was das Lesen wesentlich vereinfacht.
Auch in diesem Fall wird in Zeile 20 die Code-Nummer 1 bis 5 eingegeben. Falls eine andere Nummer verwendet wird, dann tritt Fehler ein, da die Adresse des Sprungbefehls nicht spezifiziert ist.

4-5-6 Allgemeine Funktionen

Zu den allgemeinen Funktionen zählen die trigonometrischen Funktion (Sinus, Cosinus und Tangens) sowie die eingebauten Funktionen. Diese Funktionen können mit Hilfe der Buchstabentasten der Tastatur oder aber auch mittels Eintastenbefehl unter Verwendung nur einer Taste (gleich wie bei einem herkömmlichen elektronischen Taschenrechner) eingegeben werden. Die allgemeinen Funktionen können sowohl für manuelles Rechnen als auch für Programmrechnungen verwendet werden.

Funktionsbezeichnung		Format	Beispiel
Trigonometrische Funktionen	$\sin x$ $\cos x$ $\tan x$	SIN x COS x TAN x	SIN 30 SIN A SIN (N+5) COS 3.14 COS I COS (L−3) TAN 70 TAN F TAN (F X 2)
Inverse trigonometrische Funktionen	$\sin^{-1} x$ $\cos^{-1} x$ $\tan^{-1} x$	ASN x ACS x ATN x	ASN 0.07 ASN P ASN (Z+Y) ACS π/5 ACS D ACS (C−X) ATN 6.5 ATN V ATN (Q+0.5)
Quadratwurzel	$\sqrt{x}$	SQR x	SQR 69 SQR A SQR (R X S)
Exponentialfunktion (Mit dieser Funktion wird die Anweisung eingegeben, den entsprechenden Zahlenwert aus der Exponentialtabelle abzurufen.)	e	EXP 1	EXP 1
Natürlicher Logarithmus	$\log_e x$	LN x	LN 43 LN P LN (U+T)
Briggsscher Logarithmus	$\log_{10} x$	LOG x	LOG 24.6 LOG R LOG (G+15)
Integration (maximale ganze Zahl nicht über x)	INT x	INT x	INT 347.457 INT V INT (Q+U) INT −45.43
Bruchteil (x mit ganzzahligem Teil entfernt)	FRAC x	FRAC x	FRAC 73.54 FRAC N FRAC (H+B)
Absolutwert	$\lvert x \rvert$	ABS x	ABS −9.43 ABS L ABS (K/P)
Vorzeichen (Wenn $x > 0$, 1) (Wenn $x = 0$, 0) (Wenn $x < 0$, −1)		SGN x	SGN 79 SGN E SGN (P−0)
Bezeichnung der wichtigsten Stellen (x wird bis hinunter zur 10^y-wichtigsten Stelle durch Rundung bestimmt)		RND (x, y)	RND (123.456, 2) RND (A, C) RND (F+E, G−5)
Erzeugung einer Zufallszahl (Gleichmäßige Erzeugung einer Zufallszahl im Bereich von $0 < x < 1$)		RAN #	RAN #
Winkelargument	Altgrad Bogenmaß Neugrad	MODE 4 MODE 5 MODE 6	Rechter Winkel = 90 Altgrad Rechter Winkel = $\frac{\pi}{2}$ Bogenmaß Rechter Winkel = 100 Neugrad

* x und y sind Konstante, Veränderliche oder numerische Ausdrücke.

Da diese Funktionen gespeichert sind, können sie auf Tastendruck auch in Programmen verwendet werden.

> EXP (e) kann nicht in einer kontinuierliche Berechnung bzw. als Mehrfachanweisung verwendet werden.

Beispiel 1: Erstelle ein Programm zur Berechnung der Seitenlänge eines Dreiecks, wenn die Länge der beiden anderen Seiten und der von diesen beiden Seiten eingeschlossene Winkel bekannt sind.

