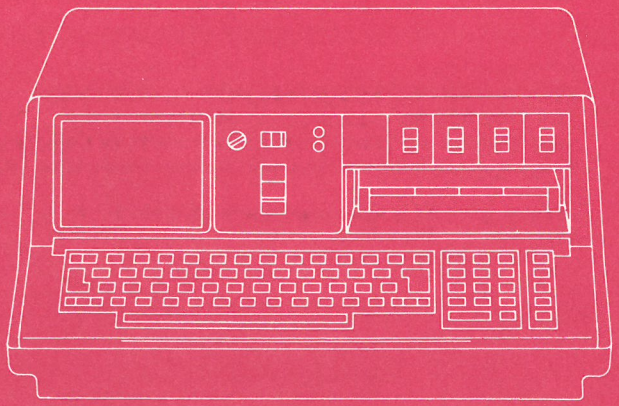
IBM

APL
Einführung
IBM 5100



Benutzerhandbuch
BASIC

Einführung
BASIC

Benutzerhandbuch
APL

Einführung
APL

Benutzerhandbuch
DFV

Benutzerhandbuch
Finanzplanung/BASIC

Benutzerhandbuch
Statistik/APL

STATIK
Demonstrations-
programme

Benutzerhandbuch
Mathematik/BASIC

Benutzerhandbuch
Mathematik/APL

Benutzerhandbuch
Statistik/BASIC

DEMO-Handbuch

Computerunterstützter
Unterricht
APL

Computerunterstützter
Unterricht
BASIC

Benutzerhandbuch
Serieller Ein-/Ausgabe-
Anschluß

Inhaltsverzeichnis

Kapitel 1 Einführung in die IBM 5100 1
Über diese Einführungsschrift 1
Über die APL-Sprache 1
Über die IBM 5100 1
Alphanumerische Tasten 2
Numerische Tasten 2
Bedienungstasten 2
APL-Tasten für Systemanweisungen 4
Tasten für die vier Grundrechenarten 4
Startvorbereitungen 4
Eingabe und Anzeige von Daten 5
Fehlerkorrektur eingetasteter Zeichen 8
Ändern eines Zeichens 8
Löschen eines Zeichens 9
Einfügen eines Zeichens 10

Kapitel 2 Einführung in die APL-Sprache . . . 14
Arten von APL-Funktionen 14
Addition, Subtraktion, Multiplikation und Division 14
Potenzen und Wurzeln 16
Radizieren 17
Speicherung von Daten zur späteren Verwendung . 17
Mehrere Funktionen in derselben Anweisung . . 18
Bestimmung der Reihenfolge der Ausführung durch Klammern 19
Verwendung von Zahlenreihen und Tabellen . . . 19
Verwendung von Zahlenreihen (Vektoren) mit APL 20
Verwendung von Tabellen (Matrizen) 22
Verwendung einzelner Elemente aus Vektoren oder Matrizen (Indizierung) 24
Nicht nur Zahlen können verarbeitet werden . . 26

Kapitel 3 APL-Funktionen mit einem Argument . 28
Wieviele Argumente sind bei den Elementarfunktionen nötig? 28
APL-Funktionen, die aus zwei Zeichen zusammengesetzt sind 28
Auf- und Abrunden 29
Kaufmännisches Runden 29
Sortieren eines Vektors in auf- oder absteigender Folge 30
Zufallszahl auswählen 31
Bilden von fortlaufenden Zahlenreihen 31
Bildung eines leeren Vektors 32
Bestimmen der Dimension von Variablen 32

Kapitel 4 APL-Funktionen mit zwei Argumenten . 33
Vergleichsfunktionen 34
Warum zwei gleiche Zahlen doch nicht gleich sind 36
Beispiel für eine Vergleichsfunktion 36
Logische Daten verknüpfen 37
Und 38
Oder 39
Die größere von zwei Zahlen bestimmen 41
Die kleinere von zwei Zahlen bestimmen 42
Index eines Wertes in einem Vektor anzeigen . . 43
Stichproben nehmen 43
Vektoren oder Matrizen komprimieren 44
Vektoren oder Matrizen expandieren 45
Zwei oder mehrere Variablen verketten . . 47
Bilden eines Ergebnisvektors mit der Funktion Verketten 49
Logarithmieren 51

Kapitel 5 Elemente eines Vektors aufsummieren (Reduzieren) 53
Plus-Reduktion (Aufsummieren) 53
Verwendung der Plus-Reduktion zur Ermittlung eines Durchschnitts 53
Verwendung der Plus-Reduktion für das Aufsummieren zweier Vektoren 56
Minus-Reduktion (alternierende Summe) . . . 56
Maximum-Reduktion: Ermitteln des größten Wertes eines Vektors 57
Minimum-Reduktion: Ermitteln des kleinsten Wertes eines Vektors 57
Oder-Reduktion: Prüfen, ob ein bestimmter Wert im Vektor vorhanden ist 58
Und-Reduktion: Prüfen, ob alle Werte zweier Vektoren übereinstimmen 58

Kapitel 6 Anwendung von Elementarfunktionen . 59
Jetzt laßt uns sehen, was Sie mit APL alles können . 59

Kapitel 7 Funktionen definieren 71
Was ist eine definierte Funktion? 71
Wie wird eine Funktion definiert? 71
Testen Ihrer Funktion, bevor sie angewandt wird . 73
Wie werden Test- und Stopeinrichtungen eingesetzt 75
Test T Δ 75
Stop S Δ 75

Was tun, wenn beim Definieren ein Fehler unterläuft 77
Anzeigen einer vollständigen Funktion . . . 78
Hinzufügen einer oder mehrerer Anweisungen am Schluß einer Funktion 79
Auswechseln einer Anweisung 80
Einfügen einer oder mehrerer Anweisungen . . 81
Löschen einer Anweisung 82
Eine bestimmte Anweisung oder die letzten Anweisungen anzeigen 83
Abändern einer einzelnen Anweisung 83
Ändern einer Kopfzeile 84
Schnell ist eine einzelne Anweisung eingefügt, ersetzt oder angefügt 85
Arten von Kopfzeilen 87
Warum gibt es Ergebnisvariablen 88
Lokale und globale Namen 89
Verzweigen, Anweisungsnamen und Schleifen . . 91
Verzweigen und Anweisungsnamen 91
Schleifenbildung 94
Wie werden Daten während der Ausführung einer Funktion eingegeben 95

Kapitel 8 Was tun, wenn eine Funktion unterbrochen wurde? 98
Wenn die Unterbrechungstaste gedrückt wurde . . 98
Wenn die Stopeinrichtung verwendet wurde . . . 100
Wenn sich ein Fehler in der Funktion befindet . . 101
Wenn das ⓤ-Zeichen anstelle einer Texteingabe eingegeben wurde 102
Anzeigen der hängenden Funktion 102
Die HOLD-Taste stoppt die Verarbeitung 103

Kapitel 9 Der Gebrauch von Bandkassetten . . . 104
Was sind Systemvariablen? 108

Anhang A. Zusammengesetzte Zeichen 112

Anhang B. 114
Stichwortverzeichnis 119

Kapitel 1 Einführung in die IBM 5100

Über diese Einführungsschrift

Diese Einführungsschrift soll Ihnen zeigen, wie die IBM 5100 unter Anwendung der APL-Sprache bedient wird. Wenn Sie im Umgang mit der APL-Sprache noch nicht vertraut sind, können Sie die beschriebenen Funktionen oder Beispiele an Ihrer IBM 5100 während des Lesens dieser Einführungsschrift ausprobieren. Sind Sie bereits firm in der Anwendung der APL-Sprache, sollten Sie die Kapitel 1 und 2 lesen, um die Bedienung der IBM 5100 zu erlernen; die Kapitel 3 bis 6 können Sie überspringen, gehen Sie dann zu Kapitel 7. Nicht alle Möglichkeiten und Funktionen der APL-Sprache werden in dieser Einführungsschrift besprochen. Tiefergehende Informationen über die IBM 5100 oder über die APL-Sprache finden Sie im IBM 5100 APL-Benutzerhandbuch, IBM Form A12-2151.

Dieses Handbuch geht von der Voraussetzung aus, daß die IBM 5100 bereits aufgestellt und durchgetestet ist.Wenn die IBM 5100 noch nicht aufgestellt ist, sollten die Installationshinweise, wie sie im IBM 5100 APL-Handbuch beschrieben sind, durchgeführt werden, bevor diese Einführungsschrift gelesen wird.

Über die APL-Sprache

APL besitzt eine Reihe von Elementarfunktionen, mit denen Sie normale Aufgaben lösen können. Wenn Sie ausgefallene Probleme lösen wollen, können Sie sich Ihre eigenen Funktionen definieren. Die von Ihnen selbst definierten Funktionen entsprechen Programmen, wie sie in anderen Computersprachen geschrieben werden.

APL ist eine Sprache, mit der es sich gut experimentieren läßt. Keine noch so falsche Eingabe über die Tastatur kann die IBM 5100 beschädigen; je mehr Sie probieren, je mehr erfahren Sie über APL.

Über die IBM 5100

Die IBM 5100 (Abb. 1) ist ein tragbarer Tischrechner, wie geschaffen, um bei der Lösung der vielfältigsten Aufgaben zu helfen. Der Bildschirm und die Anzeigelampen „führen" Sie; die Tastatur und die Schalter der IBM 5100 gestatten Ihnen die Durchführung verschiedenster Operationen.

Vor Beginn der ersten Gehversuche mit Ihrer IBM 5100 sollten Sie sich mit den Tasten und dem Steuerpult vertraut machen (Abb. 1). Die Schalter des Steuerpultes werden später erläutert.

Alphanumerische Tasten

Die alphanumerischen Tasten sind genauso wie an einer normalen Schreibmaschine angeordnet, aber es sind keine Kleinbuchstaben vorhanden. Die Großbuchstaben sind alle auf den Positionen der Kleinbuchstaben; es braucht also nicht die Umschalttaste  betätigt zu werden.

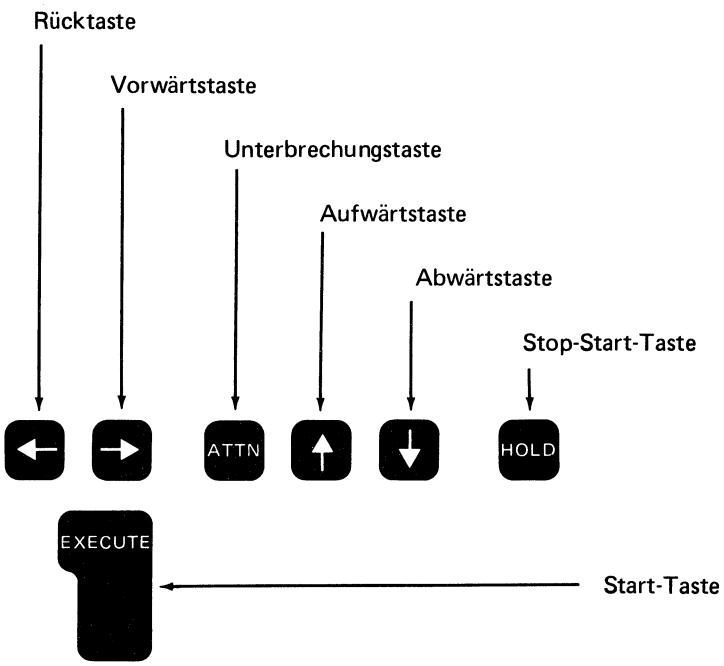
Wenn Sie ein Zeichen außerhalb des Alphabets eingeben wollen, müssen Sie die Umschalttaste gedrückt halten, während Sie das Zeichen tippen, gerade so, als wenn Sie einen Großbuchstaben auf einer gewöhnlichen Schreibmaschine schreiben.

Numerische Tasten

Wenn Sie Ziffern eingeben möchten, können Sie entweder die Tasten der Schreibmaschinentastatur (obere Reihe) oder die Zehnertastatur (Additionsmaschinentastatur) benützen.

Bedienungstasten

Die schwarzen Tasten, bezeichnet mit CMD, die grauen Tasten mit der Aufschrift EXECUTE, ATTN, HOLD und die grauen Tasten mit den Pfeilen sind besondere Bedienungstasten.



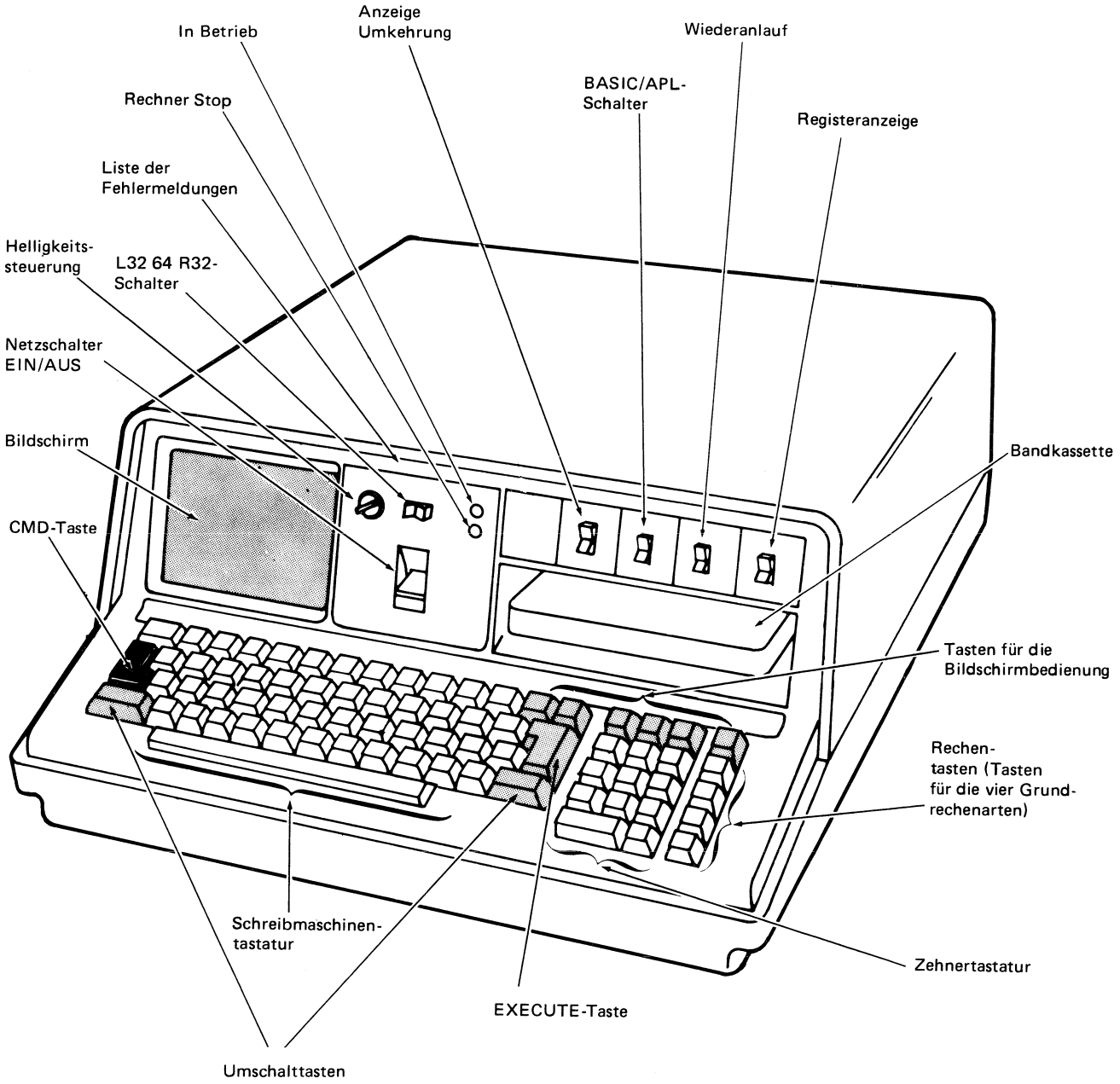


Abb. 1 Die IBM 5100

APL-Tasten für Systemanweisungen

Die Bezeichnungen über der Zahlenreihe der Schreibmaschinentastatur stellen die Systemanweisungen dar. Sie werden ausgelöst, wenn Sie die CMD-Taste (Funktionen-Umschalttaste) drücken und festhalten und gleichzeitig eine Zahlentaste drücken. Beispiel: Um)LOAD einzugeben, drücken Sie zuerst die CMD-Taste und gleichzeitig die Taste 1. Die Systemanweisungen und deren Einsatz werden später noch besprochen, und zwar im Kapitel 8.

Tasten für die vier Grundrechenarten

Die vier Tasten rechts neben der Zehnertastatur sind für die vier Grundrechenarten bestimmt. Diese Tasten werden für die Ausführung von Division, Multiplikation, Subtraktion und Addition verwendet. Innerhalb der Schreibmaschinentastatur befinden sich Tasten mit den gleichen arithmetischen Funktionen. Beachten Sie, daß die Zeichen ÷ Division und × Multiplikation bedeuten.

Startvorbereitungen

Folgende Schalter Ihrer IBM 5100 müssen so gesetzt sein:

Schalter	Stellung
L32 64 R32	64
BASIC/APL (nur bei kombinierten Maschinen)	APL
REGISTER-ANZEIGE/NORMAL	NORMAL

Wenn Ihre IBM 5100 den Schalter BASIC/APL hat, können Sie wahlweise in den Sprachen BASIC oder APL Programme ausführen.

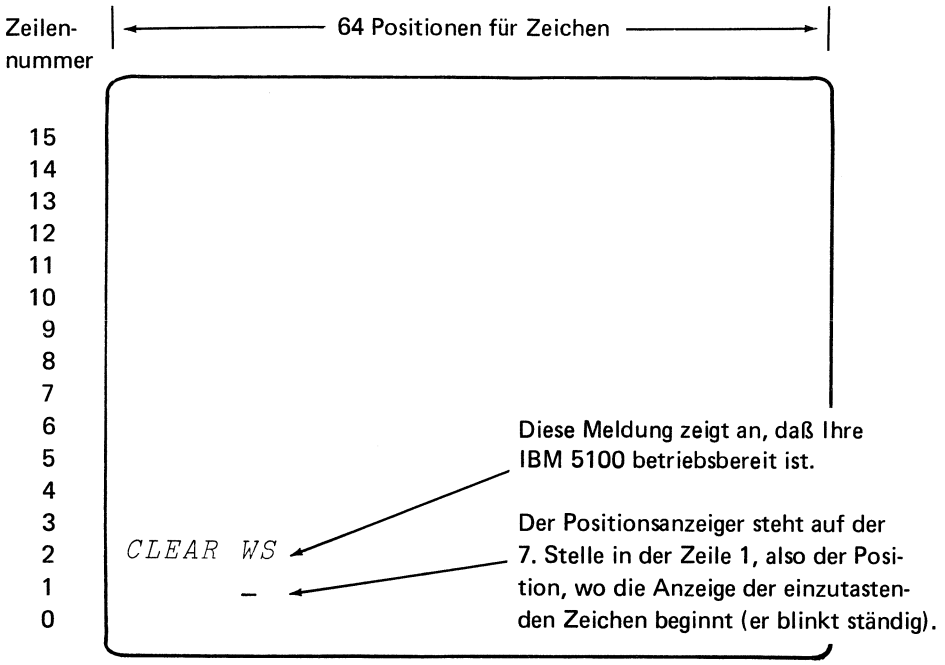
Vorausgesetzt, daß der Netzstecker in die Steckdose gesteckt wurde und Strom auf der Leitung ist, sollten Sie jetzt den Netzschalter auf EIN schalten. Nun drücken Sie den Schalter WIEDERANLAUF und warten ca. 20 Sekunden. Während dieser Zeit werden interne Prüfungen durchgeführt, die sicherstellen, daß die Maschine korrekt arbeitet. Wenn nach dieser Zeit die Meldung CLEAR WS auf dem Bildschirm in der linken unteren Ecke erscheint, ist alles in Ordnung, und Sie können mit der Arbeit beginnen. Wenn sich nach 30 Sekunden nichts tut, wiederholen Sie den Vorgang: Sie drücken WIEDERANLAUF, die IBM 5100 wiederholt dann die internen Prüfungen. Erscheint die Meldung CLEAR WS nach mehrmaligen Versuchen nicht, so rufen Sie bitte Ihren zuständigen Mitarbeiter unseres technischen Außendienstes.

Wenn Sie von APL auf BASIC übergehen wollen, so drücken Sie die Schalter BASIC und anschließend WIEDERANLAUF.

Eingabe und Anzeige von Daten

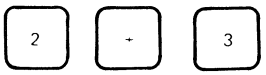
Zuerst wollen wir einen Blick auf den Bildschirm werfen. Grundsätzlich wird alles bei APL in der linken unteren Ecke beginnend angezeigt. Alles, was eingetastet wird, ist auch sofort auf dem Bildschirm sichtbar. Der kleine waagerechte, blinkende Strich zeigt genau die Position an, wo das nächste einzutastende Zeichen (Ziffer, Buchstabe, usw.) einmal zu sehen sein wird. Diesen Strich, der ständig blinkt, nennen wir Positionsanzeiger.

Auf dem Bildschirm können bis zu 16 Zeilen mit Daten gefüllt werden. Jede Zeile kann 64 Zeichen aufnehmen. Die beiden unteren Zeilen nehmen die Eingabedaten auf, die anderen 14 Zeilen zeigen immer den letzten Stand.



Es gibt maximal 128 Eingabepositionen, wenn über die Tastatur eingegeben wird, und zwar 64 Positionen in der Zeile 1 und 64 in der Zeile 0. Wurde das 64. Zeichen in der Zeile 1 eingegeben, so wandert der Positionsanzeiger zum linken Rand der Zeile 0. Er steht jetzt auf Position 65, und es können nun weitere 64 Zeichen bis zur Position 128 eingegeben werden.

Nun wollen wir einige Daten über die Zehnertastatur der IBM 5100 eintippen und zusätzlich die Arbeitsweise der Rechentasten kennenlernen. Geben Sie bitte folgendes ein:



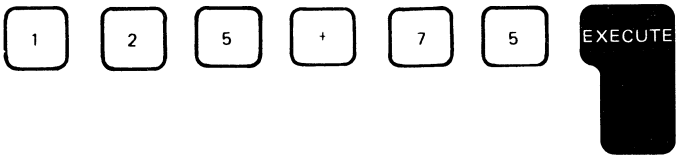
Alle Zeichen werden sofort beim Eintasten auf dem Bildschirm sichtbar. Um die Daten zu verarbeiten, die wir soeben eingetastet haben, müssen wir jetzt die Taste EXECUTE drücken.

Auf dem Bildschirm erscheint folgendes:

```
CLEAR WS
      2+3
5
      _
```

Sie haben sicher bemerkt, daß Ihre eingegebenen Zahlen 6 Stellen eingerückt stehen und das Ergebnis, nämlich die 5, in der nächsten Zeile auf der Position 1. Der Positionsanzeiger steht jetzt auf der 7. Position in der nächsten Zeile. Die Bildschirminformationen verschieben sich mit jeder Betätigung der EXECUTE-Taste um eine Zeile nach oben.

Geben Sie jetzt folgende Rechenaufgabe ein und führen Sie diese durch:



Die Bildschirmanzeige sieht so aus:

```
CLEAR WS
      2+3
5
      125+75
200
      _
```

Das Aussehen der Daten Ihres Bildschirmes kann mit Hilfe eines Schalters am Steuerpult gewechselt werden. Der Schalter ANZEIGEUMKEHRUNG hat zwei Einstellungen: Schwarze Zeichen auf weißem Grund oder weiße Zeichen auf dunklem Grund. Probieren Sie einmal aus, welche Einstellung Ihnen am besten liegt. Betätigen Sie bitte auch den Helligkeitsschalter, um die richtige Einstellung zu finden.

Versuchen Sie jetzt einmal die Wirkung des Schalters L32 64 R32; drücken Sie den Schalter nach links in die Position L32. Es erscheinen jetzt nur die linken 32 Stellen auf dem Bildschirm. Zwischen allen Zeichen werden Leerstellen eingeschoben, die 32 rechten Stellen werden nicht angezeigt. Der Bildschirm zeigt folgende Darstellung:

```
C L E A R   W S
            2 + 3
5
            1 2 5 + 7 5
2 0 0
            _
```

Jetzt bringen Sie bitte den Schalter in die R32-Position, da sich im Augenblick in diesem Bereich keine Daten befinden, ist der Bildschirm leer.

Gehen Sie jetzt wieder zurück in die Position 64 und Sie sehen, daß das ursprüngliche Bild wieder erscheint und die Leerschritte verschwunden sind. Alle Aufgaben dieser Einführungsschrift werden in der Position 64 angezeigt.

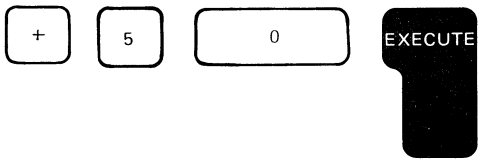
Über der obersten Reihe der Zehnertastatur befinden sich zusätzliche Bedienertasten, um die Bildschirmanzeige nach oben oder unten zu verschieben. Die Taste mit dem Pfeil nach oben ist die Aufwärtstaste; bei Betätigung dieser Taste  wird die Bildschirmanzeige um eine Zeile nach oben verschoben; durch Drücken der Abwärtstaste  (Pfeil nach unten) wird die ganze Anzeige um eine Zeile nach unten gezogen. Zeilen, die beim Verschieben nach oben oder unten herausfallen, sind verloren. Durch ständiges Drücken der Aufwärts-/Abwärtstasten wird das zeilenweise Verschieben solange wiederholt, wie die Taste gedrückt bleibt. Jetzt drücken Sie bitte die Abwärtstaste, um die gesamte Anzeige um eine Zeile nach unten zu verschieben.

So sieht jetzt die Bildschirmanzeige aus:

```
CLEAR WS
      2+3
5
      125+75
200   _
```

Der Wert 200 befindet sich jetzt in der Eingabezeile und kann wie ein eingetasteter Wert verwendet werden. Eingegebene Werte können in jeder beliebigen Stelle der Zeilen 0 und 1 stehen.