[$C = \sqrt{a^2 + b^2 - 2ab\cos\theta}$]

```
10 MODE 4
20 INPUT A,B,C
30 D=SQR (A*A+B*B-2*A*B*COS DEG C)
40 PRINT D
50 GOTO 20
```

Da bei trigonometrischen Funktionen das Winkelargument "Altgrad" verwendet wird, erscheint MODE 4 in der Zeile-Nummer 10 in diesem Programm. Danach werden die Längen der beiden Seiten und der von diesen Seiten eingeschlossene Winkel eingegeben.
Gemäß der mathematischen Formel werden die Rechenoperationen für Quadratwurzel und Cosinus mit SQR bzw. COS in den numerischen Ausdruck eingeschrieben.

Beispiel 2: Erstelle ein Programm zur Berechnung des Gewinns einen Vestärkers in Dezibel (dB), wenn die Eingangsspannung X und die Ausgangsspannung Y bekannt sind.

[$dB = 20 \cdot \log_{10} \frac{Y}{X}$]

```
10 INPUT X,Y
20 Z=20*LOG (Y/X)
30 PRINT Z
40 GOTO 10
```

Wird die Eingangsspannung mit X und die Ausgangsspannung mit Y bezeichnet, dann kann die Formel gemäß Zeile 20 geschrieben werden, um den Gewinn (Z) zu erhalten.

Beispiel 3: Erstelle ein Programm zur Erzeugung einer dreistelligen Zufallszahl unter Verwendung der Zufallszahlfunktion und einer Funktion der Bezeichnung der wichtigsten Stellen.

```
10 N=RND(RAN#,-4)*1000
20 PRINT N
30 END
```

Da die Zufallszahl im Bereich von $0 < x < 1$ mit 10-stelliger Mantisse erzeugt werden kann, müssen die drei ersten Stellen als wichtigste Stellen bezeichnet und mit 1000 multipliziert werden, um eine dreistellige Zufallszahl zu erhalten.

Beispiel 4: Erstelle ein Programm zur Ausführung einer Rechnung mit Exponentialfunktion.

$$\left[\text{Berechne } \frac{A + e^{1.5}}{B}\right]$$

```
10 E=EXP 1
20 INPUT A,B
30 C=(A+E↑1.5)/B
40 PRINT C
50 GOTO 20
```

Die Grundzahl (e) des natürlichen Logarithmus wird in die Veränderliche (E) in Zeile 10 eingegeben, worauf e^x unter Verwendung dieses Wertes (Veränderliche E) und der Exponentialfunktion in Zeile 30 berechnet wird.

4-5-7 Sonderzubehör

■ Cassetten-Recorder

Von diesem Computer aus können über die als Sonderzubehör erhältliche Schnittstelle FA-3 Daten und Programme auf das Cassetten-Tonband eines herkömmlichen Cassetten-Recorders überspielt werden. Ist der Cassetten-Recorder mit Fernbedienungsbuchse ausgestattet, dann ist Fernsteuerung des Cassetten-Recorders vom Computer FX-700P aus über die Schnittstelle FA-3 möglich.
Die Anschlußvorgänge für den Cassetten-Recorder und die Bedienungsvorgänge sind der Bedienungsanleitung des Modelles FA-3 zu entnehmen.

- **Programmaufnahme**

Format: SAVE [Dateibezeichnung"] (Klammerausdrücke können weggelassen werden.)

Die Dateilbezeichnung kann aus einer Zeichenkette (bis zu acht Zeichen) bestehen, die in Anführungszeichen zu setzen ist (die Zeichenkette kann aus alphabetischen Buchstaben, Zahlen und Symbolen gebildet werden).

Beispiel:

"ABC"

"NO. 1"

Mit dieser Anweisung wird der Cassetten-Recorder auf Aufnahme (RECORD) geschaltet.

Bedienung:

SAVE ["Dateibezeichnung"] EXE

Die SAVE Anweisung kann nur manuell eingegeben werden.