Nun drücken Sie bitte folgende Tasten:



Auf dem Bildschirm erscheint:

```
CLEAR WS
      2+3
5
      125+75
200   +50
250
      _
```

Sie sind jetzt sicher über die Arbeitsweise des Bildschirms informiert.

Fehlerkorrektur eingetasteter Zeichen

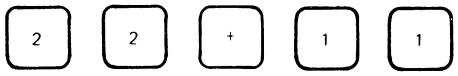
Die IBM 5100 hat eine Reihe nützlicher Einrichtungen, mit denen Sie eingetastete Daten vor ihrer Verarbeitung korrigieren können. Zeile für Zeile und zu jeder Zeit können Sie:

- Zeichen ändern
- Zeichen entfernen
- Zeichen einfügen

Ändern eines Zeichens

Um ein Zeichen zu ändern, wird der Positionsanzeiger mit der Rücktaste  oder der Vorwärtstaste  auf die Position des fehlerhaften Zeichens gebracht. Der Positionsanzeiger wandert um eine Position in Pfeilrichtung, wenn Sie die Tasten einmal drücken. Durch mehrmaliges Drücken dieser Tasten erreichen Sie jede gewünschte Position. Wenn der Positionsanzeiger an der Stelle angelangt ist, an der Sie ein Zeichen verändern wollen, drücken Sie jetzt einfach das richtige Zeichen. Wenn Sie es eilig haben, können Sie auch die Dauerfunktion ausnutzen; Sie drücken einfach die Vorwärtstaste  oder die Rücktaste  etwas länger. Der Positionsanzeiger gleitet solange in die von Ihnen gewünschte Richtung, wie Sie Ihren Finger auf der Taste halten. Bitte ausprobieren!

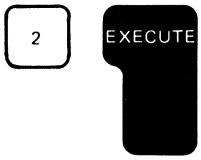
Ein Beispiel: Sie möchten 22 und 12 addieren, drücken Sie bitte folgende Tasten:



Auf dem Bildschirm erscheint:

```
22+11_
```

Um die falsche 1 zu überschreiben, drücken Sie jetzt einmal die Rücktaste . Sie haben sicher bemerkt, daß der Positionsanzeiger verschwunden ist und die falsche 1 angefangen hat, zu blinken. Das blinkende Zeichen hat die gleiche Funktion wie der Positionsanzeiger; es zeigt die Position an, wo das nächste Zeichen, das eingetastet wird, stehen wird. Jetzt korrigieren Sie den Fehler und führen die richtige Operation durch. Drücken Sie einfach folgende Tasten:



Löschen eines Zeichens

Um ein Zeichen zu löschen, benutzen Sie einfach wieder die Rücktaste  oder die Vorwärtstaste . Der Positionsanzeiger wird über das zu löschende Zeichen gebracht; dieses Zeichen blinkt jetzt. Man drückt die CMD-Taste, und währenddessen wird die Rücktaste einmal getippt. Das Zeichen ist gelöscht, und alle Zeichen rechts vom gelöschten Zeichen werden um eine Position nach links verschoben, um die Leerstelle, die das gelöschte Zeichen hinterlassen hat, zu schließen.

Probieren Sie an Hand dieses Beispiels das Löschen von Zeichen selbst aus. Wir wollen 13 + 45 rechnen, aber drücken Sie folgende Tasten:



Auf dem Bildschirm erscheint:

```
123+45_
```

Drücken Sie die Rücktaste und verschieben den Positionsanzeiger nach links bis zur 2, die nun blinkt. Über der Rücktaste und der Vorwärtstaste sehen Sie die Beschriftung: LÖSCHEN und EINFÜGEN. Um die 2 zu löschen, drücken Sie einfach die CMD-Taste und halten diese fest und gleichzeitig drücken Sie die ← -Taste einmal. Jetzt lassen Sie die CMD-Taste wieder los.

Auf dem Bildschirm steht folgendes:

```
13+45
 ↖
```

Dieses Zeichen blinkt jetzt.

Drücken Sie bitte die EXECUTE-Taste, und die Rechenoperation wird durchgeführt.

Einfügen eines Zeichens

Wenn wir ein Zeichen einfügen wollen, bewegen wir den Positionsanzeiger, wie oben beschrieben, mit der Rücktaste ← oder mit der Vorwärtstaste → ; dann drückt man die CMD-Taste und gleichzeitig die Vorwärtstaste → einmal. Dadurch werden das blinkende Zeichen und alle rechts davon stehenden Zeichen um eine Stelle nach rechts verschoben. Wo vorher das blinkende Zeichen stand, befindet sich jetzt eine Leerstelle; der Positionsanzeiger befindet sich an der gleichen Stelle. Hier kann jetzt ein neues Zeichen eingefügt werden.

Wir wollen ein Beispiel selbst durchführen. Sie möchten 123 und 6 multiplizieren, aber Sie geben ein:



Auf dem Bildschirm steht:

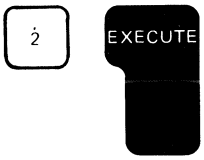
```
13×6_
```

Um den Fehler zu beheben, drücken Sie die Rücktaste und verschieben den Positionsanzeiger zurück zur 3, die jetzt blinkt. Um eine Leerstelle zum Einfügen eines weiteren Zeichens zu erhalten, drücken Sie die CMD-Taste und die -Taste gleichzeitig.

Auf dem Bildschirm erscheint:

```
1_3×6
```

Um die Rechnung richtig zu beenden, tippen Sie:



Hier noch ein anderer Weg, um Eintastfehler zu berichtigen. Nehmen wir an, Sie haben sich mehrmals in einer Zeile vertippt, so bewegen Sie den Positionsanzeiger mit der Rücktaste auf die Stelle zurück, wo der letzte Eingabefehler ist. Dieses Zeichen blinkt jetzt. Wenn Sie nun die Unterbrechungstaste (ATTN-Taste) drücken, ist nicht nur das Zeichen, auf dem der Positionsanzeiger steht, gelöscht, sondern auch alles, was rechts von diesem Zeichen steht.

Eingetastete Daten werden zwar angezeigt, aber erst verarbeitet, wenn man die EXECUTE-Taste drückt. Sie haben also vor der Verarbeitung die Möglichkeit, Ihre Eingaben visuell zu prüfen. Es kann ja auch passieren, daß Sie einen Eingabefehler erst bemerken, nachdem Sie die EXECUTE-Taste gedrückt haben. Sie können die Eingabe komplett wiederholen und die EXECUTE-Taste drücken; Sie können aber auch die Abwärtstaste  drücken und somit die zu korrigierende Zeile in die Eingabezeile zurückholen und, wie vorher beschrieben, Zeichen ändern, einfügen oder entfernen. Nach durchgeführter Korrektur drücken Sie wieder die EXECUTE-Taste.

Sie möchten zum Beispiel 135 und 280 addieren, aber Sie geben 134 und 280 ein und haben die EXECUTE-Taste bereits gedrückt. Auf dem Bildschirm steht:

```
        134+280
414
        _
```

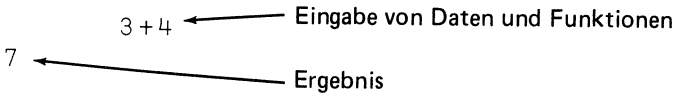
Um die Eingabe zu korrigieren, drücken Sie die Abwärtstaste  dreimal, um die falsche Eingabe aus dem Bildschirm zu entfernen. Der Bildschirm zeigt an:

```
134+280
```

Jetzt drücken Sie die Aufwärtstaste  und bringen damit die ursprüngliche Eingabe in die erste Eingabezeile zurück. Berichtigen Sie den Fehler nun wie vorher beschrieben.

Von jetzt an wollen wir die Form der Darstellung vereinfachen. In den Beispielen erscheinen nicht mehr die Tasten für Ziffern, Rechenfunktionen und EXECUTE. In dieser Einführungsschrift werden die Ergebnisse so dargestellt, wie Sie auf dem Bildschirm Ihrer IBM 5100 tatsächlich abgebildet sind.

Beispiel



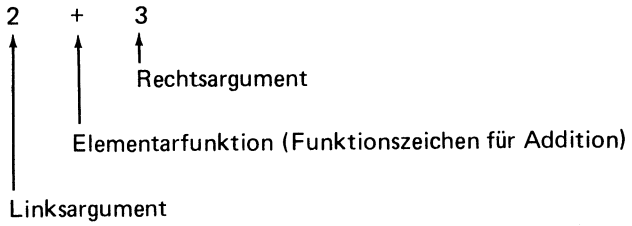
Erinnern Sie sich bitte, daß eingetastete Daten erst dann verarbeitet werden können, wenn Sie die EXECUTE-Taste gedrückt haben.

Kapitel 2 Einführung in die APL-Sprache

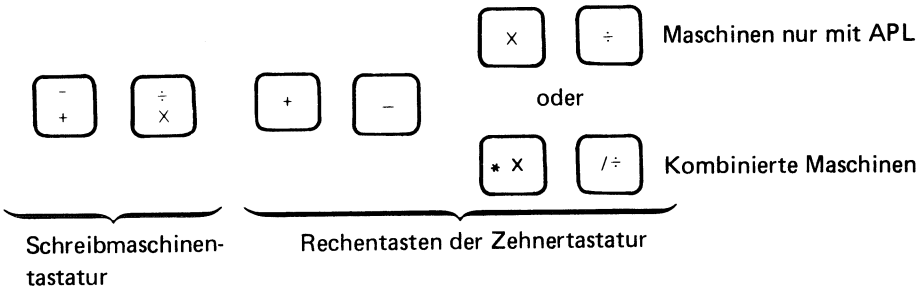
Arten von APL-Funktionen

Es gibt zwei Arten von Funktionen im APL: Von Ihnen selbst definierte Funktionen (Programme) und solche, die standardmäßig in APL vorhanden sind, sogenannte Elementarfunktionen. Die APL-Elementarfunktionen werden durch besondere Symbole dargestellt. Wie man eigene Funktionen erstellt, wird im Kapitel 7 besprochen.

Die Elementarfunktionen verarbeiten Daten, die links und rechts vom Funktionszeichen stehen. Diese Daten werden Argumente genannt. Zum Beispiel:

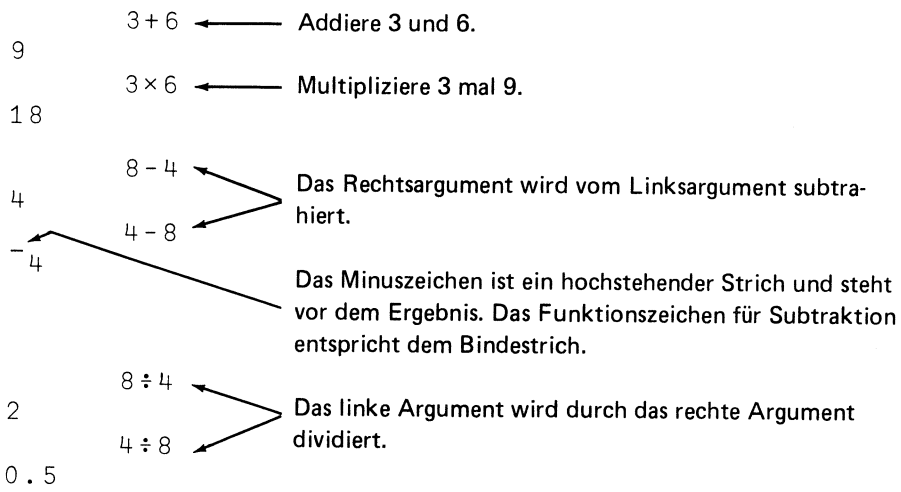


Addition, Subtraktion, Multiplikation und Division



Die vier Elementarfunktionen (+ − × ÷) werden verwendet, um die vier Grundrechenarten auszuführen. Die Tasten sind doppelt vorhanden, sie befinden sich einmal in der oberen Reihe der Schreibmaschinentastatur und außerdem rechts neben der Zehnertastatur.

Beispiele



Wie Sie sehen, unterscheidet sich das Minuszeichen vom Funktionszeichen. Verwenden Sie daher nicht das Funktionszeichen für die Subtraktion zur Kennzeichnung von negativen Zahlen oder das Minuszeichen [-] nicht für die Subtraktion.

Aufgaben: Üben Sie Addition, Subtraktion, Multiplikation und Division

1. Ermitteln Sie die Gesamtzahl von PKW's eines Autoverkäufers während einer Woche; die täglichen Verkaufszahlen sind 3, 5, 2, 6, 7, 3 und 4.

2. Errechnen Sie die Anzahl der Aufträge des gleichen Verkäufers, die keinen Gebrauchtwagen in Zahlung gaben; 20 Käufer gaben ihren alten Wagen in Zahlung.

3. Wie hoch ist der Durchschnittsgewinn pro Woche bei einem Gesamtgewinn von DM 2700? (Gleiche Verkaufszahlen wie in Aufgabe 1).

4. Die Verkaufsprämie pro PKW beträgt DM 20, wie hoch ist die Gesamtprämie?

Mögliche Lösungen

Aufgabe 1:

```
      3+5+2+6+7+3+4
30
```

Aufgabe 2:

```
      30-20
10
```

Aufgabe 3:

```
      2700÷30
90
```

Aufgabe 4:

```
      20×30
600
```

Potenzen und Wurzeln



Eine weitere arithmetische Funktion ist das Potenzieren. In APL verwenden wir die *-Funktion, um das Produkt einer Potenz zu ermitteln, dabei bedeuten das Linksargument die Basis (Grundzahl) und das Rechtsargument den Exponenten (Hochzahl).

Beispiele

```
      3*2 ◄——— 3 hoch 2
9
      2*3 ◄——— 2 hoch 3
8
```

Radizieren

Sie können die Potenzfunktion * auch für die Wurzelfunktion verwenden. Den Wurzelexponenten erhält man durch eine Division.

Beispiele

```
      4*(1÷2)  ◄──── Die Quadratwurzel aus 4.
2
      4*.5     ◄──── Eine andere Schreibweise für die Ermittlung der
2                    Quadratwurzel, da .5 das gleiche ist wie 1 ÷ 2.
      8*(1÷3)
2     ◄──────────── Die Kubikwurzel aus 8.
```

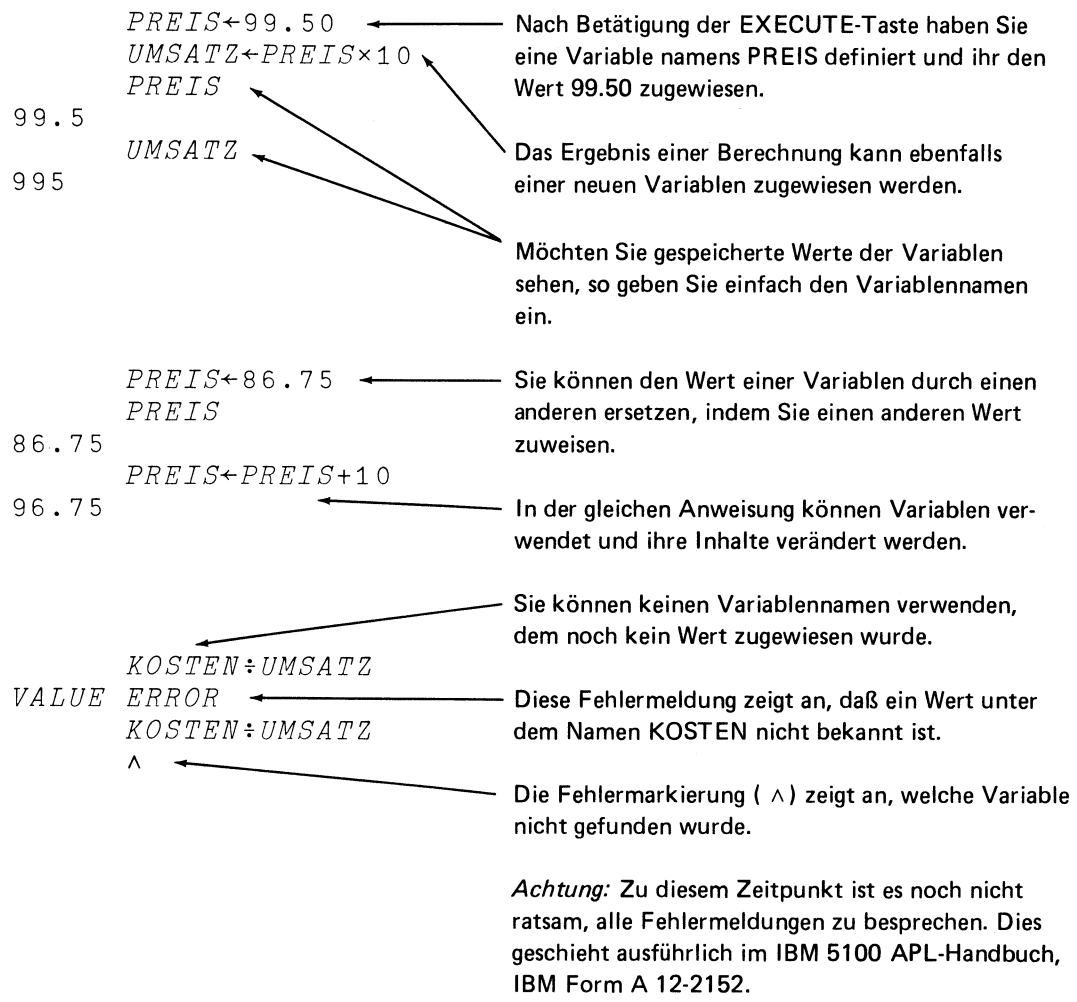
Speicherung von Daten zur späteren Verwendung



Daten, die Sie auf der Tastatur eingeben, oder Rechenergebnisse können zur Weiterverarbeitung in der IBM 5100 gespeichert werden. Diese gespeicherten Daten nennen wir Variable. Jede Variable hat einen Namen, unter dem die gespeicherten Daten wiedergefunden und in Funktion angesprochen werden können. Ein Variablenname kann bis zu 77 Zeichen lang sein (aber ohne Leerstellen darin); das erste Zeichen eines Namens muß immer ein Buchstabe sein, die folgenden Zeichen können beliebige Kombinationen aus Buchstaben und Ziffern sein. Es ist allgemein üblich, daß bei der Wahl von Variablennamen eine Beziehung zum Charakter der zu speichernden Daten hergestellt wird. Wenn Sie zum Beispiel Maße von Längen und Breiten speichern wollen, so könnten Sie den Namen ABMESSUNG vergeben, oder eine Reihe von Preisen könnten Sie PREISE nennen.

Um eine Variable zu erstellen, geben Sie auf der Schreibmaschinentastatur den von Ihnen gewählten Namen ein. Um Werte mit diesem Namen zu verbinden, verwenden wir das Wertzuweisungszeichen, einen Pfeil nach links ←zeigend. Alle Werte, die rechts vom Wertzuweisungszeichen ←stehen, werden dem Namen, der links vom ← Pfeil steht, zugeordnet.

Beispiele



Achtung: Zu diesem Zeitpunkt ist es noch nicht ratsam, alle Fehlermeldungen zu besprechen. Dies geschieht ausführlich im IBM 5100 APL-Handbuch, IBM Form A 12-2152.

Mehrere Funktionen in derselben Anweisung

In den vorhergehenden Beispielen wurde immer nur eine arithmetische Funktion in einer Anweisung gezeigt. Sie können Anweisungen schreiben, die eine ganze Anzahl von verschiedenen arithmetischen Funktionen enthalten können. Wenn Sie dies tun, beachten Sie bitte, daß die Reihenfolge der Durchführung anders ist. Es gilt in APL *nicht* die Regel, Punkt-Rechnen geht vor Strich-Rechnen. Die am weitesten rechts stehende Funktion wird zuerst durchgeführt, es folgt dann die nächste rechts stehende Funktion usw.

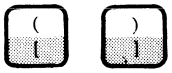
Beispiele

Die Reihenfolge der Durchführung erfolgt von rechts nach links.

```
      3×2+4 ←—— 4 wird zur 2 addiert und das Ergebnis mit 3 multipliziert.
18
      4+3×2 ←—— 3 wird mit 2 multipliziert und das Produkt wird zur 4
10              addiert.
```

Erinnern wir uns, daß eine APL-Anweisung immer das Ergebnis des rechten Ausdrucks auch als rechtes Argument verwendet.

Bestimmung der Reihenfolge der Ausführung durch Klammern



In APL werden Klammern wie beim bekannten Rechnen verwendet: Die Funktion innerhalb der Klammer wir zuerst ausgeführt, die Funktion außerhalb der Klammer anschließend, und zwar von rechts nach links.

Beispiele

```
      (3×2)+4 ←— Der Ausdruck 3 x 2 wir zuerst ausgeführt und das
10               Produkt zur 4 addiert.
      (4+3)×2
14               Der Ausdruck 4 + 3 wir zuerst ausgeführt und das Pro-
                 dukt mit 2 multipliziert.
```

Erinnern wir uns, daß Klammer-Ausdrücke zuerst ausgeführt werden und die außerhalb der Klammer stehenden Funktionen wieder von rechts nach links.

Verwendung von Zahlenreihen und Tabellen

Die APL-Sprache verfügt über Befehle von hoher Mächtigkeit, um Zahlenreihen und Tabellen zu handhaben. Bisher haben wir nur mit alleinstehenden Zahlen (Skalaren) gearbeitet, aber mit APL arbeiten wir sehr häufig mit Zahlenreihen (Vektoren) und Tabellen (Matrizen). Die Funktionen, die wir bisher kennengelernt haben, lassen sich für jedes Element einer Zahlenreihe oder Tabelle verwenden. Sie haben zum Beispiel eine Zahlenreihe mit dem Variablennamen UMSATZ; wenn Sie zu jedem einzelnen Element dieser Zahlenreihe eine 2 addieren wollen, genügt die Eingabe 2+ UMSATZ.

Verwendung von Zahlenreihen (Vektoren) mit APL

Eine Zahlenreihe nennen wir *Vektor.* Während Sie eine Zahlenreihe eingeben, müssen Sie zwischen den einzelnen Zahlen einen Leerschritt einfügen. Jede einzelne Zahl eines Vektors nennen wir Element.

Beispiele:

```
      1 44 2 9 35
1 44 2 9 35
      REIHE←144 16 39 2
      REIHE
144 16 39 2
      MENGEN←125 220 316 90
      MENGEN×10
1250 2200 3160 900
```

Sie haben einen 5-Elemente-Vektor (eine Reihe von fünf Zahlen).

Einem Variablennamen wird ein Vektor zugewiesen.

Jedes Element (Zahl) innerhalb eines Vektors kann mit einer einzelnen Zahl (Skalar) verarbeitet werden.

```
      MENGEN
125 220 316 90
      PREIS←.50 1.00 .75 1.10
      GESAMT←MENGEN×PREIS
      GESAMT
62.5 220 237 99
```

Sie sehen, daß der Inhalt von MENGEN nicht verändert wurde.

Jedes Element in einem Vektor kann mit dem entsprechenden Element eines anderen Vektors verarbeitet werden, vorausgesetzt, daß beide Vektoren die gleiche Anzahl von Elementen haben.

```
      1 2 4+4 5 6
5 7 10
      124+456
580
```

Es müssen unbedingt Leerschritte zwischen allen Elementen eines Vektors sein, sonst wird ein anderes Ergebnis erzielt.

```
      1 2 3+4 5
LENGTH ERROR
      1 2 3 + 4 5
            ^
```

Sie können keine zwei Vektoren mit unterschiedlicher Anzahl von Elementen verarbeiten. Mit einer Ausnahme: Eines der beiden Argumente ist eine einzelne Zahl (Skalar).

Die Fehlermeldung weist auf einen Längenfehler hin.

Aufgaben: Anwendung von Vektoren

1. Quadrieren Sie die Zahlen von 1 bis 5.

2. Wie sind die Produkte aus den Basen 2 und 3 unter Verwendung der Exponenten 2, 3 und 4.

3. Ein Vertreter von Investmentfonds hat sich auf fünf Fonds spezialisiert. Für wieviel DM hat er Anteilscheine von den einzelnen Fonds am Tage verkauft? Um 16.00 Uhr hatte er DM 1500, 3200, 1200, 2300 und 2400 von den fünf Fonds verkauft. In der letzten Stunde des Tages verkauft er noch DM 100, 500, 300, 200, vom letzten 0. Schreiben Sie eine APL-Anweisung über die Gesamtabschlüsse dieses Tages für alle fünf Fonds.

4. Die Preise der einzelnen Investmentzertifikate der in Aufgabe 3 verkauften fünf Fonds betragen: DM 7.30, 11.58, 3.45, 2.17 und 5.56 je Stück. Wieviele Zertifikate hat er je Fond verkauft?

5. Der Vertreter erhält für die fünf verschiedenen Fonds folgende Provisionssätze: 2.25, 2.5, 3.0, 3.75 und 3.5 %. Wieviel hat er an den einzelnen Fonds verdient; wie groß ist die Gesamtprovision?