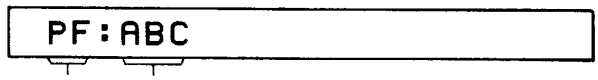
- **Einlesen eines Programms**

Format: LOAD ["Dateibezeichnung"] (Klammerausdrücke können weggelassen werden.)

Mit dieser Anweisung wird der Cassetten-Recorder auf Wiedergabe (PLAYBACK) geschaltet.

Bedienung: LOAD ["Dateibezeichnung"] EXE

Anzeige während des Einlesens



Programmdatei Dateibezeichnung

Auch wenn in dem angewählten Programmbereich bereits ein Programm geschrieben ist, wird durch das Einlesen das alte Programm gelöscht, so daß nur das neue Programm erhalten wird.

- **Aufnahme aller Programme**

Format: SAVE A ["Dateibezeichnung"] (Klammerausdrücke können weggelassen werden.)

Mit dieser Anweisung werden alle in den Programmbereichen P0 bis P9 gespeicherten Programme auf Band aufgenommen.
Die Bedienung erfolgt gleich wie mittels SAVE Anweisung, wobei der Cassetten-Recorder auf Aufnahme (RECORD) geschaltet wird.

Bedienung:

SAVE A ["Dateibezeichnung"] EXE

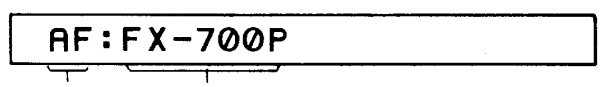
- **Einlesen aller Programme**

Format: LOAD A ["Dateibezeichnung"] (Klammerausdrücke können weggelassen werden.)

Mit dieser Anweisung werden alle Programme von Band in den Programmbereich überspielt, die vorher mittels SAVE A Anweisung auf Band aufgezeichnet wurden.
Die Bedienung erfolgt gleich wie für die LOAD Anweisung, d.h. der Cassetten-Recorder wird auf die Wiedergabe (PLAYBACK) geschaltet.

Bedienung: LOAD A ["Dateibezeichnung"] EXE

Anzeige während des Einlesens



Alle Programme Dateibezeichnung

Auch wenn bereits Programme in die Programmbereiche eingeschrieben sind, werden durch das Einlesen aller Programme die alten Programme gelöscht, so daß nur die neuen Programme erhalten bleiben. Sowohl die SAVE A Anweisung als auch die LOAD A Anweisung können nur manuell eingegeben werden.

- **Datenaufnahme**

Format: PUT ["Dateibezeichnung"] Veränderliche 1 [, Veränderliche 2]
(Klammerausdrücke können weggelassen werden.)

Durch die obige Anweisung werden die in den Veränderlichen 1 bis 2 gespeicherten Daten auf Band aufgezeichnet.

Beispiel:
PUT "PB" A ..... Daten der Veränderlichen A
PUT "1-2" A, Z ..... Daten der Veränderlichen A bis Z
PUT "DT" $, A, Z(10) ..... Daten der Zeichenveränderlichen $ und der Veränderlichen A bis Z(10)

Für die Aufnahme der in der exklusiven Zeichenveränderlichen $ gespeicherten Daten, ist $ zuerst einzuschreiben.
Diese Anweisung kann sowohl manuell eingetastet als auch in ein Programm geschrieben werden. Für manuelle Eingabe ist der Cassetten-Recorder auf Aufnahme (RECORD) zu schalten.

Bedienung:

PUT ["Dateibezeichnung"] Veränderliche 1 [, Veränderliche 2] EXE

Ist die PUT Anweisung in ein Programm geschrieben, die PUT Anweisung und die Adresse eingeben und das Programm ausführen.

- **Lesen der Daten**

Format: GET ["Dateibezeichnung"] Veränderliche 1 [, Veränderliche 2] EXE
(Klammerausdrücke können weggelassen werden.)

Diese Anweisung kann sowohl manuell eingetastet als auch in ein Programm geschrieben werden. Für manuelle Eingabe ist der Cassetten-Recorder auf die Wiedergabe (PLAYBACK) zu schalten, worauf wie folgt vorgegangen werden muß.