Mögliche Lösungen

Aufgabe 1:

```
      1 2 3 4 5*2
1 4 9 16 25
```

Aufgabe 2:

```
      2 3*2
4 9
      2 3*3
8 27
      2 3*4
16 81
```

oder

```
      2*2 3 4
4 8 16
      3*2 3 4
9 27 81
```

Aufgabe 3:

```
      1500 3200 1200 2300 2400+100 500 300 200 0
1600 3700 1500 2500 2400
```

Aufgabe 4:

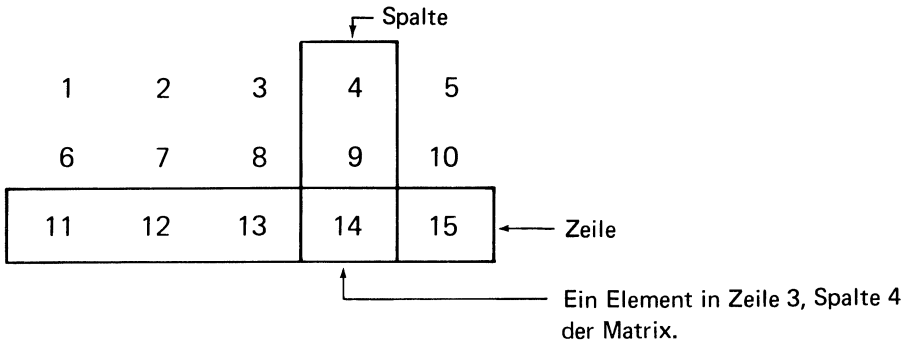
```
      1600 3700 1500 2500 2400÷7.30 11.58 3.45 2.17 5.56
219.18 319.52 434.78 1152.1 431.65
```

Aufgabe 5:

```
      1600 3700 1500 2500 2400×.0325 .0250 .0300 .0375
52 92.5 45 93.75 84
      52.00+92.50+45.00+93.75+84.00
367.25
```

Verwendung von Tabellen (Matrizen)

Eine Tabelle von Zahlen nennen wir eine Matrix. Die Zahlen in einer Matrix sind in Zeilen und Spalten angeordnet; jede Zahl ist ein Element der Matrix.



Um eine Matrix zu bilden, verwenden wir immer die Strukturierfunktion ρ. Das linke Argument gibt die Anzahl Zeilen und Spalten an, und das rechte Argument enthält Daten oder den Namen einer Variablen, mit denen die Matrix gefüllt werden soll.

Beispiele

```
      2 3ρ1 2 3 4 5 6
1 2 3
4 5 6
```

Die erste Zahl des linken Arguments nennt die Anzahl der Zeilen; die zweite Zahl gibt die Anzahl der Spalten an.

Das rechte Argument enthält die Daten, so, wie sie zeilenweise Element für Element in die Matrix eingesetzt werden.

Zwischen der Eingabe von Zeilen und Spalten muß immer eine Leerstelle sein.

```
      TABELLE←2 3ρ1 2 3 4
      TABELLE
1 2 3
4 1 2
```

Es sind nicht genügend Elemente vorhanden, um die Matrix zu füllen. Die Elemente werden wiederholt, bis die Matrix voll ist.

```
      VEKTOR←1 2 3 4 5 6 7 8 9 10 11 12
      MATRIX←3 3ρVEKTOR
      MATRIX
1 2 3
4 5 6
7 8 9
```

Sind mehr Elemente beim rechten Argument vorhanden als die Matrix aufnehmen kann, so werden die übrigen rechten Elemente nicht verwendet.

Die Strukturierfunktion ρ kann auch zum Bilden eines Vektors verwendet werden.

```
      VIER←4ρVEKTOR
      VIER
1 2 3 4
```

Die Anzahl der Elemente des Vektors.

```
      MATRIX+10
11 12 13
14 15 16
17 18 19
```

Jedes Element einer Matrix kann mit einer einzelnen Zahl verarbeitet werden. Der Inhalt von MATRIX wird dadurch nicht verändert.

```
      ZAHLEN←1 2 3 4 5 6 7 8 9
      BEISPIELE←3 3ρZAHLEN+5
      BEISPIELE
 6  7  8
 9 10 11
12 13 14
```

Erinnern Sie sich bitte, daß APL eine Anweisung von rechts nach links abarbeitet. Das Ergebnis von ZAHLEN +5 wird als rechtes Argument für die Strukturierfunktion ρ verwendet.

```
      ERGEBNISSE←BEISPIELE+MATRIX
      ERGEBNISSE
 7  9 11
13 15 17
19 21 23
```

Jedes Element einer Matrix kann mit dem korrespondierenden Element einer anderen Matrix mit gleicher Struktur verarbeitet werden.

Beispiele (Fortsetzung)

```
      AB←2 2⍴ZAHLEN
      AB
1 2
3 4

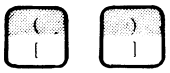
      MATRIX×ZAHLEN
RANK ERROR
      MATRIX×ZAHLEN
      ∧
      MATRIX×AB
LENGTH ERROR
      MATRIX×AB
      ∧
```

Eine Matrix und ein Vektor

oder

zwei Matrizen, die nicht die gleichen Strukturen (Anzahl Zeilen und Spalten) haben, können nicht miteinander verarbeitet werden. Eins der Argumente kann aber eine einzelne Zahl (Skalar) sein.

Verwendung einzelner Elemente aus Vektoren oder Matrizen (Indizierung)



Die Auswahl einzelner oder mehrerer Elemente aus Vektoren oder Matrizen ist durch Angabe der Positionen der Elemente möglich. Wir sprechen vom Indizieren. Die Positionsnummer der Elemente nennt man Indexwerte oder Indizes. Die Indizes werden in eckigen Klammern eingeschlossen und stehen rechts von Vektoren oder Matrizen.

Beispiele

```
      TEMP←68 74 78 65 80 85 72
      TEMP[2]
74
      TEMP[3 1 2]
78 68 74

      TEMP[2]+TEMP[3 1 2]
152 142 148

      TEMP[7]←88
      TEMP
68 74 78 65 80 85 88
      TEMP[3 6]←70
      TEMP[3 6]
70 70
```

Ein einzelnes Element wird ausgewählt.

Mehrere Elemente werden ausgewählt. Die Reihenfolge kann dabei verändert werden.

In der gleichen Anweisung kann jedes indizierte Element verarbeitet werden.

Ein einzelnes Element eines Vektors kann ausgetauscht werden.

Mehrere Elemente können ausgetauscht werden.

Beispiele (Fortsetzung)

```
      TEMP[5 1]←32 56
      TEMP
56 74 70 65 32 70 88
```

Neue Werte werden in der gleichen Reihenfolge wie die der Indexwerte ausgetauscht.

Bei Matrizen benötigt man einen Indexwert für Zeilen und einen für Spalten. Indexwerte werden durch ein Semikolon getrennt.

```
      MATRIX←3 3⍴ZAHLEN
      MATRIX
1 2 3
4 5 6
7 8 9
```

Vorher haben wir die Werte der Variablen ZAHLEN zugewiesen.

Links vom Semikolon wird die Zeile angegeben.

Rechts vom Semikolon wird die Spalte spezifiziert.

Ein einzelnes Element wird ausgewählt

```
      MATRIX[2;2]
5
```

oder

```
      MATRIX[3;2 3]
8 9
```

mehrere Elemente werden indiziert. In diesem Beispiel wird das zweite und dritte Element der dritten Zeile angesprochen.

```
      MATRIX[1;1 2]
1 2
      MATRIX[2 3;2 3]
  5 6
  8 9
```

Werden mehr als eine Zeile und mehr als eine Spalte indiziert, ist das Ergebnis eine Matrix.

Beispiele (Fortsetzung)

```
      MATRIX[2;]
4 5 6
      MATRIX[;3]
3 6 9
```

Wenn keine Spalte angegeben wird, wird die ganze Zeile ausgewählt.

Wenn keine Zeile indiziert wird, so wird die Spalte insgesamt ausgewählt.

```
      MATRIX[;1 3]
1 3
4 6
7 9
```

Die Werte der dritten Spalte werden horizontal angezeigt, weil es sich um eine Zahlenreihe (Vektor) handelt.

Achtung: Bei Auswahl von Zeilen oder Spalten ist in jedem Falle ein Semikolon nötig, um die Indizes von Zeilen und Spalten zu erkennen und zu unterscheiden.

```
      MATRIX[2;1]×MATRIX[3;3]
36
      MATRIX[2;2]←0
      MATRIX
1 2 3
4 0 6
7 8 9
      MATRIX[1;1]←2+3
      MATRIX[1;1]
5
```

In der gleichen Anweisung kann jedes Element mit jedem Element verarbeitet werden.

Einzelne oder mehrere Elemente einer Matrix können verwendet werden.

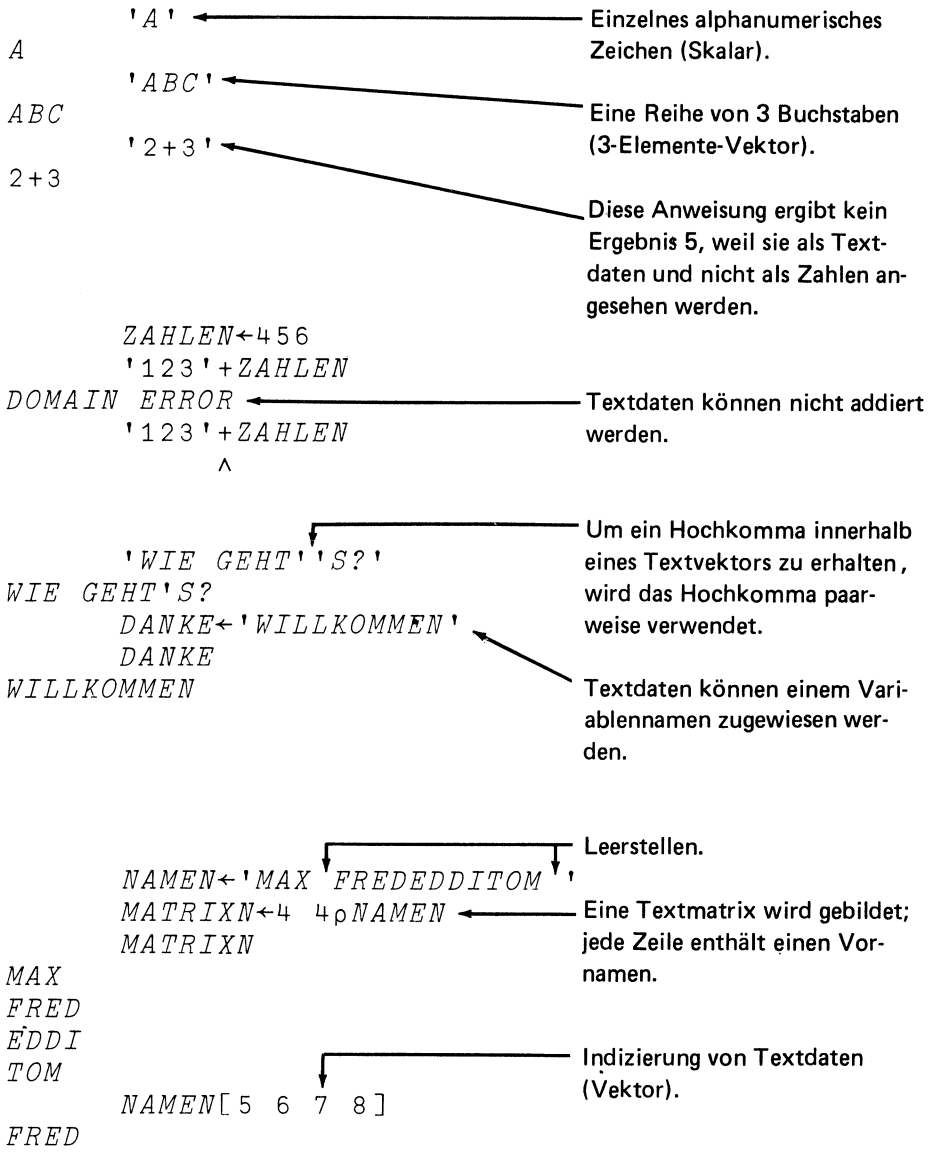
Nicht nur Zahlen können verarbeitet werden



Bei allen bisherigen Beispielen haben wir numerische Daten verarbeitet; APL kann auch alphanumerische Daten (Textdaten) verarbeiten. Textdaten verwenden wir zum Beispiel für Überschriften oder Namenslisten. Bei der Eingabe von Textdaten werden diese in Hochkomma ' eingeschlossen. Das Hochkomma zeigt an, daß es sich um Textdaten und nicht um Variablennamen einer Zahl oder einer Funktion handelt. Werden Textdaten angezeigt, erscheint kein Hochkomma.

Textdaten können wie numerische Daten ein einzelnes Zeichen (Skalar), eine Reihe von Zeichen (Vektor) oder eine Tabelle mit Zeichen (Matrix) sein. Im Gegensatz zu numerischen Daten sind in Textvektoren oder Textmatrizen alle Zeichen als Elemente anzusehen. Die Elemente werden nicht durch Leerzeichen getrennt. Ein Leerzeichen in einer Textmatrix oder einem Vektor ist ebenso ein einzelnes Element.

Beispiele



Bisher haben Sie nur die in APL vorhandenen allgemeinen Rechenarten kennengelernt. Sie haben gesehen, wie APL mit Skalaren (einzelne Zahl), Vektoren (Zahlenreihen) und Matrizen (Tabellen) arbeitet. APL hat außer den bisher besprochenen Funktionen noch eine große Zahl weiterer interessanter Funktionen, die in den folgenden Kapiteln besprochen werden. Sie werden sehen, APL kann noch viel mehr.

Kapitel 3 APL-Funktionen mit einem Argument

In diesem Kapitel werden Sie einige APL-Funktionen kennenlernen, mit denen Sie folgendes bewirken können:

- Abrunden und Aufrunden.
- Sortieren eines Vektors in aufsteigender oder absteigender Folge.
- Zufallszahl auswählen.
- Bilden von fortlaufenden Zahlenreihen.
- Dimension einer Variablen bestimmen.

Es gibt noch weitere APL-Funktionen, die mit einem Argument arbeiten; sie werden in Kapitel 6 besprochen.

Wieviele Argumente sind bei den Elementarfunktionen nötig?

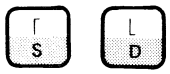
In diesem Kapitel werden Sie APL-Funktionen sehen, die mit einem Argument arbeiten. Im nächsten Kapitel werden einige dieser APL-Funktionszeichen mit zwei Argumenten verwendet. Sie werden sehen, daß diese gleichen Symbole sehr verschiedene Funktionen darstellen, je nachdem, ob mit einem Argument oder mit zwei Argumenten gearbeitet wird. Wenn Sie eine APL-Funktion mit einem Argument verwenden, so muß das Argument rechts vom Funktionszeichen stehen.

APL-Funktionen, die aus zwei Zeichen zusammengesetzt sind

Einige APL-Funktionen, die Sie benutzen werden, stellen eine Kombination von zwei Zeichen dar. Sie erinnern sich sicher noch, daß bei der Korrektur von eingetasteten Zeichen der Positionsanzeiger auf die Stelle des zu korrigierenden Zeichens gebracht werden mußte. Das neu eingetastete Zeichen ersetzt dann das vorherige. APL-Funktionszeichen, die aus zwei Zeichen zusammengesetzt sind, werden einfach übereinandergetippt. Bitte die Rücktaste auf der Schreibmaschinentastatur benützen. Es ist völlig gleichgültig, welches Zeichen zuerst eingetastet wird. Funktionen, die mit zwei Zeichen dargestellt werden, nennt man Mehrfachzeichen. In Anhang A sind alle zusammengesetzten Zeichen aufgeführt.

Wenn Sie ein Mehrfachzeichen eingegeben haben und es durch ein anderes ersetzen wollen, so bewegen Sie den Positionsanzeiger auf diese Stelle und geben ein neues Zeichen ein. Das Mehrfachzeichen ist dann gelöscht.

Auf- und Abrunden



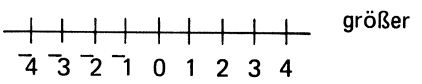
Wollen Sie eine Dezimalzahl mit Stellen hinter dem Komma zu einer ganzen Zahl auf- oder abrunden, so verwenden Sie das Funktionszeichen ⌊ und ⌈. Das ⌊ -Zeichen wird verwendet, wenn man alle Stellen nach dem Komma abschneiden will. Die erste Stelle vor dem Komma bleibt unverändert. Das ⌈ -Zeichen bewirkt das Aufrunden auf die nächstgrößere ganze Zahl; die Stellen nach dem Komma werden abgeschnitten.

Beispiele

```
      B←3.529
      ⌊B
3
      ⌈B
4
      C←3
      ⌊C
3
      ⌈C
3
      B←¯3.529
      ⌊B
¯4
      ⌈B
¯3
```

Ist die Zahl eine ganze Zahl, so ist das Ergebnis das Gleiche wie das rechte Argument.

Das Auf- und Abrunden von negativen Dezimalzahlen geschieht entsprechend dem Zahlenstrahl:



Kaufmännisches Runden

In der kaufmännischen Praxis wird bei .4 ab- und bei .5 aufgerundet. Wir verwenden hierzu die ⌊ - Funktion und addieren .5 zur Zahl, die gerundet werden soll.

Beispiele

```
      X←4.4+.5
      ⌊X
4
      X←4.6+.5
      ⌊X
5
      ⌊X←4.4+.5
4
      X←⌊4.6+.5
      X
5
```

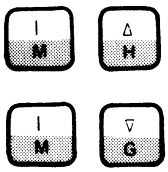
Runde 4.4 zur nächsten ganzen Zahl.

Runde 4.6 zur nächsten ganzen Zahl.

Diese Beispiele führen zu gleichen Ergebnissen wie die ersten beiden.

Sortieren eines Vektors in auf- oder absteigender Folge

Zum Sortieren eines Vektors verwenden wir die Funktionszeichen ⍋ für aufsteigendes und ⍒ für absteigendes Sortieren. Das Ergebnis ist ein Indexvektor; die Funktion heißt: Sortierindexvektor bilden.



Beispiele

```
      A←80 45 62 37 29 74 58 15 96
      ⍋A
8 5 4 2 7 3 6 1 9
```

Der größte Wert ist das 9. Element.

Der kleinste Wert ist das 8. Element.

```
      B←A[⍋A]
      B
15 29 37 45 58 62 74 80 96
```

Durch die Indizierung von A werden die Elemente von A in aufsteigender Folge sortiert.

Beachten Sie bitte: Bei der Indizierung der Elemente eines Vektors müssen die Indizes oder die Funktion für die Sortierindexbildung in eckigen Klammern [ ] stehen.

```
      ⍒A
9 1 6 3 7 2 4 5 8
      C←A[⍒A]
      C
96 80 74 62 58 45 37 29 15
```

Die Elemente von A sind in absteigender Folge sortiert worden.

Zufallszahl auswählen



Um eine Zufallszahl auszuwählen,verwenden wir die ? -Funktion. Die Zufallszahl wird ausgewählt zwischen 1 und dem Wert des rechten Arguments.

Beispiele:

```
      X←?6
      X
5
      X←?6 6 6 6
      X
1 3 1 3
```

Zufallszahl zwischen 1 und 6 auswählen.

Das Ergebnis ist irgendeine Zahl zwischen 1 und 6.

Wenn in dieser Funktion ein Vektor angegeben wird, so wird für jedes Element eine Zufallszahl ausgewählt.

Bilden von fortlaufenden Zahlenreihen



Es kann oft vorkommen, daß Sie eine laufende Zahlenreihe benötigen. Geben Sie einfach ein:

```
      VEKTOR←1 2 3 4 5 6 7 8
      VEKTOR
1 2 3 4 5 6 7 8
```

Diese sehr wichtige Funktion bezeichnen wir mit Indexvektor bilden ∿. Es wird ein Indexvektor mit laufenden Nummern von 1 bis zum Wert des rechten Arguments gebildet.

Beispiele:

```
      ⍳8
1 2 3 4 5 6 7 8
      VEKTOR←5+⍳5
      VEKTOR
6 7 8 9 10
      2*⍳6
2 4 8 16 32 64
```

Acht fortlaufende Zahlen.

Zu jeder laufenden Zahl wird eine 5 addiert.

$2^1, 2^2, 2^3, 2^4, 2^5, 2^6$

Bildung eines leeren Vektors

Ein leerer Vektor ist ein Vektor ohne Inhalt (keine Elemente). Wir werden später noch die Verwendung eines leeren Vektors kennenlernen. Eine Möglichkeit, einen leeren Vektor zu bilden, ist die Verwendung von ⍳0.

Beispiele

```
      NAME
VALUE ERROR
      NAME
      ∧
```

Diese Fehlermeldung erscheint, wenn Sie einen Variablennamen verwenden, dem noch kein Wert zugewiesen wurde.

```
NAME←⍳0
NAME
```

Bildung eines leeren Vektors.

Eine Leerzeile auf dem Bildschirm (kein Wert vorhanden).

Bestimmen der Dimension von Variablen



Wie Sie im Kapitel 2 bereits gesehen haben, wird durch das linke Argument bei der Funktion Strukturieren die Anzahl der Elemente eines Vektors und die Anzahl Zeilen und Spalten einer Matrix bestimmt. Die Anzahl der Elemente eines Vektors oder einer Matrix stehen im Zusammenhang mit der Dimension eines Vektors oder einer Matrix; zum Beispiel die Dimension der Matrix M, bestehend aus zwei Zeilen und drei Spalten, ist: 2 3. Um die Dimension einer vorhandenen Variablen zu finden, verwenden wir die Funktion Dimension zeigen ⍴.

Beispiele

```
      SKALAR←4
      VEKTOR←2 4 6 8
      MATRIX←2 3⍴⍳6
      ⍴SKALAR
```

Die Funktion Strukturieren hat zwei Argumente.

```
      ⍴VEKTOR
4
```

Eine Leerzeile – die Dimension eines Skalars ist ein leerer Vektor.

```
      ⍴MATRIX
2 3
```

Anzahl Elemente eines Vektors.

Anzahl Zeilen und Spalten einer Matrix.

```
LEER←⍳0
LEER
```

Bilden eines leeren Vektors.

Leerzeile auf dem Bildschirm

```
⍴LEER
0
```

Anzahl von Elementen ist ein leerer Vektor.

Kapitel 4 APL-Funktionen mit zwei Argumenten

In diesem Kapitel wollen wir weitere APL-Funktionen besprechen, in denen zwei Argumente verwendet werden. Mit diesen neuen Funktionen können Sie:

- Argumente vergleichen (gleich, größer, kleiner)
- Logische Daten verknüpfen (1 ist = wahr und 0 ist = unwahr).
- Die größere von zwei Zahlen bestimmen.
- Die kleinere von zwei Zahlen bestimmen.
- Sich den Index eines Wertes in einem Vektor zeigen lassen.
- Stichproben nehmen.
- Einen Vektor oder eine Matrix komprimieren.
- Einen Vektor oder eine Matrix expandieren (durch 0 oder Leerzeichen erweitern).
- Zwei oder mehrere Variablen verketten.
- Logarithmieren.

Es gibt noch weitere APL-Funktionen, die zwei Argumente benötigen; sie werden in Kapitel 6 eingehend behandelt.

Vergleichsfunktionen

Bei der Lösung der verschiedensten Aufgaben mit APL werden Sie immer wieder Vergleiche von zwei Werten vornehmen müssen. Sie wollen zum Beispiel in einem Programm testen, ob ein Zähler einen bestimmten Wert erreicht hat; oder Sie wollen abhängig davon, ob eine Bedingung zutrifft oder nicht, unterschiedliche Berechnungen durchführen. Die folgenden APL-Funktionen werden verwendet, um die Abhängigkeit von zwei Werten zu testen:

Funktionen	Zeichen	Tasten
Größer als	>	> 7
Kleiner als	<	< 3
Größer oder gleich	≥	≥ 6
Kleiner oder gleich	≤	≤ 4
Gleich	=	= 5
Ungleich	≠	≠ 8

Werden diese Funktionen eingesetzt, ist das Ergebnis entweder 1, d. h. die Bedingung trifft zu, oder 0, die Bedingung trifft nicht zu.

Beispiele

```
      A←10
      B←20
      A=B
0
      A=B-10
1
      A≤B
1
      A≠B
1
      A≥B
0
      A←'ABC'
      B←'DEF'
      A=B
0 0 0
      A≠B
1 1 1
```

Die = und ≠ Vergleichsoperatoren können auch für Textdaten verwendet werden. Jedes Element wird mit dem korrespondierenden Element des anderen Arguments verglichen.

Warum zwei gleiche Zahlen doch nicht gleich sind

APL speichert alle numerischen Werte intern mit einer Genauigkeit von 15 Dezimalstellen. Dezimalwerte mit mehr als 5 Stellen werden, bevor sie auf dem Bildschirm angezeigt werden, aufgerundet. Das bedeutet nicht, daß zwei Werte wirklich gleich sind; in den nicht angezeigten Stellen können unterschiedliche Werte vorhanden sein.

Beispiele

```
      A←1÷3
      B←.33333
      A
0.33333
      B
0.33333
      A=B
0
      ⎕PP
5
```

Nur fünf der 15 Ziffern werden angezeigt.

Die Werte sind nicht gleich.

⎕ PP ist eine *Systemvariable*, die angibt, wieviele Stellen angezeigt werden sollen. Diese Variable wird automatisch gesetzt, wenn die Maschine eingeschaltet wird oder RESTART gedrückt wird. (Die Systemvariablen werden in Kapitel 9 besprochen.)

```
      ⎕PP←15
      A
0.333333333333333
      B
0.33333
      ⎕PP←5
```

Der Systemvariablen ⎕ PP wird der Wert 15 zugewiesen. Es werden jetzt 15 Ziffern angezeigt.

Differenz zwischen den beiden Werten.

⎕ PP wird wieder auf fünf gesetzt.

Beachten Sie: Angezeigte Werte sind nicht exakt die Werte, wie sie in der IBM 5100 gespeichert sind.

Beispiel für eine Vergleichsfunktion

Das korrekte Ergebnis einer Aufgabe ist in einer Variablen mit dem Namen RICHTIG gespeichert. Die Lösung dieser Aufgabe, die ein Schüler angibt, wird mit dem Variablennamen ANTWORT gespeichert. Für jede richtige Antwort dieses Schülers soll eine 1 gezählt werden. Für die falsche Antwort wird keine 1 weitergezählt, der ZAEHLER bleibt unverändert.

Wenn der Schüler die richtige Antwort gibt, dann gilt: ANTWORT=RICHTIG. Eine 1 soll nur dann in den ZAEHLER addiert werden, wenn seine Antwort mit dem Wert in RICHTIG übereinstimmt. Sie schreiben hierzu folgende Anweisung:

```
ZAEHLER ←ZAEHLER+ANTWORT=RICHTIG
```

Damit wird eine 1 zum ZAEHLER addiert, wenn beide Werte gleich sind, und eine 0, wenn sie ungleich waren.

Anstelle der Addition von 1, wenn der Schüler die richtige Antwort gibt, können Sie auch die richtige Antwort anders bewerten. Die Bewertung wird unter der Variablen GEWICHT gespeichert. Wenn der Schüler die richtige Lösung hat, wird das GEWICHT zum Zähler addiert, andernfalls wird der Zähler nicht verändert. Sie können folgende Anweisung schreiben:

```
ZAEHLER=ZAEHLER+GEWICHTxANTWORT=RICHTIG
```

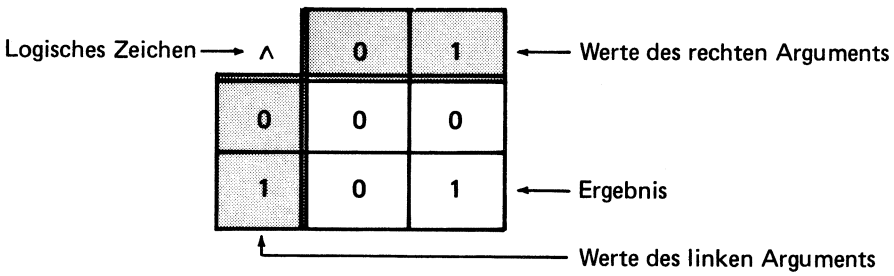
Wenn die Antwort des Schülers richtig ist, dann ergibt ANTWORT=RICHTIG den Wert 1, nun wird GEWICHTx1 zum Zähler addiert. Ist die Antwort ungleich, so wird GEWICHTx0 addiert.

Logische Daten verknüpfen

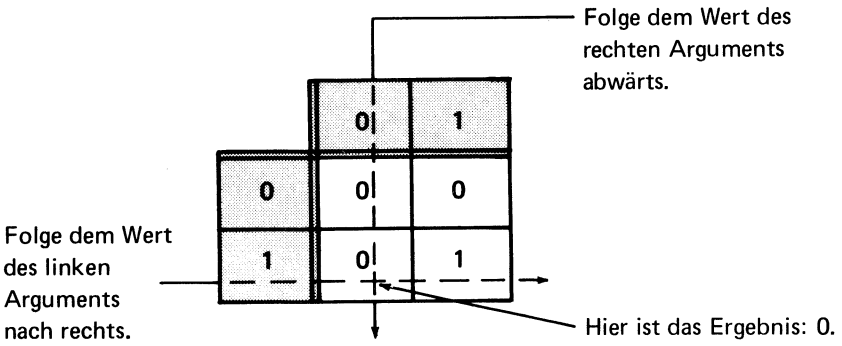
Die Verknüpfungsfunktionen verwenden ausschließlich Einsen und Nullen als Argumente und werden zum Testen von verschiedenen logischen Bedingungen verwendet (hauptsächlich zum Prüfen von Ergebnissen aus Vergleichsfunktionen). Die logischen Zeichen sind:

Funktionen	Zeichen	Tasten
Verknüpfen mit Und	∧	∧ 0
Oder	∨	∨ 9

In unserer Besprechung der Verknüpfungsfunktionen verwenden wir Tabellen, wie sie nachstehend aufgeführt sind:



Die Werte für das rechte Argument befinden sich am Kopf der Tabelle, und die Werte des linken Arguments stehen links neben der Tabelle. Das Ergebnis finden Sie beim Schnittpunkt der Spalte für das rechte Argument und der Zeile für das linke Argument:



Und ∧ 0

∧	0	1
0	0	0
1	0	1

← Rechtes Argument.

← Das Ergebnis ist nur dann 1, wenn beide Argumente 1 sind.

← Linkes Argument.

Die Verknüpfung Und wird verwendet, um festzustellen, ob zwei Bedingungen wahr sind.

Sie wollen zum Beispiel feststellen, ob ein Artikel mehr als DM 100,00 kostet und die Menge kleiner als 10 ist. Sie verwenden dazu folgende Anweisung:

```
(KOSTEN>100)∧(MENGE<10)
```

Das Ergebnis ist 1, wenn die Menge kleiner 10 ist.

Das Ergebnis ist 1, wenn die Kosten größer 100 sind.

Die Klammern in diesem Beispiel sind notwendig, um die richtige Reihenfolge der Durchführung dieser Anweisung zu gewährleisten.

Beispiele:

```
      MENGE←8
      KOSTEN←120
      (KOSTEN>100)∧(MENGE<10)
1 ←————————————————————————————  Beide Bedingungen sind wahr.
      MENGE←25
      (KOSTEN>100)∧(MENGE<10)
0 ←————————————————————————————  Eine der Bedingungen ist nicht wahr.
```

Oder 

∨	0	1
0	0	1
1	1	1

← Rechtes Argument

← Das Ergebnis ist 1, wenn eines der Argumente (oder beide) 1 ist.

Linkes Argument.

Die Verknüpfung mit Oder wird verwendet, um zu prüfen, ob eine von zwei Bedingungen zutrifft.

Wenn Sie zum Beispiel wissen wollen, ob der Bestand eines Artikels kleiner als 10 ist oder die bestellte Menge den Lagerbestand übersteigt, können Sie folgende Anweisung schreiben:

```
(BESTAND<10)∨(BESTMENGE>BESTAND)
```

Das Ergebnis ist 1, wenn die bestellten Mengen größer als der Lagerbestand sind.

Das Ergebnis ist 1, wenn der Lagerbestand kleiner als 10 ist.

Beispiele

```
      BESTAND←15
      BESTMENGE←5
      (BESTAND<10)∨(BESTMENGE>BESTAND)
0
```

Beide Bedingungen sind unwahr.

```
      BESTMENGE←25
      (BESTAND<10)∨(BESTMENGE>BESTAND)
1
```

Eine der Bedingungen ist wahr.

Aufgaben:

Verwende Vergleichs- und Verknüpfungsfunktionen

1. Es ist in allen Weltraumsystemen lebensnotwendig, Fehler sofort zu bemerken, um Katastrophen zu verhindern. Zwei Messwertgeber werden ständig überwacht. Wenn einer der beiden oder beide Messwertgeber einen kritischen Punkt erreichen, müssen sofort bestimmte Maßnahmen eingeleitet werden.

   Wir unterstellen, daß der Messwertgeber A ein Limit von 1.3725 Ampere und Messwertgeber B ein Limit von 1.5365 Ampere nicht überschreiten darf. Schreiben Sie eine Anweisung, die als Ergebnis eine 1 bringt, wenn ein Wert oder beide Messwerte das Limit erreicht oder überschreiten. Die tatsächlichen Messwerte sind 1.3732 für A, 1.5362 für B.

2. Eine Untersuchung wurde durch das staatliche Schulamt durchgeführt. Der Lehrer und die Eltern müssen 10 Fragen zum Verhalten des Kindes mit ja oder nein beantworten.

   Der Lehrer eines Kindes antwortet mit 1,0,1,1,0,1,0,0,1,0 auf die Verhaltensfragen. Die Eltern des Kindes antworten auf die gleichen Fragen mit 1,0,0,1,0,1,1,0,0,0.

   Welche Fragenummern von 1 bis 10 wurden vom Lehrer und den Eltern übereinstimmend beantwortet?

Mögliche Lösungen

Aufgabe 1:

```
      (1.5365≤1.5362)∨1.3725≤1.3732
1
```

Aufgabe 2:

```
      (⍳10)×1 0 1 1 0 1 0 0 1 0∧1 0 0 1 0 1 1 0 0 0
1 0 0 4 0 6 0 0 0 0
```

Die größere von zwei Zahlen bestimmen

Das Ergebnis der Funktion Maximieren ⌈ ist das größere von zwei Argumenten.



Beispiele

```
      A←5
      B←6
      A⌈B
6
      (B×B')⌈A×A
36
```

Ein Warenhauskonzern bietet seinen Kunden die Möglichkeit des bargeldlosen Einkaufs, alle Einkäufe werden auf dem Kundenkonto gebucht. Der Kunde kann zahlen, wann immer es ihm beliebt. Monatlich werden die Kundenkonten überprüft, für die noch nicht bezahlten Beträge wird dem Kunden eine Gebühr von 1,5 % berechnet. Die Differenz zwischen Einkäufen und Zahlungen ist unter dem Variablennamen SALDO gespeichert. Die GEBUEHR für den gewährten Kredit wird wie folgt berechnet:

```
GEBUEHR←SALDO×.015
```

Diese Anweisung ist solange richtig, wie ein Kunde nicht mehr bezahlt, als er eingekauft hat. Bei Überzahlungen würde eine negative Gebühr errechnet werden. Um dies auszuschalten, darf die Berechnung nur ausgeführt werden, wenn der SALDO größer als 0 ist. Hier hilft nun die Funktion Maximieren. Die Anweisung sieht folgendermaßen aus:

```
GEBUEHR←.015×0⌈SALDO
```

Die kleinere von zwei Zahlen bestimmen

Das Ergebnis der Funktion Minimieren ⌊ ist das kleinere von zwei Argumenten.



Beispiele

```
      A←5
      B←6
      A⌊B
5
      (B×B)⌊A×A
25
      C←1 4 7 9
      D←3 2 8 7
      C⌊D
1 2 7 7
```

Aufgabe:

Verwendung der Maximum- und Minimumfunktionen

1. Bestimme die höchste Ausgabe in DM beim Tanken bei folgenden Benzinpreisen:
   a. 27,8 l zum Preise von Pf 88,9 pro l
   b. 22,5 l zum Preise von Pf 91,2 pro l
   c. 26,1 l zum Preise von Pf 90,9 pro l
2. Bei folgenden Preisen ist das niedrigste Gewicht von Nüssen ausfindig zu machen:
   a. DM 4,85 für Walnüsse zu 2,55 das kg
   b. DM 3,30 für Haselnüsse zu 2,15 das kg
   c. DM 3,65 für Erdnüsse zu 1,95 das kg

Welche Nußsorte bringt das niedrigste Gewicht?

Lösungsvorschläge

Aufgabe 1:

```
      (27.8×.889)⌈(22.5×.912)⌈26.1×.909
24.714
```

Aufgabe 2:

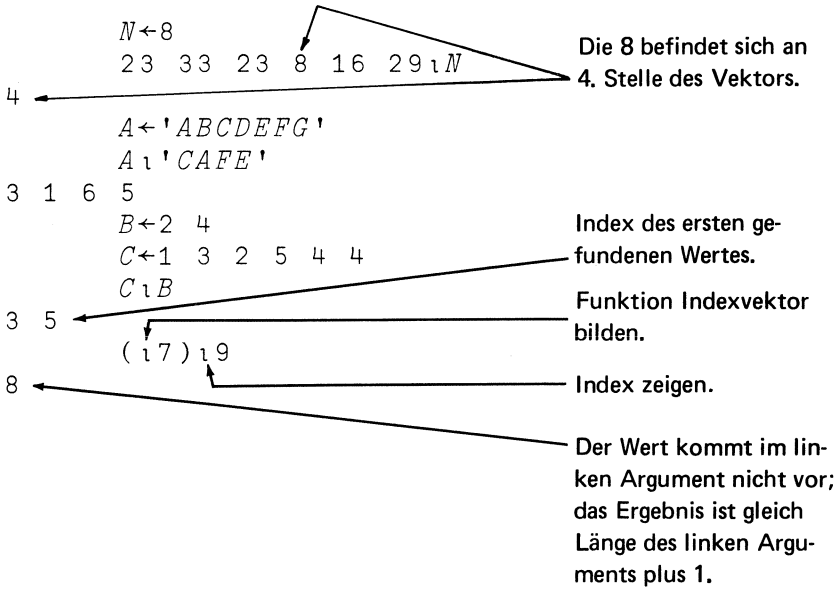
```
      (4.85÷2.55)⌊(3.30÷2.15)⌊3.65÷1.95
1.534
```

Index eines Wertes in einem Vektor anzeigen



Nehmen wir an, Sie möchten wissen, an welcher Stelle (Position) ein bestimmter Wert in einem Vektor steht, so können Sie das mit der Funktion ⍳ feststellen. Das linke Argument ist ein Vektor, aus dem Sie ganz gezielt einen Wert ansprechen möchten. Das rechte Argument ist der gesuchte Wert; befindet sich der gesuchte Wert nicht in dem linken Argument, so ist das Ergebnis gleich dem linken Argument plus 1.

Beispiele



Stichproben nehmen



In Kapitel 3 haben Sie die Funktion Zufallszahlen auswählen benutzt (? mit einem Argument). Beim Einsatz der ? -Funktion mit zwei Argumenten können Sie eine Stichprobe nehmen, ohne daß die gleiche Zahl zweimal ausgewählt wird. Diese Funktion wählt mehrere Zufallszahlen aus, und zwar die Anzahl, die im linken Argument angegeben ist. Die Zahlenfolge beginnt mit 1 bis zu dem Wert, der im rechten Argument angegeben ist. Die Zufallszahlen werden dem Vektor entnommen, so daß eine Zahl in einer Stichprobe nicht zweimal vorkommen kann. Das linke Argument kann nicht größer sein als das rechte Argument. Wenn linkes und rechtes Argument gleich sind, so werden alle Zahlen von 1 bis zum Wert des rechten Argumentes in einer Zufallsfolge angezeigt.

Beispiele

```
      6?49
28 41 3 37 9 17
      10?10
5 3 8 10 6 9 2 4 1 7
```

Lottozahlen für den nächsten Tip; keine Zahl ist doppelt.

Diese Zahlen können in jeder beliebigen Reihenfolge kommen; bitte machen Sie mehrere Versuche.

Vektoren oder Matrizen komprimieren



Die Funktion Komprimieren / wird dazu verwendet, um aus einem Vektor oder aus einer Matrix bestimmte Elemente zu selektieren. Das linke Argument muß immer ein Vektor, bestehend aus Einsen oder Nullen, sein. Wenn wir aus einem Vektor (rechtes Argument) bestimmte Elemente auswählen wollen, muß in derselben Position des linken Arguments des korrespondierenden Elements eine 1 stehen.

Beispiele

```
      V←1 2 3 4 5 6 7 8
      1 0 1 1 0 1 1 1/V
1 3 4 6 7 8
      0 0 0 0 0 0 0 0/V

      A←1 0 0 0 0 0 0 1
      A/V
1 8
      1 0 1 0/'ABCD'
AC
```

Der Vektor wurde komprimiert (verkleinert).

Das Ergebnis ist ein leerer Vektor.

Wir wollen jetzt eine Matrix komprimieren. Hierzu müssen wir festlegen, ob wir Zeilen oder Spalten unterdrücken wollen. Um dies zu erreichen, sagen wir, um welche Koordinate es sich handelt: Die erste Koordinate sind Zeilen [1], und die zweite Koordinate sind Spalten [2]. Die in eckigen Klammern stehenden Zahlen sind Indexwerte. Wenn Sie sich folgende Beispiele anschauen, wissen Sie sofort, was gemeint ist.

Beispiele

```
      B←3 4⍴⍳12
      B
 1  2  3  4
 5  6  7  8
 9 10 11 12
```

Bitte erinnern Sie sich, daß eine 1 ein Element auswählt und eine 0 jedes Element unterdrückt.

```
      0 1 0/[1]B
5 6 7 8
      1 0 1 0/[2]B
 1  3
 5  7
 9 11
      1 0 1 0/B
 1  3
 5  7
 9 11
```

Die erste Koordinate (Zeilen) wurde angegeben.

Die zweite Koordinate (Spalten) wurde angegeben.

Wird kein Index angegeben, so wird die zweite Koordinate (also Spalten) angenommen.

Vektoren oder Matrizen expandieren



Sie können die Funktion Expandieren \ dazu verwenden, um Leerstellen oder Nullen in einen Vektor oder in eine Matrix einzufügen. Das linke Argument muß ein Vektor sein, bestehend aus Einsen oder Nullen. Die Zahl 1 im linken Argument entspricht dem Element im rechten Argument. Die Nullen im linken Argument geben an, daß Leerstellen oder Nullen im Ergebnis eingefügt werden sollen. Leerstellen werden in einem Textvektor oder einer Matrix und Nullen in einem numerischen Vektor oder einer numerischen Matrix eingefügt. Das hört sich schwerer an als es ist!

Beispiele

```
      1 1 0 1 0 1\1 2 3 4
1 2 0 3 0 4
      1 1 0 1 0 1\'ABCD'
AB C D
```

Vektor wird um Nullen erweitert.

Vektor wird um Leerstellen erweitert.

Beim Expandieren einer Matrix werden Zeilen oder Spalten von Leerstellen oder Nullen eingefügt. Wie bei der Funktion Komprimieren müssen die Koordinaten (Zeilen oder Spalten) als Indexwerte [I] angegeben werden. Der Indexwert 1 bestimmt die erste Koordinate (Zeilen) und die 2 die zweite Koordinate (Spalten).

Beispiele

```
      B←2 2⍴⍳4
      B
1 2
3 4
      1 0 1\[2]B
1 0 2
3 0 4
      1 1 0\[1]B
1 2
3 4
0 0
```

Das linke Argument enthält eine 1 für jede Zeile oder Spalte.

Einfügen von Spalten.

Einfügen von Zeilen.

```
      1 0 1\B
1 0 2
3 0 4
      C←2 2⍴'ABCD'
      C
AB
CD
      1 0 1\C
A B
C D
```

Wird kein Indexwert angegeben, so wird die letzte Koordinate (Spalten) angenommen.

Aufgaben: Die Funktionen Komprimieren und Expandieren sollen verwendet werden

1. Weisen Sie einem Vektor mit dem Namen SALDEN die Salden von fünf Konten zu: 56 103 100 13 0. Wählen Sie alle SALDEN mit DM 100 oder mehr aus.

2. Bilden Sie eine Matrix WERTE←3 3⍴⍳9. Dann fügen Sie zwischen der ersten und der zweiten Zeile eine Zeile mit Nullen ein, und füllen Sie diese neue Zeile mit den Werten 20, 21 und 22.

Mögliche Lösungen

Aufgabe 1:

```
      SALDEN←56 103 100 13 0
      (SALDEN≥100)/SALDEN
103 100
```

Aufgabe 2:

```
      WERTE←3 3⍴⍳9
      WERTE
1 2 3
4 5 6
7 8 9
      WERTE←1 0 1 1\[1]WERTE
      WERTE
1 2 3
0 0 0
4 5 6
7 8 9
      WERTE[2;]←20 21 22
 1  2  3
20 21 22
 4  5  6
 7  8  9
```

Zwei oder mehrere Variablen verketten



Die Funktion Verketten ist dazu da, um zwei oder mehrere Vektoren zu einem Vektor zu verbinden. Das Komma steht zwischen dem linken und dem rechten Argument. Die Anzahl der Elemente im Ergebnisvektor ist gleich der Summe der Elemente der zwei Vektoren, die verbunden (verkettet) worden sind.

Beispiele

```
      A←1 2 3
      B←4 5 6
      A,B
1 2 3 4 5 6
      B,A
4 5 6 1 2 3
      C←'VER'
      D←'BIN'
      E←'DUNG'
      C,D,E
VERBINDUNG
```

```
      A,C
DOMAIN ERROR
      A,C
      ^
```

Vektoren mit numerischen Daten können nicht mit Vektoren aus Textdaten verkettet werden.

Zwei Matrizen können ebenfalls mit der Funktion Verketten verbunden werden. Um das zu bewirken, können wir den Indexwert [I] angeben. Der Indexwert 1 steht wieder für die erste Koordinate (Zeilen) und 2 für die zweite Koordinate (Spalten).

Beispiele

```
      A←2 2⍴⍳4
      B←2 2⍴4+⍳4
      A
1 2
3 4
      B
5 6
7 8
      A,[2]B
1 2 5 6
3 4 7 8
      A,B
1 2 5 6
3 4 7 8
      A,[1]B
1 2
3 4
5 6
7 8
```

Es wurden zwei Spalten mit zwei weiteren Spalten verbunden, die Anzahl der Spalten wurde damit erweitert.

Wenn keine Koordinate angegeben wird, so wird die letzte Koordinate (Spalten) angenommen.

In diesem Beispiel wurden zwei Zeilen an zwei vorhandenen Zeilen angehängt; die Anzahl der Zeilen wurde damit erweitert.

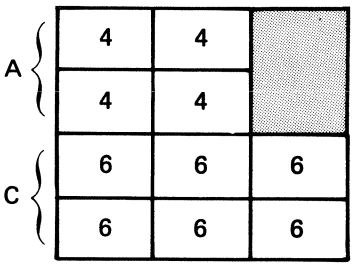
Wenn zwei Matrizen verkettet werden, müssen die Argumente übereinstimmen, d. h. es muß die Länge der Spalten bzw. die Länge der Zeilen gleich sein.

Beispiele

```
      A←2 2⍴4
      A
4 4
4 4
      C←2 3⍴6
      C
6 6 6
6 6 6
```

```
      A,[1]C
LENGTH ERROR
      A,[1] C
      ^
```

Der Längenfehler erscheint, wenn die Zeilen in A und C verschieden lang sind.




```
      A,[2]C
4 4 6 6 6
4 4 6 6 6
```

Das geht, weil die Anzahl der Zeilen in beiden Argumenten gleich ist. Die Länge der Zeilen spielt hier keine Rolle.

Bilden eines Ergebnisvektors mit der Funktion Verketten

Es gibt in der Praxis immer wieder Aufgaben, wo es nötig ist, neue Ergebnisse an einen bestehenden Vektor anzuhängen. Eine Möglichkeit hierzu wird jetzt beschrieben. Ein neues Ergebnis ist in der Variablen PLUS gespeichert. Vorangegangene Ergebnisse befinden sich im Vektor mit dem Namen ERGEBNIS. Sie können folgende Anweisung schreiben:

```
ERGEBNIS←ERGEBNIS,PLUS
```

Achtung: Wird diese Anweisung zum erstenmal ausgeführt, kann es sein, daß diesem ERGEBNIS noch kein Wert zugewiesen wurde. Für diesen Fall sollte vorher die Anweisung:

```
ERGEBNIS←⍳0
```

eingefügt werden: Durch diese Anweisung wird ERGEBNIS zu einem leeren Vektor.

Beispiele

```
      PLUS←10+5

      ERGEBNIS←ERGEBNIS,PLUS
VALUE ERROR
      ERGEBNIS←ERGEBNIS,PLUS
                ^

      ERGEBNIS←⍳0
      ERGEBNIS

      ERGEBNIS←ERGEBNIS,PLUS
      ERGEBNIS
15
      PLUS←15+10
      ERGEBNIS←ERGEBNIS,PLUS
      ERGEBNIS
15 25
```

ERGEBNIS hat noch keinen Wert und ist deshalb noch keine Variable.

Eine Variable mit dem Namen ERGEBNIS wird gebildet (leerer Vektor).

Eine Leerzeile wird angezeigt.

Jetzt kann ERGEBNIS verwendet werden.

Aufgabe: Verwendung der Funktion Verketten

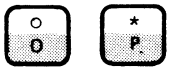
Bilden Sie bitte folgende Variablen: A←'I', B←'S', C←'A', D←'T', E←'R', F←'CH', G←'D', H←'G' und I←' '. Sehen Sie, welche Nachricht auf dem Bildschirm erscheint, wenn Sie die Variablen in folgender Reihenfolge verketten:

G C B I A B D I E A F D A H

Mögliche Lösung:

```
      A←'I'
      B←'S'
      C←'A'
      D←'T'
      E←'R'
      F←'CH'
      G←'D'
      H←'G'
      I←' '
      G,C,B,I,A,B,D,I,E,A,F,D,A,H
DAS IST RICHTIG
```

Logarithmieren



Sie können die Funktion Logarithmieren dazu verwenden, um log des rechten Arguments zur Basis, welche im linken Argument angegeben wurde, zu finden. Der log einer Zahl B zur Basis A ist der Exponent, um aus der Grundzahl A das Produkt B zu erhalten.

Beispiele

```
      A←2
      B←A*3
      B
8
      A⍟B  ◄——— log von B zur Basis A.
3
```

Aufgabe: Anwendung der Funktion Logarithmieren

1. Was ist der log von 256 zur Basis 2?

2. Mit welchen Exponenten muß 10 potenziert werden, um das Ergebnis 100000 zu erhalten?

Mögliche Lösungen

Aufgabe 1:

```
      2⊛256
8
```

Aufgabe 2:

```
      10⊛100000
5
```

Kapitel 5 Elemente eines Vektors aufsummieren (Reduzieren)

Es ist häufig von Nutzen, alle Elemente eines Vektors aufsummieren zu können. APL hat hierfür ein sehr einfaches Verfahren zur Verfügung. Diese Funktion nennen wir Reduzieren, weil ein numerischer Vektor auf eine einzelne Zahl reduziert wird, eben das Produkt. Das Funktionszeichen für reduzieren ist /. Das linke Argument ist zum Beispiel das + - Zeichen, das rechte Argument ist der Vektor.

Sie haben sicher schon bemerkt, daß das Funktionszeichen für Reduzieren das gleiche ist, wie für die Funktion Komprimieren. Den Unterschied erkennt das APL-System durch das linke Argument. Bei einer Funktion Komprimieren ist das linke Argument ein Vektor, der aus Einsen und Nullen besteht. Bei der Reduktion ist das linke Argument eine APL-Elementarfunktion.

Plus-Reduktion (Aufsummieren)

Beispiel

```
      A←1 2 3 4 5
      +/A
15
      1+2+3+4+5
15
```

← Addition aller Elemente von A, ist das gleiche wie +/A.

Verwendung der Plus-Reduktion zur Ermittlung eines Durchschnitts

Die Funktion Reduzieren ist sehr praktisch, um den Durchschnitt aller Elemente eines Vektors zu ermitteln. Der Vektor X hat folgenden Inhalt.

```
X←2 4 3 3 2.5 2
```

Verwendung der Plus-Reduktion für das Aufsummieren zweier Vektoren

Die Variable PREIS enthält Preise für verschiedene Artikel und die Variablen M1 und M2 sind zwei Vektoren, die von je zwei Kunden bestellte Mengen dieser Artikel enthalten. Der Gesamtrechnungsbetrag für Kunde 1 ist die Summe der Produkte von PREIS mal M1, und die Gesamtrechnung für Kunde 2 ist die Summe des Produktes von PREIS mal M2.