Bedienung:

GET ["Dateibezeichnung"] Veränderliche 1 [, Veränderliche 2]

Für Verwendung in einem Programm, die GET Anweisung und die Adresse eingeben und mit dem Programm beginnen.

- **Kontrolle der auf Band aufgezeichneten Datei**

Mittels VER Anweisung kann kontrolliert werden, ob die Programme und Daten richtig auf Band aufgezeichnet wurden.

Format: VER ["Dateibezeichnung"] (Klammerausdrücke können weggelassen werden.)

Die Bedienungsreihenfolge ist ähnlich wie für das Einlesen der Programme.

■ Drucker

An den PB-100 kann ein exklusiver Mini-Drucker angeschlossen werden, worauf Programme und Daten ausgedruckt werden können. Es ist auch möglich, während der Ausführung von Programmen die Rechenergebnisse auszudrucken.
Anschluß- und Bedienungsvorgänge sind der Bedienungsanleitung des Mini-Druckers zu entnehmen.
Um ein Programm auszudrucken, die Tastenfolge [MODE][7] eingeben, um auf die Betriebsart PRINT zu schalten.

Auflisten eines Programms

[MODE][0][MODE][7]

LIST [EXE] oder **LIST A** [EXE]

[MODE][8] (Freigabe der Betriebsart PRINT)

Nach dem Druckvorgang ist die Tastenfolge [MODE][8] einzugeben, um die Betriebsart PRINT freizugeben. Um die Rechenergebnisse und die Rechenvorgänge automatisch auszudrucken, können die Anweisungen "MODE 7" und "MODE 8" auch in ein Programm eingeschrieben werden.

Beispiel:

```
)
100 MODE 7
110 PRINT A
120 MODE 8
```

Wenn "MODE 7" in ein Programm geschrieben ist, dann muß vor dem Programmende auch die Anweisung "MODE 8" geschrieben sein, um die Betriebsart PRT wieder freizugeben.

Fehleranzeigeliste

Fehler-Code	Bedeutung	Ursache	Abhilfe
1	Speicher-Überlauf oder Systemstapel-Überlauf	● Anzahl der Programmschritte nicht ausreichend. Programm kann nicht eingelesen werden. ● Stapel-Überlauf	● Nicht benötigte Programme löschen oder die Anzahl der Speicher verringern. ● Die Formeln auftrennen und einfacher gestalten.
2	Syntax-Fehler	● Formatfehler im Programm usw. ● Linkes und rechtes Format in einer Anweisung unterschiedlich usw.	● Fehler im Eingabeprogramm usw. berichtigen.
3	Mathematischer Fehler	● Das Ergebnis einer Berechnung eines numerischen Ausdruckes übersteigt 10^{100} ● Eingabe eines Wertes, der außerhalb des zulässigen Eingabebereiches einer Funktion liegt. ● Ergebnis unendlich oder unmöglich.	● Die Berechnungsformel oder die Daten berichtigen ● Die Daten kontrollieren
4	Fehler aufgrund einer nicht eingegebenen Zeilenzahl	● Keine Zeilen-Nummer Adresse für die GOTO oder GSB Anweisung	● Die richtige Zeilen-Nummer eingeben
5	Argument-Fehler	● Bei Funktionen, die ein Argument erfordern, liegt das eingegebene Argument außerhalb des Eingabebereiches	● Argument-Fehler berichtigen
6	Veränderlichen-Fehler	● Es wurde versucht, einen Speicher zu verwenden, der nicht vorhanden ist (nicht durch Erweiterungsfunktion bezeichnet). ● Es wurde versucht, den gleichen Speicher für eine Zahlenveränderliche und eine Zeichenveränderliche gleichzeitig zu verwenden.	● Die Speicherzahl wie erforderlich erweitern. ● Der gleiche Speicher kann nicht gleichzeitig für eine Zahlenveränderliche und eine Zeichenveränderliche verwendet werden.
7	Verschachtelungs-Fehler	● RET Anweisung wird ausgegeben, wenn das Unterprogramm nicht ausgeführt wird. ● NEXT Anweisung wird ausgegeben, wenn nicht in einer FOR Schleife gerechnet wird. ● Unterprogramme auf mehr als 8 Ebenen. ● FOR·NEXT Schleifen auf mehr als 4 Ebenen.	● Nicht erforderliche RETURN Anweisung oder NEXT Anweisungen löschen. ● Die Unterprogramme oder FOR·NEXT Anweisungen innerhalb der zulässigen Ebenen behalten.
9	Sonderzubehör-Fehler	● Kein Drucker oder Cassetten-Recorder angeschlossen und ein PRT Befehl oder ein SAVE Befehl wird eingegeben.	● Den Drucker oder Cassetten-Recorder anschließen. ● PRT Befehl wieder löschen.