Beispiele

```
      PREIS←.66 1.40 27.10 2.39 14.00 7.60 8.45 2.80
      M1←0 0 2 1 0 0 0 0
      M2←12 7 0 5 0 0 0 10
      +/M1×PREIS
56.59
      +/M2×PREIS
57.67
```

Minus-Reduktion (alternierende Summe)

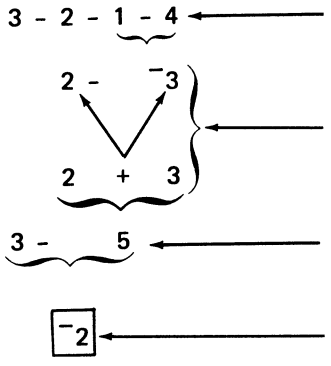
Beispiele

```
      A←3 2 1 4
      -/A
¯2
      3-2-1-4 ←——
¯2
```

—/A zeigt das gleiche Ergebnis wie diese Anweisung.

Im folgenden wird erklärt, warum die Antwort ¯2 ist.

Es wird immer von rechts nach links abgearbeitet.



Die erste Operation (subtrahiere 4 von 1; das Ergebnis ist ¯3).

Die zweite Operation (subtrahiere ¯3 von 2; das Ergebnis ist 5).

Die dritte Operation (subtrahiere 5 von 3; das Endergebnis ist ¯2).

Ergebnis

Maximum-Reduktion: Ermitteln des größten Wertes eines Vektors

Um den größten Einzelwert aus einem Vektor zu selektieren, können Sie die Funktionen Maximieren ⌈ und Reduzieren einsetzen.

Beispiele

```
      SALDO←62.15 127 4.42 18.65
      ⌈/SALDO
127
```

Vektor mit einer Reihe Kundensalden.

Größter Saldo.

Minimum-Reduktion: Ermitteln des kleinsten Wertes eines Vektors

Um den kleinsten Wert aus einem Vektor herauszufinden, können die beiden Funktionen Minimieren und Reduzieren verwendet werden.

Beispiel

```
      NUMMER←1 16 4 7 ¯9
      ⌊/NUMMER
¯9
```

Was sie tun können	Bezeichnung der Funktion	Tasten
• Sortieren eines Vektors in absteigender Folge	Sortierindex (fallend)	∇ G 1 M

```
      ⍒A←3 7 2 9 1
4 2 1 3 5
      A[⍒A]
9 7 3 2 1
```

Sortierindex von A in fallender Reihenfolge

Sortierung von A unter Verwendung des Sortierindexes

• Bilden einer Zufallszahl	Zufallszahl bilden	? Q

```
    ?6
3
```

Das Ergebnis kann eine Zahl zwischen 1 und 6 sein

• Bilden einer fortlaufenden Zahlenreihe	Indexvektor bilden	⍳ I

```
    ⍳5
1 2 3 4 5
```

Ein Vektor mit 5 fortlaufenden Zahlen wird gebildet.

• Bestimmen der Länge eines Vektors oder der Anzahl Zeilen und Spalten einer Tabelle	Dimension zeigen	⍴ R

```
    ⍴A
5
    ⍴MATRIX←2 3⍴⍳6
2 3
    MATRIX
1 2 3
4 5 6
```

Länge eines Vektors A

Bilden einer Matrix und gleichzeitig die Dimension anzeigen (Anzahl Zeilen und Spalten)

Die Funktion Strukturieren (siehe Kapitel 2)

Funktion Dimension zeigen

Was Sie tun können	Bezeichnung der Funktion	Tasten

APL-Funktionen, die zwei Argumente verlangen (weitere Informationen finden Sie in Kapitel 4).

Das Ergebnis der folgenden 6 Funktionen ist 1, wenn die Bedingung der APL-Vergleichsfunktion zutrifft, sonst ist das Ergebnis 0.

- Prüfen, ob beide Werte gleich sind — Prüfen auf gleich — 

```
      3 3=3 3
1
```

- Prüfen, ob das linke Argument größer als das rechte ist — Prüfen auf größer — 

```
      1 6>7
1
```

- Prüfen, ob das linke Argument kleiner als das Rechtsargument ist — Prüfen auf kleiner — 

```
      3<4
1
```

- Prüfen, ob das linke Argument größer oder gleich dem Rechtsargument ist — Prüfen auf größer/gleich — 

```
      1 2≥1 1  1 2
1 1
```

- Prüfen, ob das linke Argument kleiner oder gleich dem rechten Argument ist — Prüfen auf kleiner/gleich — 

```
      6≤6  9
1 1
```

- Prüfen, ob beide Werte ungleich sind — Prüfen auf ungleich — 

```
      7≠7 7  7
1 0
```

Was Sie tun können	Bezeichnung der Funktion	Tasten

Die beiden folgenden logischen Funktionen werden zum Prüfen der Ergebnisse aus Vergleichsoperationen verwendet. Logische Funktionen können als Argumente nur Einsen oder Nullen haben. Das Ergebnis 1 kommt, wenn die Bedingung, die geprüft wird, zutrifft, sonst ist das Ergebnis 0.

- Prüfen, ob beide Bedingungen wahr sind

  Mit 'Und' verknüpfen




```
      1∧1  0
1  0
```

- Prüfen, ob eine von beiden Bedingungen wahr ist

  Mit 'Oder' verknüpfen




```
      1∨1  0
1  1
```

- Auswählen der größeren von zwei Zahlen

  Maximieren




```
      5⌈4
5
```

- Auswählen der kleineren von zwei Zahlen

  Minimieren




```
      5⌊4
4
```

- Bestimmen des Indexes eines gegebenen Wertes in einem Vektor.

  Index zeigen




```
      9 8 7 6 5⍳7
3
```

Das rechte Argument ist in der dritten Position des linken Arguments gefunden worden. Das linke Argument ist immer ein Vektor.

- Auswahl mehrerer unterschiedlicher Zufallszahlen

  Stichproben nehmen




```
      3?6
2  3  1
```

Als Ergebnis können irgendwelche unterschiedlichen Zufallszahlen zwischen 1 und 6 vorkommen.

Was Sie tun können	Bezeichnung der Funktion	Tasten
• Selektieren bestimmter Elemente aus einem Vektor oder einer Matrix	Komprimieren	

```
      1 0 0 1/2 3 4 5
2 5
```

Auswählen von Elementen, die eine eins im linken Argument haben

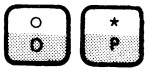
• Erweitern eines Vektors oder einer Matrix	Expandieren	

```
      1 0 1 0 1 0 1 0\2 3 4 5
2 0 3 0 4 0 5 0
```

Einfügen von Elementen in Abhängigkeit von Nullen im linken Argument

• Verbinden von Variablen	Verketten	

```
      'VER','KET','TEN'
VERKETTEN
```

• Bestimmen von log einer Zahl	Logarithmieren	

```
      2⍟8
3
```

log von 8 zur Basis 2

APL-Funktionen, die bisher noch nicht besprochen wurden (im IBM 5100 APL-Handbuch, IBM Form A12-2152, finden Sie hierzu weitere Erläuterungen)

• Kehre das Vorzeichen einer Zahl um	Vorzeichen umkehren	

```
      -3 ¯4
¯3 4
```

• Bestimmen des Vorzeichens einer Zahl	Vorzeichen prüfen	

```
      ×¯2 0 2
¯1 0 1
```

Das Ergebnis ist ¯1, wenn eine Zahl negativ ist, 0 wenn eine Zahl 0 ist und 1, wenn eine Zahl positiv ist.

• Den Reziprokwert einer Zahl ermitteln	Kehrwert bilden	

```
      ÷3
0.33333
```

Was Sie tun können	Bezeichnung der Funktion	Tasten

• e (2.71828) hoch 3
`*1 3`
`2.7183 20.086`
	Potenzieren zur Basis e	

• Bestimmen des log einer Zahl zur Basis e
`⊛2.7183 20.086`
`1 3`
	Logarithmieren zur Basis e	

• Eine Zahl wird mit Pi multipliziert
`○1 3`
`3.1416 9.4248`
	Mit ∏ multiplizieren	

• Das Produkt aller Zahlen zwischen 1 und n (einer angegebenen Zahl) `!4` `24` ← Das Ergebnis ist gleich 1x2x3x4	Fakultät bilden	

• Vertauschen: Aus 1 wird 0, und aus 0 wird 1
`~1 0`
`0 1`
	Boole'sch Negieren	

• Bestimmen, ob beide Bedingungen falsch sind `1⍲1 0` `0 1` ← Das Ergebnis ist 0, wenn beide Argumente 1 sind, sonst ist das Ergebnis 1	Mit 'nand' verknüpfen	
• Zeigen, ob eine von zwei Bedingungen falsch ist `1⍱1 0` `0 0` ← Das Ergebnis ist 1, wenn beide Argumente 0 sind, sonst ist das Ergebnis 0.	Mit 'nor' verknüpfen	

Was Sie tun können	Bezeichnung der Funktion	Tasten

- Umwandlung eines Skalars oder einer Matrix in einen Vektor

Aufreihen

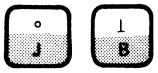


```
      MATRIX←2 3⍴⍳6
      MATRIX
1 2 3
4 5 6
      ,MATRIX
1 2 3 4 5 6
```

Die Matrix wurde in einen Vektor umgebildet

- In einem Textvektor enthaltene APL-Funktion soll ausgeführt werden

Aktivieren



```
      ⍎'2×4'
8
```

- Numerische Daten werden in Textdaten verwandelt

Deaktivieren



```
      A←⍕24
      A
24
      ⍴A
2
```

Über die Funktion Formatisieren, welche zwei Argumente erfordert, finden Sie im IBM 5100 APL-Handbuch nähere Einzelheiten (IBM Form A12-2152)

Dies sind Textzeichen

A ist ein 2-Elemente-(Text-)Vektor

- Die Vorzeichen unterdrücken

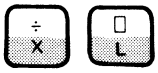
Absolutbetrag bilden



```
      |53 ¯46
53 46
```

- Invertieren einer quadratischen Matrix

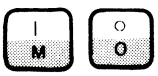
Invertieren



```
      ⌹A←2 2⍴1 3 5 7
¯0.875    0.375
 0.625   ¯0.125
```

- Herumdrehen der Elemente eines Vektors oder einer Matrix

Parallel spiegeln



```
      ⌽'REGEN'
NEGER
```

Was Sie tun können	Bezeichnung der Funktion	Tasten
• Den Rest einer Division festhalten	Rest bilden	

```
      3|8
2
```

2 ist der Rest aus der Division 8 durch 3

• Bestimmen der Werte für die trigonometrische Funktion eines Winkels	Kreisfunktion bilden	

```
      3○B←○÷4
1
```

Das linke Argument spezifiziert die trigonometrische Funktion (in unserem Beispiel: tan)

Das Ergebnis ist tan von 45° ($\pi \div$ Radius 4)

• Die Anzahl der Möglichkeiten der Kombinationen von Elementen, wie im linken Argument angegeben, aus einer Gesamtheit des linken Arguments bestimmen.	Binomialkoeffizienten bilden	

```
      2!4
6
```

Die Anzahl von 6 verschiedenen Kombinationen ist möglich, wobei die Reihenfolge keine Rolle spielt

• Herausfinden, wenn ein bestimmter Wert (linkes Argument) in einem Vektor oder einer Matrix vorhanden ist	Existenz prüfen	

```
      'ABC'∈'BANANA'
1 1 0
```

Das Ergebnis ist 1, wenn der Wert (oder das Zeichen) des linken Arguments im rechten Argument vorkommt, sonst ist das Ergebnis 0.

• Einen Wert in ein anderes Zahlensystem übertragen	Entschlüsseln	

```
      24 60 60⊥1 30 15
5415
```

Übertrage eine Std., 30 Min, 15 Sek. in Gesamtsek.

• Darstellung eines Wertes im angegebenen Zahlensystem	Verschlüsseln	

```
      24 60 60⊤5415
1 30 15
```

Darstellung von 5415 Sekunden in Stunden, Minuten und Sekunden.

Was Sie tun können	Bezeichnung der Funktion	Tasten
• Lösen eines oder mehrerer konstanter Vektoren von linearen Gleichungssystemen mit Koeffizientenmatrizen	Lineares Gleichungssystem lösen	÷ X ⌹ L

```
      26 9⌹B←2 2⍴3 5 1 2
7 1
```

• Entnehmen einer angegebenen Anzahl von Elementen aus einem Vektor oder einer Matrix	Entnehmen	↑ Y

```
      3↑A←1 2 3 4 5
1 2 3 ◄──────────────
```

Diese 3 Elemente sind dem Vektor entnommen worden

• Entfernen einer angegebenen Anzahl von Elementen aus einem Vektor oder einer Matrix	Entfernen	↓ U

```
      3↓A
4 5 ◄──────────────
```

Das Ergebnis ist der Rest der verbleibenden Elemente nach dem Entfernen der angegebenen ersten 3 Elemente.

• Erweitern einer Variablen um eine zusätzliche Dimension	Schichten	; ,

```
      1 2 3 4 5,[.5]6 7 8 9 0
1 2 3 4 5 ◄──────────────
6 7 8 9 0
```

2 Vektoren sind zu einer Matrix umgeformt worden

• Elemente eines Vektors oder einer Matrix um soviel Stellen rotieren lassen, wie im linken Argument angegeben sind	Rotieren	| M ○ O

```
      2⌽1 2 3 4 5
3 4 5 1 2 ◄──────────────
```

Rotieren des Vektors um 2 Positionen

Was Sie tun können	Bezeichnung der Funktion	Tasten
• Bilden von Datenbereichen mit mehr als 2 Dimensionen	Strukturieren	⍴ R

```
      ARRAY←2 3 3 ⍴⍳18
      ARRAY
 1  2  3
 4  5  6
 7  8  9

10 11 12
13 14 15
16 17 18
      ARRAY[2;3;1]
16
```

Jede Zahl im linken Argument ist eine Koordinate, in diesem Beispiel 3

Die letzte Koordinate sind Spalten

Die vorletzte Koordinate gibt die Zeilen an

Die erste Koordinate gibt die Anzahl der Matrizenebenen an

2 Matrizen (2 Ebenen)

Sie können aus einer 3- oder mehrdimensionalen Matrix ein einzelnes Element ansprechen, indem Sie ein Semikolon zwischen jede Koordinate setzen.

• Die Koordinaten werden von hinten nach vorne angegeben (aus Spalten werden mehrere Matrizen, aus Anzahl der Matrizen werden Spalten)	Transponieren oder diagonal spiegeln	⍉ O \ /

```
      ⍉ARRAY
1 10
4 13
7 16

2 11
5 14
8 17

3 12
6 15
9 18
```

Wird nur das Rechtsargument angegeben, so wird diagonal gespiegelt

Werden zwei Argumente angegeben, so besagt das linke Argument, wie die Koordinaten vertauscht werden sollen.

Was Sie tun können	Bezeichnung der Funktion	Tasten

APL-Operatoren

Der APL-Operator setzt sich aus zwei oder drei Elementarfunktionen zusammen. Das Ergebnis ist wieder eine Funktion. Das Symbol für eine derartige Funktion nennen wir Operator. Den Operator für die Reduktion haben wir bereits in Kapitel 5 kennengelernt.

- Verwenden der gleichen Funktionen für alle Elemente eines Vektors

Reduzieren 

```
      +/1 2 3 4 5
15
      ⌈/32 6 77 19 2
77
      ⌊/32 6 77 19 2
2
```

Die Summe aller Elemente

Das größte Element

Das kleinste Element

- Verwenden der gleichen Funktionen für alle Elemente eines Vektors (das Ergebnis jeder Operation wird in der nächsten wieder verwendet)

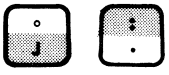
Aufstufen 

```
      +\1 2 3 4
1 3 6 10

      1
1
      1+2
3
      1+2+3
6
      1+2+3+4
10
```

Die Funktion Aufstufen arbeitet genauso, als wenn wir diese einzelnen Anweisungen durchführen würden.

- Aufbereiten von Arbeitstabellen für die verschiedensten APL-Funktionen oder Daten

Äußeres Produkt bilden 

```
      A←1 2 3 4
      A ∘.×A
1  2  3  4
2  4  6  8
3  6  9 12
4  8 12 16
```

Eine Multiplikationstabelle der Zahlen von 1 bis 4

Was Sie tun können	Bezeichnung der Funktion	Tasten
• Bilden des Matrixproduktes von zwei Matrizen	Inneres Produkt bilden	

```
      A←2 2⍴1 2 3 4
      B←2 2⍴ 5 6 7 8
      A+.×B
19 22
43 50
```

Das Matrixprodukt der Matrizen A und B

Kapitel 7 Funktionen definieren

Was ist eine definierte Funktion?

APL hat eine große Anzahl von Elementarfunktionen, doch es ist oft erforderlich, den Kreis der vorhandenen Funktionen um *eigene* Funktionen zu erweitern. Damit ist es erst möglich, die eigentlichen Aufgaben der Praxis zu lösen. Diese von Ihnen selbst erstellten Funktionen nennen wir definierte Funktionen. Nachdem diese gespeichert wurden, können Sie diese immer wieder anwenden.

Wie wird eine Funktion definiert?



Sie verwenden dazu vorhandene APL-Funktionen. Aus mehreren APL-Funktionen erstellen Sie eine eigene Funktion. Jede neue Funktion, die wir auch als Programm bezeichnen, erfordert:

- Eine Kopfzeile, bestehend aus einem Namen dieser Funktion (Funktionsname und weiteren Informationen, auf die später in diesem Kapitel näher eingegangen wird).

- Eine Anweisung oder eine Reihe von Anweisungen, also Programmzeilen, welche die Operationen definieren, die ausgeführt werden sollen.

Wenn APL-Anweisungen ausgeführt werden, befindet sich die IBM 5100 in der Arbeitsweise „Ausführung". Bevor eine neue Funktion definiert werden soll, muß diese Arbeitsweise gewechselt werden und zwar zur Arbeitsweise „Funktion definieren". Das ∇ (Nabla)-Zeichen wird verwendet, um von der einen zur anderen Arbeitsweise wechseln zu können. Um vom Ausführungsmodus in den Funktionsmodus zu wechseln, wird ∇ als erstes Zeichen vor die Kopfzeile gesetzt. Wenn die Funktion vollständig definiert wurde, wird wieder ein ∇ -Zeichen gesetzt, und wir befinden uns wieder im Ausführungsmodus. Wenn Sie wieder zum Ausführungsmodus zurückgekehrt sind, können Sie Ihre soeben erstellte eigene Funktion ausführen.

Wir zeigen eine definierte Funktion, mit der die Hypotenuse eines rechtwinkligen Dreiecks ermittelt werden kann. Die Anweisung hierfür kann so geschrieben werden: ((A * 2) + B * 2)) * 5, wobei die linken zwei Seiten von A und B quadriert werden und aus der Summe die Wurzel gezogen wird. Das Ergebnis ist die Länge der Hypotenuse. Diese Funktion muß einen Namen bekommen, um später diese Funktion wieder aufrufen zu können. Wir nennen diese Funktion HYP. Jetzt tippen wir ∇ ein (die IBM 5100 befindet sich nun im Definitionsmodus) und geben die Kopfzeile wie folgt ein:

```
      ∇HYP←A HYP B
[1]
```

Kopfzeile

APL antwortet mit der Nummer für die erste Anweisung, die eingegeben werden kann (erste Programmzeile).

Nach Eingabe einer Programmzeile erscheint die nächste Zeilennummer. Jetzt geben Sie bitte die folgende Programmzeile ein:

```
HP←((A*2)+(B*2))*.5
∇
```

Anweisung (Programmzeile)

∇-Zeichen: Wechselt zurück zum Ausführungsmodus.

Mit Ausnahme des Funktionsnamen HYP müssen alle anderen Angaben aus der Kopfzeile in den Programmzeilen verwendet werden, das gilt insbesondere für den Variablennamen HP, der das Ergebnis aufnehmen soll.

Auf dem Bildschirm steht jetzt folgendes:

```
      ∇HP←A HYP B
[1] HP←((A*2)+(B*2))*.5
[2] ∇
```

Wenn Sie einen Fehler bei der Programmeingabe gemacht haben, ist das nicht schlimm. Einige Seiten später werden wir Ihnen zeigen, wie man Programmfehler beseitigt. Während der Programmeingabe dürfen Sie auf keinen Fall die Aufwärtstaste und die Abwärtstaste betätigen. Nach Eingabe des ∇-Zeichens wurde Ihre Funktion HYP im Arbeitsspeicher gespeichert; Sie können diese Funktion wie jede APL-Funktion mit zwei Argumenten verwenden.

Beispiele

```
      3 HYP 4
5
      X←6
      Y←8
      X HYP Y
10
      R←3 6
      L←4 8
      R HYP L
5 10
```

Längen von zwei Seiten.

Länge der Hypotenuse.

Wie in jeder APL-Funktion können die Argumente auch Variablennamen sein, denen vorher ein Wert zugewiesen wurde.

Wann immer Sie HYP verwenden wollen: geben Sie die Argumente ein, die Sie brauchen. Das Symbol für die Errechnung einer Hypotenuse eines rechtwinkligen Dreiecks ist HYP, es ist gleichrangig wie das Symbol für die Addition +.

Eine definierte Funktion kann aus einer Anweisung bestehen, wie HYP, oder aus einer Folge von vielen Anweisungen.

Beispiel

```
      ∇HP←A HYPL B
[1] A2←A*2
[2] B2←B*2
[3] S←A2+B2
[4] HP←S*.5∇
      3 HYPL 4
5
```

Die Funktion HYP kann auch so definiert werden.

Das ∇-Zeichen kann auch am Ende der letzten Programmzeile stehen.

Das gleiche Ergebnis wie bei HYP.

Aufgaben: Verwendung von definierten Funktionen

1. Definieren Sie eine Funktion, mit der Sie die Summe von zwei beliebigen Zahlen bilden können. Führen Sie die Funktion aus.

2. Definieren Sie eine Funktion, mit der Sie den Inhalt eines Rechtecks mit beliebigen Seitenlängen berechnen können. Anschließend sollten Sie diese Funktion benutzen.

Mögliche Lösungen

Aufgabe 1:

```
      ∇S←M UND N
[1] S←M+N∇
      6 UND 3
9
```

Aufgabe 2:

```
      ∇A←LAENGE MAL BREITE
[1] A←LAENGE×BREITE
[2] ∇
      4 MAL 5
20
```

Testen Ihrer Funktion, bevor sie angewandt wird

Nachdem Sie eine Funktion definiert haben, sollten Sie diese mit Zahlen ausprobieren, die sich leicht nachrechnen lassen, um sicherzustellen, daß die Funktion richtig arbeitet. Nehmen wir an, Sie haben in der folgenden Funktion HYP diesen Fehler gemacht:

```
((A*2)+(B*2))×.5
```

× anstelle von *

Sie erhalten eine Antwort, aber nicht das richtige Ergebnis für die Hypotenuse eines rechtwinkligen Dreiecks. Wenn Sie ihre Funktion testen, ist folgendes möglich:

- Die IBM 5100 zeigt ein Ergebnis, daß **Sie akz**eptieren.
- Die IBM 5100 zeigt eine Fehlernachricht.
- Die IBM 5100 zeigt ein Ergebnis, aber es wird von Ihnen nicht akzeptiert.
- Nichts geschieht.

Wenn die IBM 5100 ein Ergebnis zeigt, das Sie akzeptieren

Großartig! Ihre Funktion arbeitet einwandfrei.

Auch dann, wenn das erste Ergebnis richtig erscheint, sollten Sie mehrere Tests mit extremen Werten durchführen, um sicherzustellen, daß Ihre Funktion allen Anforderungen genügt.

Wenn die IBM 5100 eine Fehlernachricht anzeigt

Sie können im IBM 5100 APL-Handbuch, IBM Form A12-2152, nachschauen, was es mit dieser Fehlernachricht auf sich hat und was Sie tun können, um den Fehler zu beseitigen.

Eine Fehlerbedingung führt immer zu einem Stop. Siehe Kapitel 8, dort finden Sie weitere Hinweise, was Sie tun können, wenn eine Funktion in der Ausführung unterbrochen wurde.

Wenn die IBM 5100 ein Ergebnis zeigt, das Sie nicht akzeptieren, oder wenn nichts geschieht.

In beiden Fällen haben Sie zwei Alternativen:

- Wir lassen uns die Funktion anzeigen. Das Anzeigen einer Funktion wird noch in diesem Kapitel besprochen.
- Einsetzen von Test- und Stopeinrichtungen (wird als nächstes besprochen), um bei der Fehlersuche zu helfen.

Wenn eine von Ihnen definierte Funktion ausgeführt wird und nichts geschieht (Sie erkennen es daran: Weder der Positionsanzeiger bewegt sich, noch ein Ergebnis erscheint auf dem Bildschirm) oder ein Ergebnis wird laufend wiederholt – dann befindet sich Ihre Funktion in einer Endlosschleife. In diesem Falle drücken Sie die Taste Taste ATTN (Unterbrechungstaste), die Funktion wird unterbrochen. In Kapitel 8 folgen Hinweise darüber, was Sie tun können.

Wie werden Test- und Stopeinrichtungen eingesetzt



Die Testeinrichtung ermöglicht es Ihnen, die Ausführung Ihrer Funktion Schritt für Schritt zu verfolgen. Das bedeutet, daß jedes Ergebnis jeder einzelnen Anweisung auf dem Bildschirm sichtbar gemacht wird. Sie können alle Anweisungen testen oder nur bestimmte Anweisungen einer Funktion. Um die Testeinrichtung einzusetzen, geben Sie ein: TΔ , den Namen der Funktion ←, und die Anweisungsnummern, die getestet und angezeigt protokolliert werden sollen.

Test TΔ

```
T△BEISPIEL←1 2 3 4 5 6
```

Die Nummern der Anweisungen, die getestet werden sollen.

Der Name der Funktion, die getestet werden soll.

Sie können auch so eingeben:

```
T△BEISPIEL←⍳6
```

Bilden eines Vektors mit den Zahlen von 1 bis 6.

Stop SΔ

Mit dieser Einrichtung können Sie die Ausführung in der Funktion anhalten, noch bevor die angegebene Anweisungsnummer zur Ausführung kommt. Die Funktion ist vorübergehend unterbrochen (die Ausführungsunterbrechung wird in Kapitel 8 noch eingehend besprochen).

Nachdem die Funktion angehalten wurde, wird von der IBM 5100 die Nummer der Anweisung angezeigt, die als nächste ausgeführt wird. Um die Stopeinrichtung einsetzen zu können, geben Sie ein: SΔ, den Namen der Funktion ←, und die Anweisungsnummer, vor der die Funktion angehalten werden soll.

Beispiel

```
S△BEISPIEL←3 6
```

Anweisungsnummern, bleibt nach Anweisung 2 und 5 stehen.

Funktionsname.

Nachdem die Funktion in der Ausführung unterbrochen wurde, können Sie diese wieder starten, und zwar durch Eingabe von →□LC. □LC ist eine Systemvariable, die immer die als nächstes auszuführende Anweisungsnummer enthält. Im Kapitel 9 befinden sich noch weitere Einzelheiten über Systemvariablen, im IBM 5100 APL-Handbuch, IBM Form A12-2152, finden Sie eine vollständige Beschreibung der Systemvariablen □LC.

Wir wollen Test und Stop einsetzen, um Fehler zu finden.

Beispiel

```
      ∇HP←A HYPX B
[1] 'DIE HYPOTENUSE IST'
[2] A2←A*2
[3] B2←B*2
[4] S←A2+B2
[5] HP←S×.5∇
      3 HYPX 4
DIE HYPOTENUSE IST
12.5
```

Definieren Sie eine Funktion, um die Hypotenuse eines rechtwinkligen Dreiecks zu berechnen.

Diese Funktion ist falsch.

Testen Sie diese Funktion mit Werten, von denen Sie das richtige Ergebnis kennen. Das Ergebnis müßte 5 sein.

Anwendung der Testeinrichtung um die richtige Lösung zu finden.

```
      T∆HYPX←2 3 4 5
      3 HYPX 4
DIE HYPOTENUSE IST
HYPX[2] 9
HYPX[3] 16
HYPX[4] 25
HYPX[5] 12.5
12.5
```

Die IBM 5100 antwortet mit dem Namen der Funktion, der Anweisungsnummer und dem Ergebnis dieser Anweisung.

Das richtige Ergebnis wird in allen Anweisungen, mit Ausnahme der Anweisung Nummer 5, angezeigt.

```
      T∆HYPX←⍳0
```

Um das Protokollieren zu beenden (die Testeinrichtung auszuschalten), gibt man ein: ←⍳0 oder ←0.

```
      S∆HYPX←4 5
```

Anwendung der Stopeinrichtung, um die richtige Funktion zu definieren.

```
      3 HYPX 4
DIE HYPOTENUSE IST
HYPX[4]
      A2
9
      B2
16
```

Die IBM 5100 antwortet mit dem Funktionsnamen und der Anweisungsnummer, die als nächstes ausgeführt werden soll.

Wenn die Funktion unterbrochen wurde, können Sie die Variablen eingeben, um sich deren Inhalt anzeigen zu lassen.

```
      →□LC
HYPX[5]
      S
25
      →□LC
12.5
```

Fortsetzung der Ausführung durch Eingabe von →□LC.

Die Ausführung wird unterbrochen vor Durchführung der nächsten Anweisungsnummer, die bei der Stopeinrichtung angegeben wurde.

Alle Variablen zeigen das richtige Ergebnis, der Fehler muß in der Anweisung Nummer 5 sein.

```
      S∆HYPX←⍳0
```

Mit der Eingabe ⍳0 wird der Programmstop wieder unwirksam.

Wie man einen Fehler in einer Funktion beseitigt, sehen Sie jetzt.

Was tun, wenn beim Definieren ein Fehler unterläuft?

Wenn Sie während der Definition Ihrer Funktion einen Fehler gemacht haben und die Funktion ist bereits abgeschlossen, können Sie folgendes versuchen:

- Die Funktion vollständig anzeigen lassen.
- Hinzufügen einer oder mehrerer Anweisungen am Schluß einer Funktion.
- Auswechseln von Anweisungen.
- Einfügen einer oder mehrerer Anweisungen.
- Löschen einer Anweisung aus der Funktion.
- Eine bestimmte Anweisung oder die letzten Anweisungen anzeigen.
- Abändern einer einzelnen Anweisung.

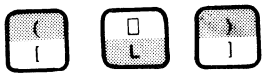
Wenn Sie während der Definition Ihrer Funktion einen Fehler feststellen, können Sie diesen Fehler sofort beseitigen! Sind Sie dagegen im Ausführungsmodus, so müssen Sie erst in den Definitionsmodus übergehen. Dies geschieht, in dem Sie ∇eingeben und dazu den Namen der Funktion. Geben Sie jedoch die gesamte Kopfzeile ein, so erscheint eine Fehlermeldung.

Wir definieren jetzt eine Funktion, die wir im Folgenden mehrfach verändern wollen. Geben Sie bitte folgende Funktion ein:

```
      ∇STAT X
[1] N←⍴X
[2] (+/X)÷N
[3] ⌊/X
[4] ⌈/X∇
```

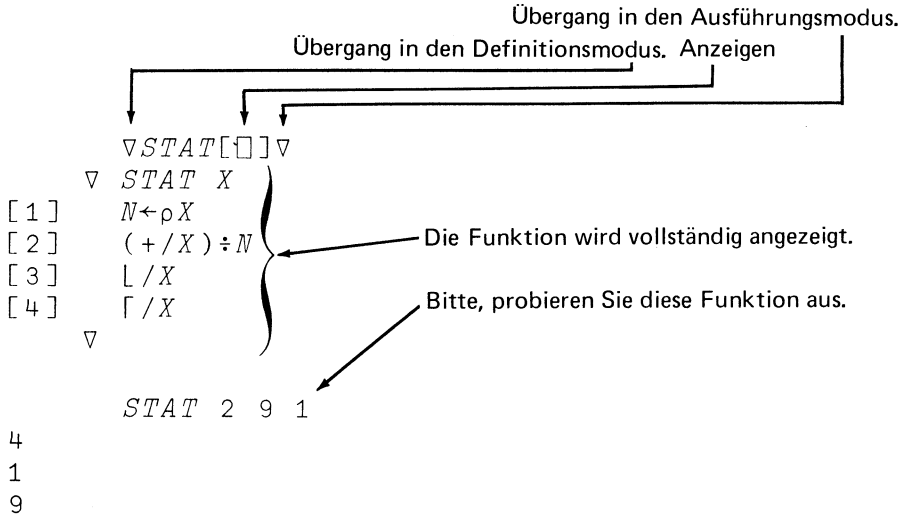
Diese Funktion errechnet den Durchschnitt und ermittelt die kleinste und die größte Zahl eines numerischen Vektors. Diese Funktion hat in der Kopfzeile keine Variablen für das Ergebnis, es werden alle einzelnen Ergebnisse auf dem Bildschirm angezeigt. Der Grund, warum eine Variable für das Ergebnis vorhanden sein sollte, wird später besprochen.

Anzeigen einer vollständigen Funktion



Um sich eine Funktion auf dem Bildschirm anzeigen zu lassen, geben Sie [□] hinter dem Funktionsnamen ein.

Beispiel



Hinzufügen einer oder mehrerer Anweisungen am Schluß einer Funktion

Um eine Anweisung am Schluß einer Funktion einzufügen, gehen wir in den Definitionsmodus. Die nächste freie Anweisungsnummer wird angezeigt. Jetzt können Sie beliebige Anweisungen hinzufügen.

Beispiele

```
      ∇STAT
[5]   (⌈/X)-⌊/X∇
```

Die IBM 5100 zeigt die nächste freie Anweisungsnummer an.

Übergang in den Definitionsmodus.

Fügen Sie diese Anweisung hinzu (es wird die Differenz zwischen dem kleinsten und dem größten Element eines Vektors ermittelt); das ∇-Zeichen bewirkt das Verlassen des Definitionsmodus.

```
      ∇STAT[□]∇
    ∇ STAT X
[1]   N←ρX
[2]   (+/X)÷N
[3]   ⌊/X
[4]   ⌈/X
[5]   (⌈/X)-⌊/X
    ∇
```

Anzeigen der vollständigen Funktion.

Die Funktion wird angezeigt.

```
      STAT 9 2 1
4
1
9
8
```

Die ergänzte Funktion wird ausprobiert.

Auswechseln einer Anweisung

Um eine Anweisung auszutauschen, wird die betreffende Anweisungsnummer in eckige Klammern [ ] gesetzt.

Beispiele:

```
      ∇STAT[□]
    ∇ STAT X
[1]   N←ρX
[2]   (+/X)÷N
[3]   ⌊/X
[4]   ⌈/X
[5]   (⌈/X)-⌊/X
    ∇
```

Anzeigen einer Funktion und Verbleiben im Definitionsmodus.

Angezeigte Funktion

Die IBM 5100 zeigt die nächste freie Anweisungsnummer an.

Wir wollen Anweisungsnummer 2 durch eine andere Anweisung ersetzen, um einen Durchschnitt zu errechnen. Das ∇-Zeichen bewirkt den Übergang in den Ausführungsmodus.

```
[6] [2] (+/X)÷ρX∇
```

```
      ∇STAT[□]∇
    ∇ STAT X
[1]   N←ρX
[2]   (+/X)÷ρX
[3]   ⌊/X
[4]   ⌈/X
[5]   (⌈/X)-⌊/X
    ∇
```

Anzeigen der veränderten Funktion.

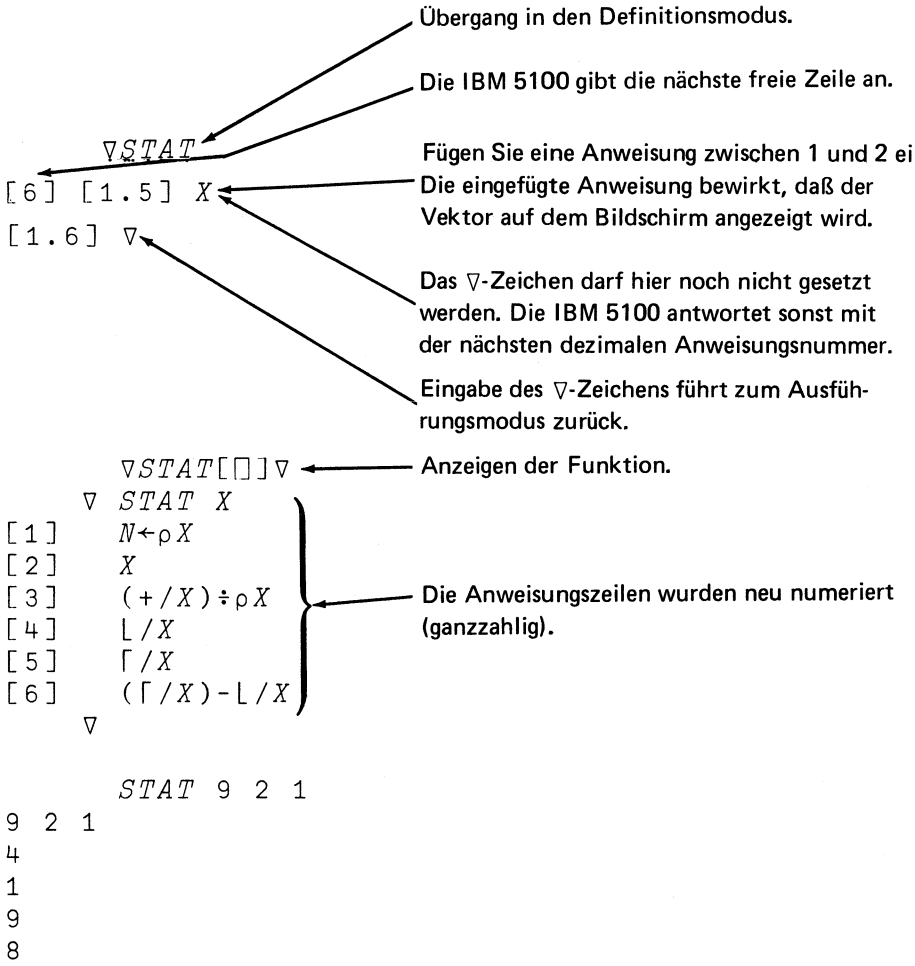
Angezeigte Funktion.

```
      STAT 9 1 2
4
1
9
8
```

Einfügen einer oder mehrerer Anweisungen

Um eine Anweisung in eine Funktion einzufügen, verwenden wir Anweisungsnummern, die zwischen den beiden Anweisungen liegen. Zwischen Anweisung 1 und Anweisung 2 soll zum Beispiel eine Anweisung eingefügt werden, die Anweisungsnummer könnte 1.5 oder irgend eine andere Dezimalzahl von 1.1 bis 1.9 sein.

Beispiele



Löschen einer Anweisung



Zum Löschen einer Anweisung innerhalb einer Funktion geben Sie [Δn] ein, wobei n die zu löschende Anweisungsnummer bedeutet.

Beispiele:

```
      ∇STAT[□]
   ∇ STAT X
[1]   N←ρX
[2]   X
[3]   (+/X)÷ρX
[4]   ⌊/X
[5]   ⌈/X
[6]   (⌈/X)-⌊/X
   ∇
[7] ⌊Δ4]
[5] ∇
```

Übergang in den Definitionsmodus und Anzeigen der gesamten Funktion.

Angezeigte Funktion.

Die IBM 5100 zeigt die nächste freie Anweisungsnummer an.

Löschen Sie Anweisung 4, die kleinste Zahl wird nicht mehr benötigt.

Achtung:
Das ∇-Zeichen darf nicht in der gleichen Zeile wie [Δn] stehen.

```
      ∇STAT[□]∇
   ∇ STAT X
[1]   N←ρX
[2]   X
[3]   (+/X)÷ρX
[4]   ⌈/X
[5]   (⌈/X)-⌊/X
   ∇
```

Anzeigen der modifizierten Funktion.

Angezeigte Funktion – die Zeile 4 ist gelöscht, und die Anweisungen sind neu numeriert worden.

```
      STAT 2 9 1
2 9 1
4
9
8
```

Eine bestimmte Anweisung oder die letzten Anweisungen anzeigen

Sie haben gesehen, wie eine vollständige Funktion angezeigt werden kann. Auch eine einzelne Anweisung kann angezeigt werden, oder ab einer bestimmten Anweisungsnummer können bis zum Ende der Funktion alle Anweisungen angezeigt werden. Um eine bestimmte Anweisung anzuzeigen, gibt man [n□] ein, wobei n die gewünschte Anweisungsnummer ist, die angezeigt werden soll. Um sich alle Anweisungen ab einer bestimmten Zeile an anzeigen zu lassen, geben Sie [□n] ein, wobei n die Anweisungsnummer bedeutet, ab welcher und bis zum Ende der Funktion die Anweisungen angezeigt werden sollen.

Beispiel

```
      ∇STAT[3□]∇
[3]   (+/X)÷ρX

      ∇STAT[□4]∇
[4]   ⌈/X
[5]   (/X)-⌊/X
```

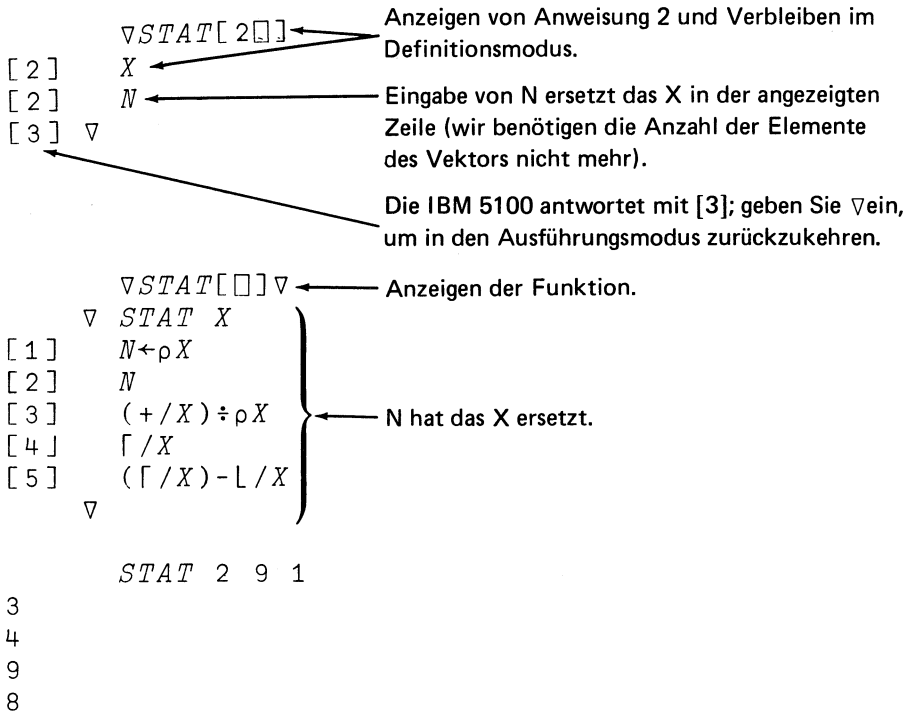
Anweisung 3 soll angezeigt werden.

Ab Anweisung 4 sollen alle Anweisungen bis zum Ende der Funktion angezeigt werden.

Abändern einer einzelnen Anweisung

Sie können Eintastfehler innerhalb einer Anweisung auf die gleiche Art und Weise beheben, wie Sie Eintastfehler während der Eingabe von Anweisungen im Ausführungsmodus berichtigen. Das gleiche Verfahren für Einfügen, Löschen und Auswechseln von Zeichen gilt auch hier. Um im Definitionsmodus innerhalb dieser Funktion Eintastfehler zu korrigieren, wird die Nummer einer fehlerhaften Anweisung eingegeben. Sie können auf keinen Fall die Aufwärts/Abwärtstasten verwenden, wenn Sie sich im Definitionsmodus befinden. Wenn Sie sich im Ausführungsmodus befinden und Sie wollen eine einzelne Anweisung abändern, so müssen Sie die abzuändernde Anweisung anzeigen lassen.

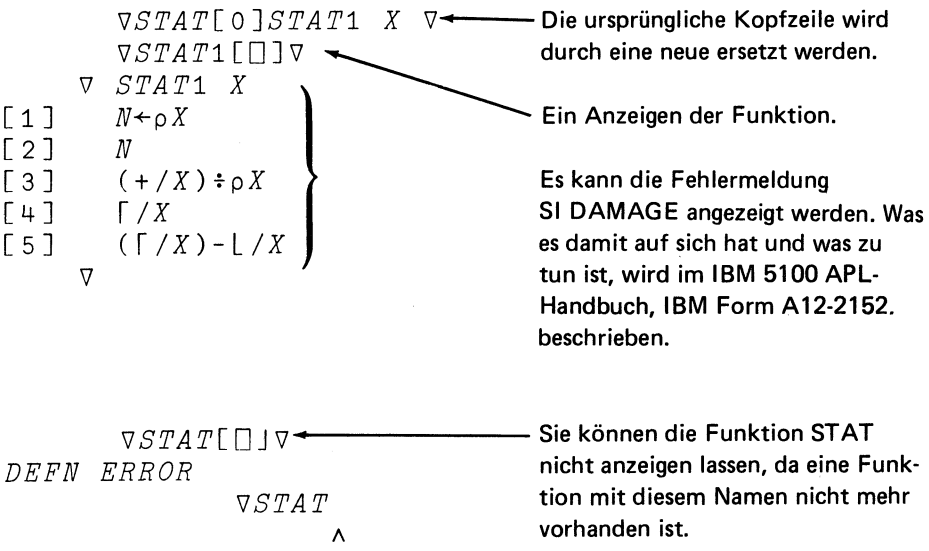
Beispiel



Ändern einer Kopfzeile

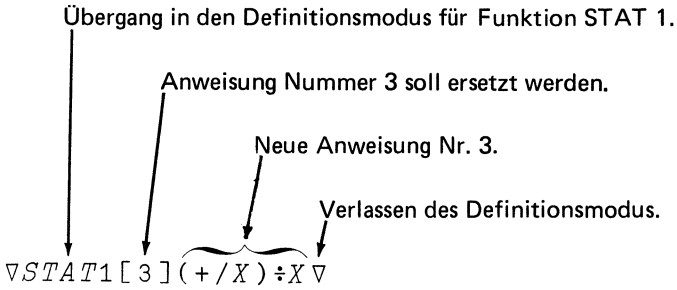
Sie können genau wie jede andere Anweisung auch die Kopfzeile ändern. Die Zeilennummer für die Kopfzeile ist die 0.

Beispiele



Schnell ist eine einzelne Anweisung eingefügt, ersetzt oder angefügt

Wenn Sie sich im Ausführungsmodus befinden, und Sie wollen nur eine einzelne Anweisung hinzufügen, einschieben oder ersetzen, so kann das in einer Zeile geschehen. Folgende Anweisung soll ersetzt werden:



Beispiele

```
      ∇STAT1[□]∇
    ∇ STAT1 X
[1]   N←ρX
[2]   N
[3]   (+/X)÷ρX
[4]   ⌈/X
[5]   (⌈/X)-⌊/X
    ∇
```

Anzeigen der Funktion STAT 1

```
      ∇STAT1[6]'ANWEISUNG WURDE ANGEFUEGT'∇
```

Hinzufügen einer Anweisung.

```
      STAT1 2 9 1
```

Die Funktion wird ausprobiert.

```
3
4
9
8
ANWEISUNG WURDE ANGEFUEGT
```

```
      ∇STAT1[3](+/X)÷N∇
      ∇STAT1⌊3.5]⌊/X∇
      ∇STAT1[□]∇
    ∇ STAT1 X
[1]   N←ρX
[2]   N
[3]   (+/X)÷N
[4]   ⌊/X
[5]   ⌈/X
[6]   (⌈/X)-⌊/X
⌈7]   'ANWEISUNG WURDE ANGEFUEGT'
    ∇
```

Eine Anweisung wird ausgewechselt.

Eine Anweisung wird eingefügt.

Anzeigen der veränderten Funktion.

```
     STAT1 2 9 1
3
4
1
9
8
ANWEISUNG WURDE ANGEFUEGT
```

Arten von Kopfzeilen

Wie bei den APL-Elementarfunktionen, so können bei den von Ihnen definierten Funktionen ein oder zwei Argumente vorkommen. Außerdem gibt es definierte Funktionen ohne Argumente. Die Anzahl der Argumente, die für eine Funktion benötigt werden, wird in der Kopfzeile festgelegt, zum Beispiel:

```
∇ERGEBNIS←ARGUMENT1 FUNKTIONSNAME ARGUMENT2
```

Diese Funktion erfordert zwei Argumente.

```
∇ERGEBNIS←FUNKTIONSNAME ARGUMENT
```

Diese Funktion erfordert ein Argument.

```
∇ERGEBNIS←FUNKTIONSNAME
```

Diese Funktion erfordert kein Argument.

Die Angaben links und rechts vom Funktionsnamen sind Variablennamen. Diese Variablennamen werden in der Funktion verwendet. In der Kopfzeile haben diese Variablen keine Werte, deshalb nennen wir sie Scheinvariablen. Die Variablennamen werden deshalb vor der Ausführung durch Werte ersetzt. Als Beispiel wollen wir uns folgende Funktion ansehen:

```
      ∇R←A GETEILT B
[1]  R←A÷B∇
```

Wenn Sie 10 GETEILT 2 eingeben, so steht der Wert 10 für A und der Wert 2 für B. Wenn die Anweisung A ÷ B ausgeführt wird, ist das Ergebnis 5.

Für definierte Funktionen gilt (genauso wie für einige Elementarfunktionen), daß die Argumente in der richtigen Reihenfolge eingegeben werden. Wenn Sie zum Beispiel 2 GETEILT 10 eingeben, ist die Antwort 0.2 anstatt 5.

Wird mit nur einem Argument gearbeitet, so muß dieses immer rechts vom Funktionsnamen stehen, sonst hält das System das Argument für den Funktionsnamen oder umgekehrt.

Beispiele

```
      ∇R←A MAL B
[1] R←A×B∇
      12 MAL 12
144

      ∇R←WURZEL X
[1] R←X*.5∇
      A←1 4 9 16 25 36
      WURZEL A
1 2 3 4 5 6

      ∇R←WUERFELN
[1] R←?6 6
[2] ∇
      WUERFELN
1 5
      WUERFELN
3 4
```

Zwei Argumente - diese Funktion ermittelt den Inhalt einer Fläche.

Ein Argument - diese Funktion errechnet die Wurzel aus einer Zahl.

Das Argument kann auch ein Vektor sein.

Kein Argument - diese Funktion simuliert das Spielen mit zwei Würfeln.

Das Ergebnis kann ein Paar von Zahlen zwischen 1 und 6 sein.

Warum gibt es Ergebnisvariablen?

In den meisten definierten Funktionen, die wir bisher gesehen haben, hatten wir eine Ergebnisvariable in der Kopfzeile benutzt. Die Ergebnisvariable ist ein Variablennamen, die das Ergebnis einer Funktion speichert, um es in einer anderen APL-Anweisung weiter zu verwenden. Funktionen mit einer Ergebnisvariablen bezeichnen wir als Funktionen mit einem expliziten Ergebnis. Ohne ein explizites Ergebnis kann Ihre Funktion nicht in einer zusammengesetzten APL-Anweisung eingesetzt werden. Funktionen ohne explizites Ergebnis können jedoch in einer Anweisung allein stehen.