Liste der Programmanweisungen

Klassifikation	Bezeichnung	Format	Funktion
Eingabeanweisung	INPUT	INPUT Variable Reihe	Die Daten können über die Tastatur während des Programmablaufes eingegeben werden. Der Programmablauf wird dabei unterbrochen, bis die Dateneingabe beendet ist. **S. 52**
	KEY	Zeichenvariable = KEY	Das während des Programmablaufes eingegebene Zeichen wird gelesen und einer Zeichenvariablen zugeordnet. Da das Programm durch diese Anweisung nicht unterbrochen wird, erfolgt keine Zuordnung zur Zeichenvariablen, wenn keine Tasteneingabe erfolgt. **S. 53**
Ausgabeanweisung	PRINT	PRINT Ausgangsregelfunktion {;, } Ausgangselement [{;, } ...]	Gibt ein bestimmtes Ausgangselement in einem bestimmten Format aus. **S. 53**
	CSR	CSR n {;, } $(0 \leqq n \leqq 11)$	Die Anzeige erfolgt ab der bezeichneten n-ten Stelle. **S. 54**
Verzweigung	GOTO	GOTO {Zeilennummer, Veränderliche}	Bewirkt einen Programmsprung zu einer bestimmten Zeilennummer. **S. 36**
	IF ··· {THEN, ;} ···	IF Vergleich {THEN Zeilennummer ; Anweisung}	Bewirkt einen Programmsprung zu einer nach der THEN Anweisung eingegebenen Zeilennummer oder führt die nach ";" eingebene Anweisung aus, wenn das Ergebnis des Vergleichs stimmt. Bewirkt eine Fortsetzung mit der nächsten Zeilennummer, wenn das Ergebnis des Vergleichs nicht stimmt. **S. 41**
	GOSUB	GOSUB {Zeilennummer, Veränderliche}	Ruft das Unterprogramm mit der bestimmten Zeilennummer auf, um dieses durchzuführen. Nach Ablauf des Unterprogramms, wird mittels RETURN Anweisung an die GOSUB Anweisung zurückgekehrt, um den dieser Anweisung nachfolgenden Befehl durchzuführen. **S. 57**
	RETURN	RETURN	Bezeichnet das Ende eines Unterprogramms und kehrt auf die Zeilennummer zurück, die der GOSUB Anweisung folgt. **S. 57**