Die folgende Funktion hat ein explizites Ergebnis.

```
                Ergebnisvariable
      ∇ERGEBNIS←MENGE PREIS1 BETRAG
[1] ERGEBNIS←BETRAG÷MENGE ∇
```

Ergebnisvariable (explizites Ergebnis)

Die Ergebnisvariable muß sowohl in der Kopfzeile als auch in der Anweisung vorhanden sein. Das Endergebnis muß in jedem Falle in der Ergebnisvariablen sein.

Beispiele

```
      ∇MENGE PREIS BETRAG
[1] BETRAG÷MENGE∇
      10 PREIS .60
0.06
```

Eine Funktion ohne explizites Ergebnis.

```
      ERGEBNIS←10 PREIS .60
0.06
VALUE ERROR
      ERGEBNIS←10 PREIS 0.6
      ∧
      10+10 PREIS .60
0.06
VALUE ERROR
      10+10 PREIS 0.6
        ∧
```

Das Ergebnis kann nicht in einer zusammengesetzten APL-Anweisung benutzt werden.

```
       ∇ERGEBNIS←M PREIS2 B
[1] ERGEBNIS←B÷M∇

          10 PREIS .60
    0.06
```

Eine Funktion mit einem expliziten Ergebnis.

```
      ERGEBNIS←10 PREIS2 .60
      ERGEBNIS
0.06
      10+10 PREIS2 .60
10.06
```

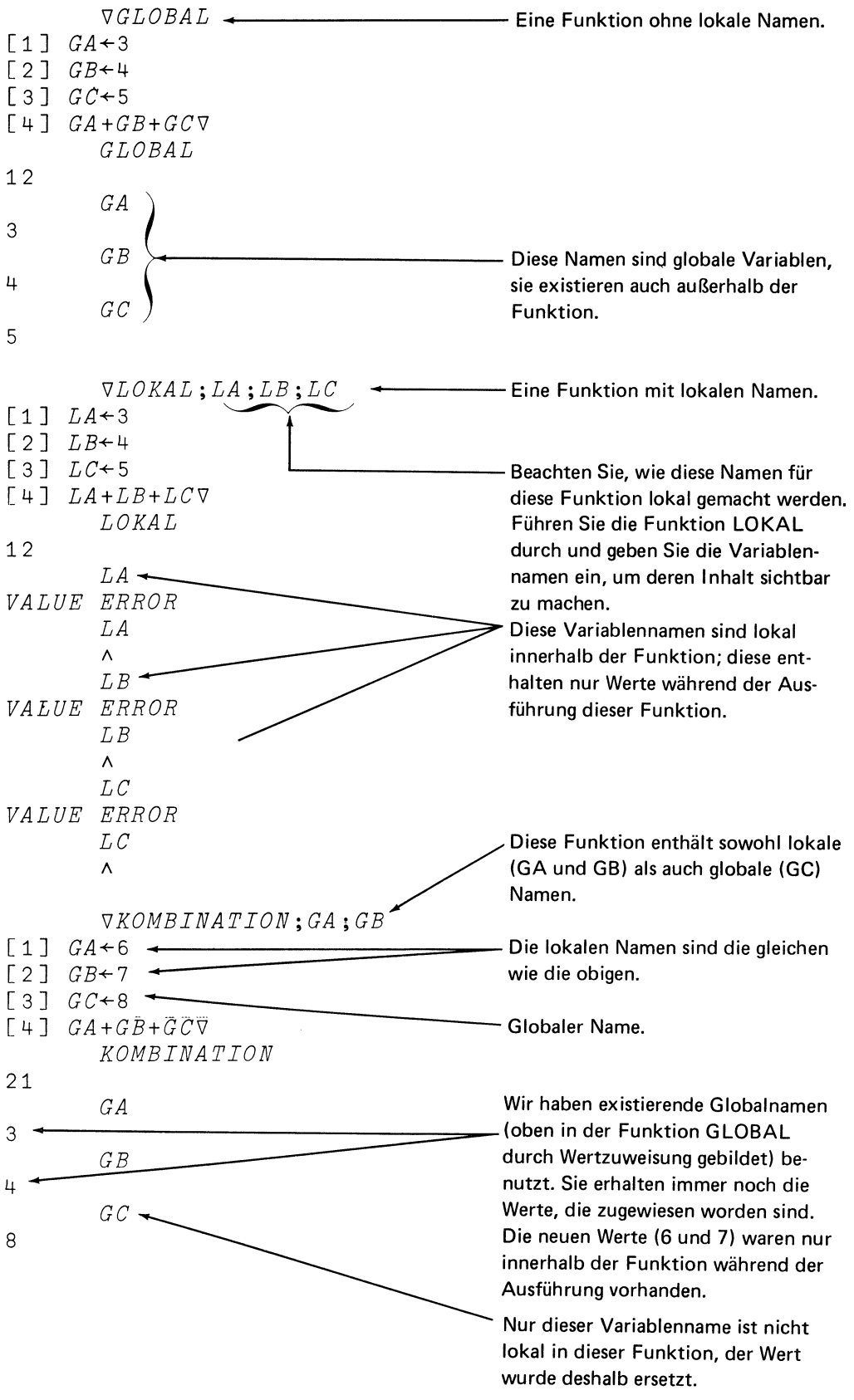
Das Ergebnis dieser Funktion kann jetzt in einer zusammengesetzten APL-Anweisung benutzt werden.

Bitte beachten Sie, wenn Sie eine Funktion in einer Anweisung mit anderen Operatoren verwenden wollen, daß diese Funktion ein explizites Ergebnis haben muß.

Lokale und globale Namen

Ein Name in einer definierten Funktion kann entweder lokal oder global sein. Ein Name für eine globale Variable hat während der Ausführung einer Funktion den gleichen Wert wie außerhalb dieser Funktion. Eine lokale Variable hat einen Wert nur in der Zeit, während diese Funktion ausgeführt wird. Jeder Name in der Kopfzeile (mit Ausnahme des Funktionsnamens) ist eine lokale Variable. Wie wir gesehen haben, kann eine Kopfzeile eine Ergebnisvariable und Argumente enthalten. Diese Variablennamen in der Kopfzeile sind nur in dieser Funktion lokale Variablen. Aber zusätzliche Variablennamen können als lokale Variablen in der Kopfzeile angegeben werden, und zwar hinter dem rechten Argument (sofern vorhanden). Diese zusätzlichen lokalen Variablen werden durch ein Semikolon getrennt. Die Kopfzeile ∇SCHLEIFE R;I;J besagt, daß das Rechtsargument R und die Variable I und J dieser Funktion lokal sind.

Beispiele


∇GLOBAL
[1] GA←3
[2] GB←4
[3] GC←5
[4] GA+GB+GC∇
GLOBAL
12
GA
3
GB
4
GC
5
Eine Funktion ohne lokale Namen.
Diese Namen sind globale Variablen, sie existieren auch außerhalb der Funktion.
∇LOKAL;LA;LB;LC
[1] LA←3
[2] LB←4
[3] LC←5
[4] LA+LB+LC∇
LOKAL
12
LA
VALUE ERROR
LA
∧
LB
VALUE ERROR
LB
∧
LC
VALUE ERROR
LC
∧
Eine Funktion mit lokalen Namen.
Beachten Sie, wie diese Namen für diese Funktion lokal gemacht werden. Führen Sie die Funktion LOKAL durch und geben Sie die Variablennamen ein, um deren Inhalt sichtbar zu machen.
Diese Variablennamen sind lokal innerhalb der Funktion; diese enthalten nur Werte während der Ausführung dieser Funktion.
∇KOMBINATION;GA;GB
[1] GA←6
[2] GB←7
[3] GC←8
[4] GA+GB+GC∇
KOMBINATION
21
GA
3
GB
4
GC
8
Diese Funktion enthält sowohl lokale (GA und GB) als auch globale (GC) Namen.
Die lokalen Namen sind die gleichen wie die obigen.
Globaler Name.
Wir haben existierende Globalnamen (oben in der Funktion GLOBAL durch Wertzuweisung gebildet) benutzt. Sie erhalten immer noch die Werte, die zugewiesen worden sind. Die neuen Werte (6 und 7) waren nur innerhalb der Funktion während der Ausführung vorhanden.
Nur dieser Variablenname ist nicht lokal in dieser Funktion, der Wert wurde deshalb ersetzt.

Sie werden sich jetzt fragen, warum Variablennamen für eine Funktion als lokal definiert werden. Hier einige Gründe, um lokale Variable einzusetzen:

- Wir unterstellen, daß eine definierte Funktion mit Namen ZAHL einen Variablennamen X enthält. Einige Zeit später weisen Sie X das Ergebnis einer wichtigen Berechnung zu. Wenn Sie jetzt ZAHL ausführen, passiert folgendes:

  1. Wenn X in der Funktion ZAHL als lokale Variable definiert war, bleibt der Wert X erhalten.

  2. Wenn X nicht als lokal angegeben wurde, ist der ursprüngliche Wert X (das Ergebnis Ihrer Berechnung) gelöscht und durch einen Wert aus der Berechnung von ZAHL ersetzt worden.

- Werte, die Sie außerhalb einer Funktion nicht benötigen, nehmen im Arbeitsbereich unnötigen Platz weg. Durch Verwendung von lokalen Variablen können Sie Platz sparen.

Verzweigen, Anweisungsnamen und Schleifen

Verzweigen und Anweisungsnamen

Anweisungen in einer definierten Funktion werden normalerweise in der Reihenfolge der Anweisungsnummern ausgeführt. Nach Durchführung der letzten Anweisung ist die Funktion zu Ende. Dieser normale Ablauf der Ausführung kann durch Verzweigen modifiziert werden, d. h., daß nicht die nächstfolgende, sondern eine andere Anweisung ausgeführt wird. Verzweigen kann man mit Hilfe des Pfeiles nach rechts → , gefolgt von einem Anweisungsnamen, der auf die Anweisung hindeutet, wo der Ablauf fortgesetzt werden soll.

Beispiel: Der Ausdruck →START bedeutet, daß zur Anweisung mit dem Namen START gesprungen werden soll. Der Anweisungsname wird durch einen Doppelpunkt von der eigentlichen Anweisung getrennt:

```
[2] START:N←N+1
      .
      .
      .
[5] →START
```

In der obigen Darstellung ist die Anweisung 2 mit dem Namen START versehen worden. In diesem Beispiel hat START den Wert 2. Wenn in dieser Funktion Anweisungen eingefügt werden, kann sich die Anweisungsnummer ändern. Wenn die geänderte Funktion ausgeführt wird, wird zu der Anweisung mit dem Namen START verzweigt, sobald die Anweisung 5 ausgeführt wurde.

Anweisungsnamen sind lokal für die Funktion, in der sie verwendet werden. Das bedeutet, daß Anweisungsnamen

- nicht als lokale Variablen in der Kopfzeile definiert werden müssen.
- Es können keine Werte zugewiesen werden.

Es gibt zwei verschiedene Arten von Verzweigungen: unbedingte und bedingte Verzweigungen:

- Unbedingte Verzweigungen sind Verzweigungen, die immer dann ausgeführt werden, wenn Anweisungszeilen mit dem Verzweigungszeichen durchlaufen werden. Im vorhergehenden Beispiel hatten wir es mit einer unbedingten Verzweigung zu tun, weil [5]→ START besagt, daß zur Anweisung mit dem Namen START verzweigt werden soll, sobald Anweisung Nummer 5 ausgeführt wurde. Ein anderes Beispiel für unbedingtes Verzweigen ist →0, was bewirkt, daß diese Funktion beendet wird.

- Bedingte Verzweigungen sind Verzweigungen, die erst dann ausgeführt werden, wenn eine Vergleichsoperation als Ergebnis eine bestimmte Bedingung festgestellt hat. Andernfalls wird die nächste Anweisung ausgeführt. Eine bedingte Verzweigung kann so aussehen:

  →(BEDINGUNG)/N

  Die Verzweigung zur Anweisung N wird durchgeführt, wenn die Bedingung wahr ist, sonst wird die nachfolgende Anweisung ausgeführt. Beispiel: In APL läuft die Verzweigungsanweisung →(I ⩾ N)/START folgendermaßen ab:

  1. Zuerst wird die Bedingung (I⩾ N) geprüft. Das Ergebnis ist eine 1, wenn die Bedingung zutrifft und 0, wenn die Bedingung nicht zutrifft.

  2. Das Ergebnis des Vergleichs wird dann als linkes Argument für die Funktion (/) Komprimieren verwendet:
     a. Ist das Ergebnis des Vergleichs eine 1, so wird START (rechtes Argument) ausgewählt und zur Anweisung mit dem Namen START verzweigt.
     b. Ist das Ergebnis des Vergleichs eine 0, so wird nichts vom rechten Argument wirksam (ein leerer Vektor ist das Ergebnis). Eine Verzweigung zu einem leeren Vektor ist gleichbedeutend mit dem Fortschreiten der Funktion in der nächsten Zeile.

Das folgende Beispiel zeigt zwei verschiedene Lösungswege für eine Funktion. Es soll eine Summe aus allen Zahlen von 1 bis zum Wert des Arguments gebildet werden, der in der Kopfzeile angegeben wird.

Beispiele

```
      ∇S←SUM2 N
[1]  S←0
[2]  I←1
[3]  VERGLEICH:→(I>N)/0
[4]  S←S+I
[5]  I←I+1
[6]  →VERGLEICH∇
      SUM2 5
15
```

Verzweige nach 0 (beende die Funktion) oder führe die nächste Funktion aus.

Unbedingte Verzweigung nach VERGLEICH.

```
      ∇S←SUM3 N
[1]  S←0
[2]  I←0
[3]  VERGLEICH:S←S+I
[4]  I←I+1
[5]  →(I≤N)/VERGLEICH∇
      SUM3 5
15
```

Verzweige nach VERGLEICH oder Ende der Funktion.

Schleifenbildung

Einen sich wiederholenden Teil einer Funktion bezeichnet man als Schleife. Wenn Sie eine Schleife in Ihrem Programm haben, müssen Sie sicherstellen, daß Sie aus dieser Schleife auch wieder herauskommen.

Beispiele

```
      ∇SCHLEIFE
[1] I←0
[2] ANAME:'DIES PROGRAMM ENTHAELT EINE SCHLEIFE'
[3] I←I+1
[4] →ANAME∇
      SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
```

Diese Funktion enthält eine Endlosschleife.

Um die Ausführung von SCHLEIFE zu unterbrechen, drücken Sie bitte die ATTN-Taste.

```
SCHLEIFE[3]
```

Der Name der Funktion und die Anweisungsnummer der unterbrochenen Funktion wird angezeigt.

```
      ∇SCHLEIFE[4]→(I≠3)/ANAME∇
      ∇SCHLEIFE[⍬]∇'
    ∇ SCHLEIFE
[1]   I←0
[2]  ANAME:'DIES PROGRAMM ENTHAELT EINE SCHLEIFE'
[3]   I←I+1
[4]   →(I≠3)/ANAME
    ∇
      SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
DIES PROGRAMM ENTHAELT EINE SCHLEIFE
```

Eine Möglichkeit, um aus einer Schleife herauszukommen.

Anzeigen der Funktion.

Die Schleife wird dreimal durchlaufen.

Wie werden Daten während der Ausführung einer Funktion eingegeben?



Bisher haben Sie Daten, die zur Verarbeitung benötigt wurden, in der Form eingegeben, daß Sie einer Variablen einen Wert zugewiesen haben (sei es außerhalb einer Funktion oder in einer Funktion selbst). Sie waren hierbei auf zwei Argumente beschränkt, außerdem war es wichtig, die Reihenfolge der Argumente zu beachten. Kurzum, es wäre sehr nützlich, wenn man Daten auch während der Ausführung einer Funktion eingeben könnte. Dies ist in APL tatsächlich möglich, und damit kann man einen Dialog innerhalb einer definierten Funktion durchführen. Wir können in einer Funktion eine Anweisung schreiben, die den Benutzer auffordert, bestimmte Daten einzugeben. Wenn in einer Funktion numerische Daten verlangt werden, so genügt es, in der Anweisung das Zeichen für Eingabe □ (Fenster) zu verwenden. Das □-Zeichen unterbricht den Ablauf einer Funktion, und es erscheint auf dem Bildschirm □:. Jetzt können numerische Daten eingegeben werden. In der Anweisung steht vor dem □-Zeichen das Zuweisungszeichen ←. Das bedeutet, daß der eingetastete Wert einer Variablen zugewiesen wird, D←□. Auch Textdaten können innerhalb einer Funktion eingegeben werden. In der Anweisung, die Textdaten anfordert, muß das Zeichen ⍞ verwendet werden (das Zeichen für das Fenster und Hochkomma übereinandergetippt). Daß eine Texteingabe verlangt wird, erkennen Sie daran, daß kein besonderes Zeichen auf dem Bildschirm erscheint und der Positionsanzeiger auf der Position 1 steht. Die Texteingabe muß nicht in Hochkommata eingeschlossen werden, weil durch ⍞ nur die Eingabe von Textdaten möglich ist.

Beispiel

Geben Sie bitte eine definierte Funktion ein, die folgendes bewirken soll: Zwei Buben (Achim und Volker) veranstalten ein Würfelspiel. Jeder hat abwechselnd einen Wurf mit sechs Würfeln. Gezählt werden alle Einsen, es werden 9 Runden gespielt, wer hat die meisten Einsen?

```
      ∇WUERFELSPIEL
[1]  'GEBE DEN NAMEN DEINES MITSPIELERS EIN'
[2]  MITSPIELER←⍞
[3]  'GEBE SEINE WUERFE EIN'
[4]  WMIT←□
[5]  'GEBE DEINEN NAMEN EIN'
[6]  ICH←⍞
[7)  'GEBE DEINE WUERFE EIN'
[8]  MEINEW←□
[9]  'DER ENDSTAND IST:'
[10] MITSPIELER
[11] +/WMIT
[12] ICH
[13] +/MEINEW∇
```

Die Eingabe auf der Tastatur weist den Variablen die eingetasteten Werte zu (entweder numerische □ oder Textdaten ⍞).

So oft hat jeder in den neun Versuchen die Eins gewürfelt:

Achim – 0 1 0 2 0 3 2 5 0

Volker – 0 0 0 2 3 1 3 0 0

Beispiele (Fortsetzung)

Jetzt wird die Funktion ausgeführt:

```
      WUERFELSPIEL
GEBE DEN NAMEN DEINES MITSPIELERS EIN
ACHIM
GEBE SEINE WUERFE EIN
□:
       0 1 0 2 0 3 2 5 0
GEBE DEINEN NAMEN EIN
VOLKER
GEBE DEINE WUERFE EIN
□:
       0 0 0 2 3 1 3 0 0
DER ENDSTAND IST:
ACHIM
13
VOLKER
9
```

Diese Anfrage erfordert eine Texteingabe.

Die Texteingabe braucht nicht in Hochkommata gesetzt zu werden, weil ⍞ in der Anweisung steht.

Das sind keine Textdaten, deshalb werden sie durch das □-Zeichen angefordert.

Beachten Sie bitte: Wenn □: auf dem Bildschirm erscheint, so wird eine numerische Eingabe verlangt. ←□ in einer Anweisung weist einer Variablen den Wert zu. Erscheint kein Zeichen auf dem Bildschirm und der Positionsanzeiger steht auf Position 1, so wird eine Texteingabe angefordert und durch ←⍞ in der Anweisung Textdaten einer Variablen zugewiesen.

Es kann sein, daß Sie, während Sie sich im Dialog mit dem Computer befinden, die Funktion vorzeitig beenden wollen. Normalerweise kann man die Ausführung einer Funktion unterbrechen, indem man die ATTN-Taste drückt. Wenn eine Funktion unterbrochen werden soll, die eine Eingabe von Daten verlangt, so **kann** man mit der Unterbrechungstaste (ATTN-Taste) die Funktion beenden. APL verfügt über andere Möglichkeiten, um eine Funktion zu unterbrechen, auch wenn eine Eingabe verlangt wird. Werden numerische Daten mit □:angefordert, so gibt man → ein. Die Ausführung der Funktion wird sofort unterbrochen.

Wurden Textdaten mit ⍞ angefordert, so muß das ⍬-Zeichen eingegeben werden. Um dieses Zeichen einzugeben, drückt man die CMD-Taste und gleichzeitig die [− / +]-Taste einmal und anschließend die EXECUTE-Taste. Dadurch wird die Ausführung der Funktion unterbrochen. Was Sie tun können, wenn eine Funktion unterbrochen wurde, sehen Sie im nächsten Kapitel 8.

Beispiele

```
      WUERFELSPIEL
GEBE DEN NAMEN DEINES MITSPIELERS EIN
MARTIN
GEBE SEINE WUERFE EIN
□:
      →
```

Laßt uns das WUERFEL-SPIEL ausführen, um zu sehen, wie wir die Funktion während der Anforderung einer Eingabe unterbrechen können.
Die Eingabe von → bewirkt, daß während der Eingabeanforderung durch □ die Funktion abgebrochen wird.

```
      WUERFELSPIEL
GEBE DEN NAMEN DEINES MITSPIELERS EIN
→
```

Eine Texteingabe wird verlangt [!]. Durch die Eingabe von → wird die Funktion nicht unterbrochen. Der Pfeil wird als Textzeichen gewertet und als Namen des Mitspielers angesehen. Die Funktion wird weiter ausgeführt.

```
GEBE SEINE WUERFE EIN
□:
      1 1 1 1 1 1 1 1 1
GEBE DEINEN NAMEN EIN
■
INTERRUPT
WUERFELSPIEL[6] ICH←[!]
                ^
```

Geben Sie jetzt einige Zahlen ein, um in der Funktion fortzufahren, bis ein Text durch [!] angefordert wird.

Geben Sie das ■-Zeichen ein (CMD-Taste drücken und festhalten und gleichzeitig die [− / +]-Taste drücken), die Funktion ist unterbrochen.

Kapitel 8 Was tun, wenn eine Funktion unterbrochen wurde?

Die Ausführung der von Ihnen definierten Funktion wird unterbrochen, wenn:

- Die ATTN-Taste (Unterbrechungstaste) gedrückt wurde.
- Die Stopeinrichtung (S△) verwendet wurde.
- Sich ein Fehler in der Funktion befindet.
- Das ⓤ-Zeichen (CMD-Taste und [- / +]) gedrückt wurde, um anstelle einer Texteingabe aus der Funktion auszusteigen.

Eine Funktion, die wie oben beschrieben unterbrochen wurde, bezeichnen wir als hängende Funktion. Die hängende Funktion ist nach wie vor aktiv. Ihre weitere Ausführung wird im Folgenden besprochen.

Wir wollen jetzt sehen, was Sie tun können, wenn Ihre Funktion in der Ausführung unterbrochen wurde.

Wenn die Unterbrechungstaste gedrückt wurde

Wenn Sie die Unterbrechungstaste (ATTN-Taste) während der Ausführung Ihrer Funktion gedrückt haben, so wird die Anweisung, die begonnen war, noch ausgeführt. Die Funktion ist jetzt unterbrochen. Die IBM 5100 zeigt auf dem Bildschirm den Funktionsnamen und die nächste auszuführende Anweisung an.

Nachdem die Funktion in ihrer Ausführung unterbrochen wurde, haben Sie folgende Möglichkeiten:

- Die Funktion abzuändern.
- Die Funktion noch einmal auszuführen.
- Eine andere definierte Funktion ausführen.
- Ausführen einer Systemanweisung mit Ausnahme von)SAVE,)COPY und)PCOPY. Die Systemanweisungen sind im IBM 5100 APL-Handbuch, IBM Form A12-2152, beschrieben.
- Die Funktion beenden durch Eingabe von →.

Grundsätzlich können Sie eine unterbrochene Funktion, herbeigeführt durch das Drücken der Unterbrechungstaste (ATTN-Taste), später wieder weiterführen. Sie brauchen nur folgendes einzugeben: →⎕LC. LC ist eine Systemvariable, die die als nächste auszuführende Anweisungsnummer enthält (siehe IBM 5100 APL-Handbuch, IBM Form A12-2152, dort finden Sie eine vollständige Beschreibung der ⎕LC Systemvariablen).

Sie können die Funktion auch an einer beliebigen Stelle fortsetzen. Sie brauchen nur →n einzugeben, wobei n die Anweisungsnummer darstellt, wo Sie die Funktion wieder aufsetzen wollen.

Beispiele

```
      ∇SFUNKTION;ZAEHLER
[1] ZAEHLER←0
[2] SCHLEIFE:'DIESE FUNKTION ENTHAELT EINE SCHLEIFE'
[3] ZAEHLER←ZAEHLER+1
[4] →SCHLEIFE
[5] 'DIESE FUNKTION WIEDERHOLT SICH'
[6] ZAEHLER
[7] 'MAL'∇
```

Definieren Sie eine Funktion mit einer Endlosschleife.

Drücken Sie die ATTN-Taste (Unterbrechungstaste), um diese Funktion anzuhalten.

```
      SFUNKTION
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE

SFUNKTION[3]
```

Die Funktion ist eine hängende Funktion. Die Anweisung, die als nächste ausgeführt werden sollte, wird auf dem Bildschirm – in eckigen Klammern [] – angezeigt.

```
      ∇SFUNKTION[4]→(ZAEHLER<3)/SCHLEIFE∇
      →⎕LC
```

Anweisung 4 wird dahingehend abgeändert, daß keine Endlosschleife mehr ausgeführt werden kann.

Setzen Sie die Ausführung der Funktion fort.

Beispiele (Fortsetzung)

```
DIESE FUNKTION WIEDERHOLT SICH
7 ←
MAL
```

Die Zahl, die Sie auf dem Bildschirm sehen, zeigt an, wie oft die Schleife durchlaufen wurde.

```
      SFUNKTION ←
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION ENTHAELT EINE SCHLEIFE
DIESE FUNKTION WIEDERHOLT SICH
3
MAL
```

Jetzt führen Sie diese Funktion noch einmal aus.