Klassifikation	Bezeichnung	Format	Funktion
Schleifenbildung	FOR	FOR v=e_1 TO e_2 [STEP e_3] * Mit v wird eine numerische Variable, mit e_1, e_2 und e_3 werden dagegen numerische Ausdrücke bezeichnet.	Bestimmt den Beginn einer Programmschleife, in der der numerische Wert v von einem Anfangswert e_1 in Schritten von e_3 auf einen Endwert von e_2 ändert. Die Programmschleife wird " $\left[\frac{e_2-e_1}{e_3}\right]$ + 1 " Mal wiederholt, und zwar zwischen den Anweisungen FOR und NEXT. Falls das Inkrement e_3 weggelassen wird, dann wird e_3 mit " 1 " angenommen. **S. 46**
	NEXT	NEXT v	Bezeichnet das Ende einer FOR Schleife. Ist das Ergebnis aus v und e_3 gleich oder kleiner als e_2, dann wird die Schleife wiederholt; ist dieses größer als e_2, dann wird mit der der NEXT Anweisung folgenden Zeile fortgesetzt. **S. 46**
Ausführungsstopp	STOP	STOP	Der Programmablauf wird vorübergehend angehalten, um den Rechner auf die Tasteneingabe-Wartefunktion zu schalten. Durch Drücken der EXE Taste kann mit dem Programmablauf fortgesetzt werden. **S. 34**
Ausführungsende	END	END	Bezeichnet das Ende eines Programms; der Rechner kehrt auf die vor dem Programmablauf eingestellte Funktion zurück. Der beendete Programmablauf kann durch Drücken der EXE Taste nicht wiederholt werden. **S. 26**
Datenlöschung	VAC	VAC	Alle veränderlichen Daten eines Programms werden gelöscht. **S. 39**
Programmliste	LIST	LIST [Zeilennummer]	Zeigt alle Anweisungen eines Programms ab einer bestimmten Stelle an. **S. 28**
Programm/ Datenliste	LIST A	LIST A	Alle Anweisungen eines Programms und die in den Speichern gespeicherten Daten werden angezeigt. **S. 66**
Programmablauf	RUN	RUN [Zeilennummer]	Das Programm läuft ab einer bestimmten Zeilennummer ab. **S. 27**
Programm-löschung	CLEAR	CLEAR	Löscht ein bestimmtes Programm. **S. 27**
	CLEAR A	CLEAR A	Löscht alle Programme. **S. 27**

Klassifikation	Bezeichnung	Format	Funktion
Argument	MODE	MODE $\left\{\begin{matrix}4\\6\\6\end{matrix}\right\}$	Dient für die Bezeichnung des Arguments in Altgraden (4), im Bogenmaß (5) bzw. in Neugranden (6) für trigonometrische Funktionen. **S. 48**
Formateingabe	SET	SET $\left\{\begin{matrix}\mathrm{E}\,n\\\mathrm{F}\,n\\\mathrm{N}\end{matrix}\right\}$ $(0 \leqq n \leqq 9)$	Dient für die Eingabe der effektiven Stellenzahl und der Dezimalstellenzahl für die Anzeige der Zahlenwerte. **S. 55**
Zeichenfunktion	LEN	LEN [Zeichenveränderliche]	Bereichnet die Größe einer Zeichenveränderlichen. **S. 56**
	MID	MID (m [, n])	Ruft n-Zeichen ab dem m-ten Zeichen der in der exklusiven Zeichenveränderlichen ($) gespeicherten Zeichenkette ab. **S. 56**
	VAL	VAL [Zeichenveränderliche]	Wandelt die in einer Zeichenveränderlichen gespeicherten Zahlen in einen Zahlenwert um **S. 57**
Sonderzubehör	SAVE	SAVE ["Dateibezeichnung"]	Zeichnet das Programm des bezeichneten Programmbereiches auf Tonband auf. **S. 64**
	LOAD	LOAD ["Dateibezeichnung"]	Liest das auf Band aufgezeichnete Programm in den bezeichneten Programmbereich ein. **S. 64**
	SAVE A	SAVE A ["Dateibezeichnung"]	Zeichnet alle Programme gleichzeitig auf Tonband auf. **S. 64**
	LOAD A	LOAD A ["Dateibezeichnung"]	Liest alle auf Band aufgezeichneten Programme in die entsprechenden Programmbereiche ein. **S. 64**
	PUT	PUT ["Dateibezeichnung"] Veränderliche	Zeichnet die in der Veränderlichen gespeicherten Daten auf Band auf. **S. 65**
	GET	GET ["Dateibezeichnung"] Veränderliche	Liest die auf Band aufgenommenen Daten in die Veränderliche ein. **S. 65**
	VER	VER ["Dateibezeichnung"]	Kontrolliert, ob alle Programme und Daten richtig auf Band aufgezeichnet wurden. **S. 65**

* Die Klammerausdrücke [ ] dürfen weggelassen werden.
Jeder in { } Klammern gesetzte Ausdruck darf verwendet werden.

Technische Daten

■ **Typ**
FX-700P

■ **Grundrechnungsfunktionen**
Negative Zahlen, Exponenten, Klammerausdrücke, Additionen, Subtraktionen, Multiplikationen, Divisionen (mit Hierarchien – tatsächliche Algebralogik)

■ **Eingebaute Funktionen**
Trigonometrische Funktionen und inverse trigonometrische Funktionen (Argument in Altgrad, Bogenmaß oder Neugrad), Logarithmus- und Exponentialfunktionen, Quadratwurzeln, Potenzen, ganzahliger Teil, Bruchteil, Absolutwert, Symbole, effektive Stellenzahl, Dezimalstellenzahl, Zufallszahl, Kreiskonstante π

■ **Funktionsstellenkapazität**	**Eingabebereich**	**Genauigkeit des Ergebnisses**
$\sin x$, $\cos x$, $\tan x$	$\|x\| < 1440°$ (8π Bogenmaß, 1600 Neugrad)	±1 an der 10. Stelle
$\sin^{-1} x$, $\cos^{-1} x$	$\|x\| \leqq 1$	– " –
$\tan^{-1} x$		– " –
$\log x$, $\ln x$	$x > 0$	– " –
e^x	$x = 1$	– " –
$\sqrt{x}$	$x \geqq 0$	– " –
x^y $(x \uparrow y)$	$x > 0$	– " –

■ **Befehle:**
INPUT, PRINT, GOTO, FOR•NEXT, IF-THEN, GOSUB, RETURN, STOP, END, RUN, LIST, LIST A, MODE, SET, VAC, CLEAR, CLEAR A, DEFM, SAVE, SAVE A, LOAD, LOAD A, PUT, GET, VER

■ **Programmfunktionen**
KEY, CSR, LEN, MID, VAL

■ **Rechenbereich**
$\pm 1 \times 10^{-99}$ bis $\pm 9.999999999 \times 10^{99}$ und 0 (interne Berechnung mit 12-stelliger Mantisse)

■ **Programmsystem**
Speichersystem

■ **Programmiersprache**
BASIC

■ **Anzahl der Programmschritte**
Maximal 1.568 Schritte

■ **Anzahl der Programmspeicher**
10 (P0 bis P9)

■ **Anzahl der Speicher**
26 bis maximal 222 Speicher und exklusive Zeichenveränderliche ($)

■ **Stapelebenen**

Unterprogramme	→ 8 Ebenen
FOR•NEXT Schleifen	→ 4 Ebenen
Numerische Werte	→ 6 Ebenen
Rechenelemente	→ 12 Ebenen

■ **Anzeigesystem**
10-stellige Mantisse (einschließlich Minuszeichen) oder 8-stellige Mantisse (7-stellige Mantisse bei negativen Zahlen) und 2-stelliger Exponent und Anzeige der Betriebsartensymbole EXT, [S], [F], RUN, WRT, DEG, RAD, GRA, TR, PRT, STOP

■ **Anzeigeelement**
12-stellige Punktmatrix-Flüssigkristallanzeige

■ **Hauptkomponente**
C-MOS-VLSI und andere

■ **Stromversorgung**
Lithium-Batterie (CR2032) x 2

■ **Leistungsaufnahme**
Max. 0,02 W

■ **Batterie-Lebensdauer**
Nur Computer – etwa 300 Betriebsstunden

■ **Abschaltautomatik**
Etwa sieben Minuten nach der letzten Tastenbetätigung wird die Stromversorgung automatisch abgeschaltet.

■ **Zul. Temperaturbereich**
0°C bis 40°C

■ **Abmessungen**
9,8 (Y) x 165 (B) x 71 (T) mm

■ **Gewicht**
118 g (einschließlich Batterien)