Wenn die Unterbrechungstaste (ATTN-Taste) zweimal während der Ausführung einer APL-Anweisung (innerhalb oder außerhalb einer definierten Funktion) gedrückt wird, wird die Ausführung in der Anweisung sofort unterbrochen. Die Nachricht INTERRUPT erscheint, die Anweisung, die sich gerade in der Ausführung befand, wird angezeigt, und das ∧-Zeichen gibt die Stelle an, an der die Anweisung unterbrochen wurde. Sie können diese Methode der Unterbrechung von Anweisungen dann verwenden, wenn eine Anweisung zu viel Zeit für die Ausführung benötigt. Nach der Unterbrechung einer Anweisung sind keine Ergebnisse aus dieser Anweisung vorhanden.

Wenn die Stopeinrichtung verwendet wurde

Sie sind mit der Stopeinrichtung aus Kapitel 7 bereits vertraut. Nachdem die Stopeinrichtung eine Funktion unterbrochen hat, können Sie wie folgt weiterfahren:

- Die Funktion abändern.
- Die Funktion nochmals ausführen.
- Eine andere definierte Funktion ausführen.
- Ausführung einer Systemanweisung mit Ausnahme von)SAVE,)COPY und)PCOPY.
- Die unterbrochene Funktion durch Eingabe von →□LC wieder anlaufen lassen.
- Die Funktion durch Eingabe von → beenden.

Wenn sich ein Fehler in der Funktion befindet

Im Gegensatz zu den beiden vorher beschriebenen Unterbrechungen, die Sie selbst auslösen können, haben Sie auf diese Art der Unterbrechung keinen Einfluß. Die IBM 5100 unterbricht die Funktion automatisch und bringt eine Fehlermeldung. Die Fehlermeldung und die Beseitigung von Fehlern wird ausführlich im IBM 5100 APL-Handbuch, IBM Form A12-2152, beschrieben.

Fehler in einer definierten Funktion sind manchmal schwer zu finden und zu beseitigen. Die Fehlermeldung, die auf dem Bildschirm erscheint, besagt, welche Anweisung in der Ausführung unterbrochen wurde und warum. Aber der Grund für einen Fehler kann durchaus in einer anderen Anweisung liegen. Auch wird nicht angezeigt, ob der Fehler durch einen Eingabefehler auftrat oder durch falsche Formulierung einer Anweisung. Hier einige Hinweise, die Ihnen beim Finden der Fehlerursache helfen sollen:

- Prüfen Sie die Anweisung auf Tippfehler.
- Analysieren Sie die Ausführung einer Anweisung von rechts nach links. Erinnern Sie sich, daß APL jeden Ausdruck von rechts nach links abarbeitet und nur durch Klammern einem Ausdruck der Vorrang gegeben werden kann.
- Verwenden Sie die Funktion Dimension zeigen ρ, um sicherzugehen, daß die Dimensionen der Argumente so sind, wie Sie es annahmen. Sie wollten beispielsweise zwei Vektoren verketten und stellen jetzt fest, daß eins der Argumente eine Matrix ist.
- Geben Sie die Namen von Variablen ein, um zu sehen, ob sie richtige Werte enthalten (lokale Variablen in einer hängenden Funktion können auch angezeigt werden, weil die Funktion noch aktiv ist).
- Zerlegen Sie eine Anweisung in mehrere kleine Teile. Die Aufwärts-  und die Abwärtstaste  helfen, um eine Anweisung schnell zu zerlegen.

  Sie führen eine Anweisung so aus, wie APL es auch tut (von rechts nach links, wobei Ausdrücke in Klammern vorher gelöst werden müssen). Geben Sie die erste Operation in APL-gerechter Form ein, und zwar für die Teile Ihrer Anweisung, für die Sie Teilergebnisse angezeigt haben möchten. Jetzt drücken Sie die Abwärtstaste dreimal und die Aufwärtstaste einmal. Damit haben Sie das Zwischenergebnis in den Eingabebereich gestellt. Nun können Sie die nächste Operation anschließen und ausführen. So können Sie nach und nach alle Operationen Ihrer Anweisung durchführen, bis Sie den Fehler gefunden haben.

Es ist zu empfehlen, daß Sie alle Zwischenergebnisse entweder von Hand aufschreiben oder über den Drucker IBM 5103 ausgeben. Sie können dann bei einer Fehlersuche alles besser zurückverfolgen. Außerdem ist solches Aufschreiben nützlich, wenn Sie den Eindruck haben, die IBM 5100 arbeitet nicht einwandfrei, um Ihrem zuständigen Techniker das Problem zu erklären.

Wenn eine Funktion wegen eines Fehlers unterbrochen wurde, haben Sie die gleichen Möglichkeiten fortzufahren, als wenn Sie die Unterbrechungstaste oder die Stopeinrichtung verwendet haben:

- Die Funktion abändern.
- Die Funktion noch einmal ausführen.
- Eine andere definierte Funktion ausführen.
- Ausführung einer Systemanweisung mit Ausnahme von)SAVE,)COPY und)PCOPY.
- Die unterbrochene Funktion durch Eingabe von →□LC wieder anlaufen lassen.
- Die Funktion durch Eingabe von → beenden.

Wenn das ⍎-Zeichen anstelle einer Texteingabe eingegeben wurde

In Kapitel 7 haben wir das ⍎-Zeichen verwendet, um anstelle einer Texteingabe die Ausführung der Funktion zu unterbrechen. In diesem Falle wird auf dem Bildschirm der IBM 5100 die Nachricht INTERRUPT angezeigt, dazu der Namen der Funktion und die Anweisung, in der die Texteingabe angefordert wurde. Nachdem die Funktion unterbrochen wurde, können Sie die gleichen Verfahren wie bei den anderen Unterbrechungsarten anwenden. In den meisten Fällen werden Sie jedoch durch Eingabe von → die Funktion beenden.

Anzeigen der hängenden Funktion

Der Statusanzeiger enthält alle Namen der hängenden Funktionen und die Nummer der Anweisungen, die als nächstes ausgeführt werden können. Wenn Sie)SI oder)SIV eingeben, erscheint der Inhalt des Statusanzeigers auf dem Bildschirm. Siehe auch IBM 5100 APL-Handbuch, IBM Form A12-2152, dort finden Sie weitere Informationen über den Statusanzeiger.

Die HOLD-Taste stoppt die Verarbeitung HOLD

Wir haben bisher gesehen, wie eine definierte Funktion unterbrochen werden kann. Diese bezeichnen wir als hängende Funktion. Sie können auch die Ausführung einer Funktion anhalten, indem Sie die HOLD-Taste einmal drücken. Um die Verarbeitung fortzusetzen, drückt man die HOLD-Taste ein zweites Mal. Die HOLD-Taste ist sehr nützlich, wenn Ergebnisse zu schnell über den Bildschirm laufen. Sie halten die Verarbeitung an, lesen irgendwelche Informationen und setzen anschließend die Verarbeitung fort.

Beispiele

```
      ∇HOLDF
[1] H←0
[2] 'DRUECKE DIE HOLD-TASTE'
[3] SCHLEIFE:H←H+1
[4] H
[5] →(H≠25)/SCHLEIFE∇
```

Definieren Sie diese Funktion.

```
      HOLDF
DRUECKE DIE HOLD-TASTE
1
2
3
.
.
.
```

Die Zahlen auf dem Bildschirm geben an, wie oft die Schleife durchlaufen worden ist, bevor die Verarbeitung angehalten wurde.

Jetzt drücken Sie die HOLD-Taste erneut, um die Verarbeitung fortzusetzen.

Wenn Ihre IBM 5100 nicht arbeitet und der Positionsanzeiger nicht auf dem Bildschirm blinkt, so wurde die HOLD-Taste einmal gedrückt, um die Verarbeitung anzuhalten.

Kapitel 9 Der Gebrauch von Bandkassetten

Bisher haben Sie ausschließlich mit dem aktiven Arbeitsbereich der IBM 5100 gearbeitet. Der aktive Arbeitsbereich ist ein Teil des internen Speichers der IBM 5100. In diesem werden Variablen und definierte Funktionen gespeichert. Wenn Sie den Netzschalter der IBM 5100 ausschalten oder WIEDERANLAUF drücken, sind alle Daten des aktiven Arbeitsbereichs verloren. Bevor Sie den Strom abschalten oder WIEDERANLAUF drücken, können Sie die Daten in Ihrem aktiven Arbeitsbereich sichern, und zwar durch Übertragen vom aktiven Arbeitsbereich zur Bandkassette. Diese Bandkassette ist wie eine Bibliothek. Sie übertragen Ihren aktiven Arbeitsbereich aufs Band (so, wie Sie ein Buch in eine Bibliothek stellen). Später nehmen Sie die gespeicherten Informationen vom Band und übertragen diese in den aktiven Arbeitsbereich (so, wie Sie ein Buch aus der Bibliothek herausholen und es wieder lesen).

Die Bandbibliothek kann eine Datei oder mehrere Dateien haben (jede Datei ist einem Buch vergleichbar); und wie jedes Buch in einer Bibliothek einen Namen hat, so hat auch jede Datei auf dem Band einen Namen (Name des Arbeitsbereichs).

Die IBM 5100-Systemanweisungen steuern den Datenverkehr zwischen aktivem Arbeitsbereich und Band (Bibliothek). Schauen Sie auf die Beschriftungen über der Schreibmaschinentastatur. Die Systemanweisungen werden durch Betätigung der CMD-Taste und einer Taste der oberen Reihe eingegeben. Auch die Eingabe über Schreibmaschinentastatur – Zeichen für Zeichen – ist erlaubt. Es ist wichtig, daß jede Systemanweisung mit dem)-Zeichen beginnt. Es gibt noch eine Reihe von Systemanweisungen, die nicht durch einen Tastendruck ausgeführt werden können. Alle diese Systemanweisungen sind ausführlich im IBM 5100 APL-Handbuch, IBM Form A12-2152, beschrieben.

In den folgenden Beispielen wollen wir Ihnen zeigen, wie einige Systemanweisungen arbeiten. Zuerst muß eine Bandkassette in die IBM 5100 eingeführt werden. Versichern Sie sich, daß das Band keine Daten enthält, die später noch gebraucht werden. Der Sicherheitsschalter darf nicht in Richtung SAFE zeigen (Abb. 2). Jetzt führen Sie die Bandkassette ein (Abb. 3).

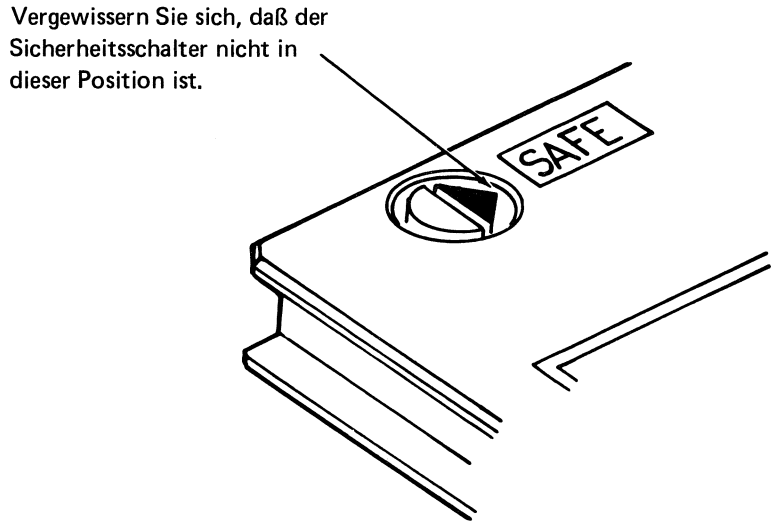


Abb. 2 Der Sicherheitsschalter (SAFE)



Abb. 3 Einführen einer Bandkassette in Ihre IBM 5100

Beispiele

Drücken Sie bitte zuerst an Ihrer IBM 5100 WIEDERANLAUF; alle Daten im aktiven Arbeitsbereich sind jetzt gelöscht.

```
CLEAR WS
```

Diese Nachricht wird angezeigt. Die IBM 5100 ist bereit, neue Daten aufzunehmen.

Geben Sie folgende Funktion ein und weisen Sie VARIABLE einen Text zu. Wir wollen später das Ganze aufs Band übertragen.

```
      ∇BEISPIEL;R;NAME
[1] 'DIESE FUNKTION ZAEHLT DIE BUCHSTABEN IHRES NAMENS'
[2] 'GEBEN SIE BITTE IHREN NAMEN EIN'
[3] NAME←⍞
[4] 'ES SIND'
[5] ⍴NAME
[6] 'BUCHSTABEN IN IHREM NAMEN'∇

      VARIABLE←'WIR WOLLEN DATEN SICHERN'
```

Jetzt probieren Sie die Funktion BEISPIEL einfach einmal aus, um zu sehen, daß sie funktioniert.

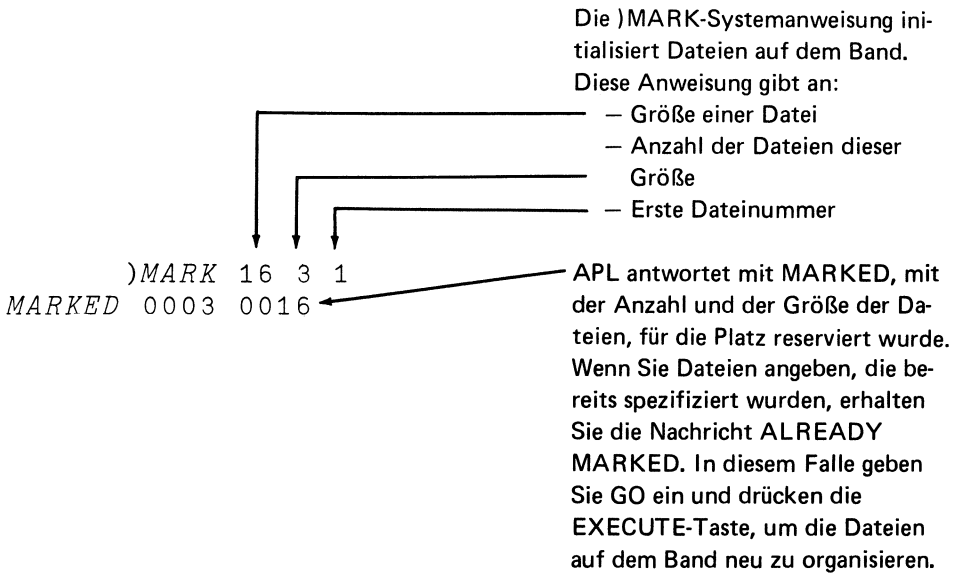
```
      )FNS
BEISPIEL
```

Die)FNS-Systemanweisung zeigt alle im aktiven Arbeitsbereich vorhandenen definierten Funktionen an.

```
      )VARS
VARIABLE
```

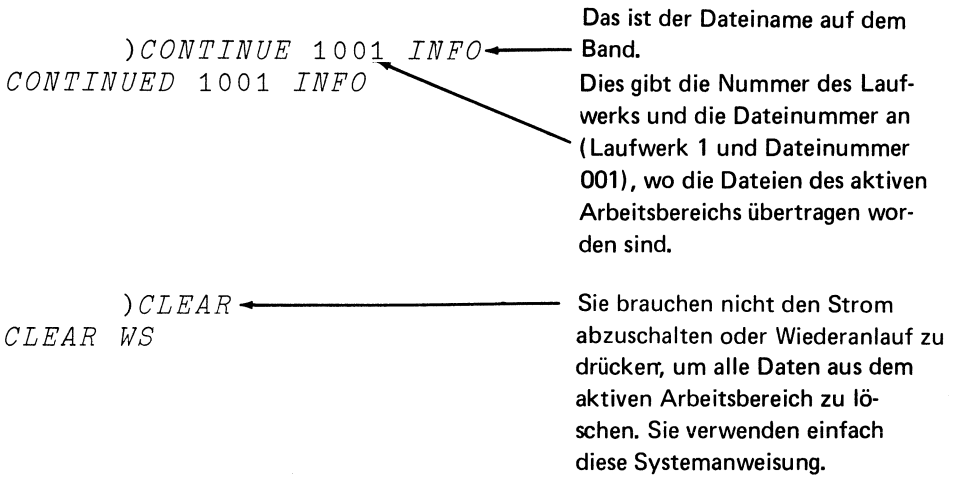
Die)VARS-Systemanweisung zeigt alle globalen Variablen an, die im aktiven Arbeitsbereich vorhanden sind.

Bevor ein Band benutzt werden kann, müssen Sie angeben, wieviele Dateien in welcher Größe und Reihenfolge Sie speichern möchten.

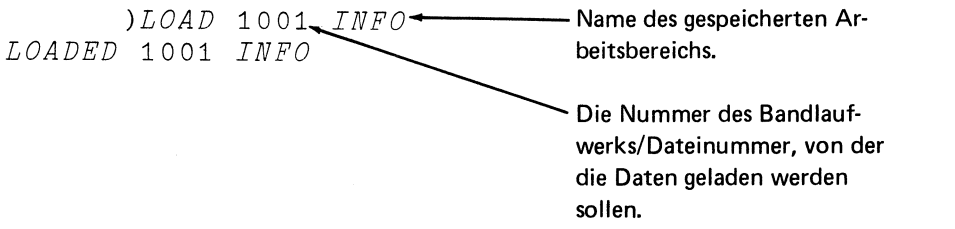


Die Dateien werden in Blöcken von 1024 Bytes eingeteilt. In unserem Beispiel haben wir 16 Blöcke, mit 1024 Bytes je Block, als Dateigröße bestimmt (siehe auch IBM 5100 APL Handbuch, IBM Form A12-2152, Angaben über Dateigrößen).

Jetzt wollen wir den Inhalt des aktiven Arbeitsbereichs aufs Band übertragen.



Die gespeicherten Daten auf dem Band können jederzeit in den aktiven Arbeitsbereich zurückgeholt werden.



```
      )FNS
BEISPIEL
      )VARS
VARIABLE
```

Jetzt sind die Daten, die auf dem Band gespeichert waren, wieder im aktiven Arbeitsbereich.

Alle Systemanweisungen sind im IBM 5100 APL-Handbuch, IBM Form A12-2152, ausführlich beschrieben. Probieren Sie ruhig die Systemanweisungen aus, um deren Arbeitsweise genau kennenzulernen.

Sie haben gesehen, wie man Daten vom aktiven Arbeitsbereich aufs Band überträgt. Sie können auch eine einzelne Variable aufs Band übertragen, diese Variablen stehen Ihnen jederzeit zur weiteren Verarbeitung zur Verfügung. Wie Sie das im Einzelnen durchführen können, sehen Sie im Kapitel 8, Bandein- und ausgabe und Druckausgabe, im IBM 5100 APL-Handbuch, IBM Form A12-2152.

Was sind Systemvariablen?

Systemvariablen sind Variablen, die innerhalb des aktiven Arbeitsbereichs das System steuern. Alle Systemvariablen beginnen mit dem⎕ -Zeichen. Sie werden festgelegt, wenn Sie einen leeren Arbeitsbereich (CLEAR WS) angezeigt bekommen haben. Im IBM 5100 APL-Handbuch finden Sie eine ausführliche Beschreibung aller Systemvariablen. In den folgenden Beispielen sehen Sie, wie einige Systemvariablen verändert werden können und wie ihre Auswirkungen auf die APL-Funktionen sind.

Beispiele

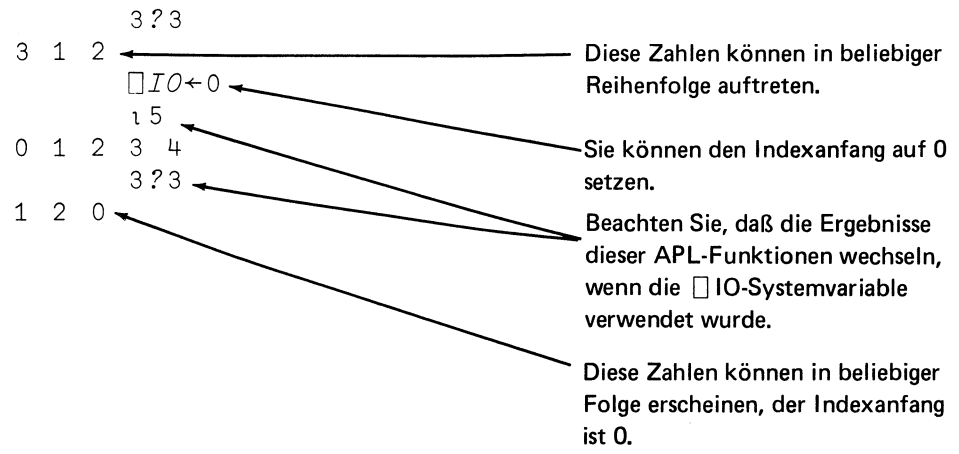
Die Systemvariable für den Indexanfang ist ⎕IO. Der Wert von ⎕IO kann 0 oder 1 sein. Das bedeutet, daß der Index für das erste Element eines Vektors oder einer Matrix sowohl 0 oder 1 sein kann, je nachdem, welchen Inhalt die Systemvariable ⎕IO hat. Die APL-Funktionen ⍳ ? ⍋ ⍒ sind davon betroffen.

```
      ⎕IO
1
      ⍳5
1 2 3 4 5
```

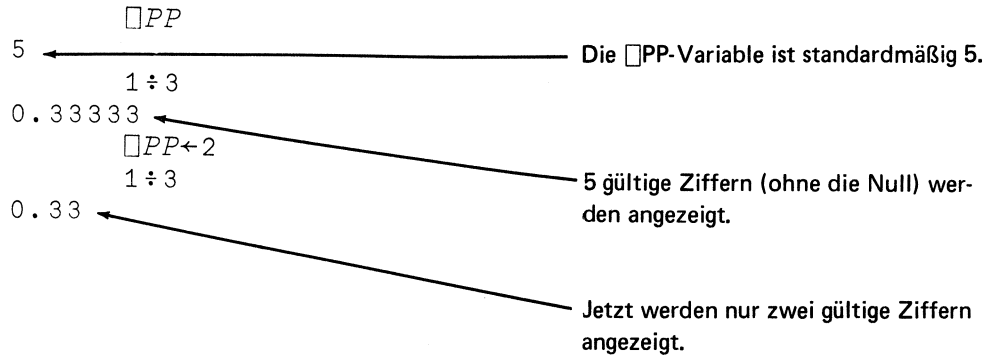
Sie können sich den Inhalt der Systemvariablen anzeigen lassen.

Die ⎕IO-Systemvariable ist im Augenblick 1.

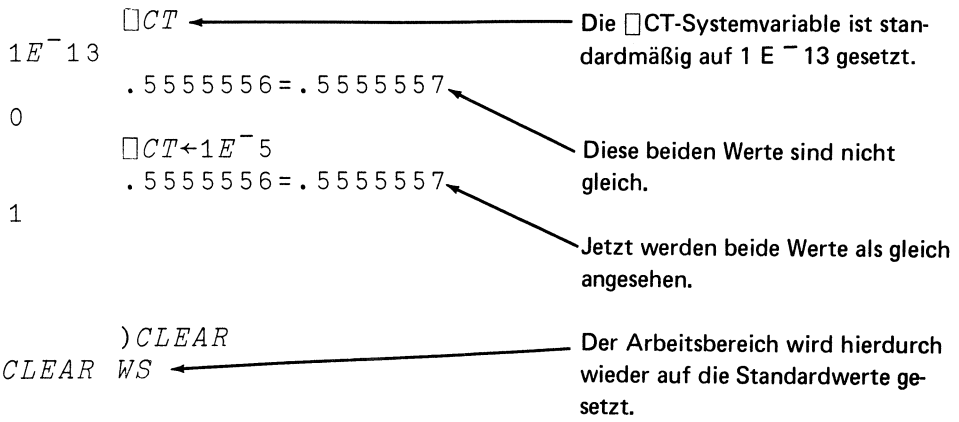
Das Ergebnis, wenn die ⎕IO-Variable auf 1 gesetzt ist.



Die Zahlenlänge (⎕PP-Systemvariable) gibt die Anzahl Stellen einer Zahl an, die auf dem Bildschirm erscheinen soll.

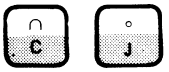
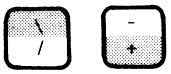
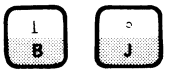
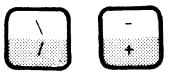
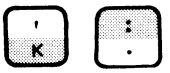
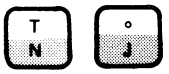
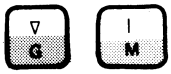
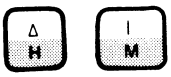
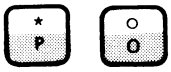
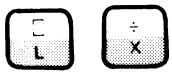
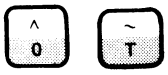


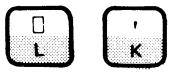
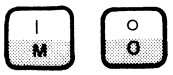
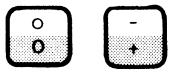
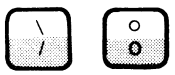
Die ⎕CT-Systemvariable (Toleranz) gibt an, wieviele Stellen bei einer Vergleichsoperation verwendet werden.



Sie haben sicher schon bemerkt, daß APL eine Sprache ist, mit der es sich gut experimentieren läßt. Je mehr Sie experimentieren, je mehr lernen Sie.

Anhang A Zusammengesetzte Zeichen

Name	Zeichen	Tasten
Kommentarzeile	⍝	
Komprimieren	⌿ (siehe Fußnote)	
Aktivieren	⍎	
Expandieren	⍀ (siehe Fußnote)	
Fakultät, Binomial-koeffizienten bilden	!	
Formatisieren	⍕	
Sortierindex bilden (fallend)	⍒	
Sortierindex bilden (steigend)	⍋	
Logarithmieren	⍟	
Invertieren	⌹	
Mit 'NAND' verknüpfen	⍲	
Mit 'NOR' verknüpfen	⍱	
Funktion verdecken	⍫	

Name	Zeichen	Tasten
Texteingabe (alphanumerische Eingabe)	⍞	
Rotieren, parallel spiegeln	⌽	
Rotieren, parallel spiegeln	⊖ (siehe Fußnote)	
Transponieren, diagonal spiegeln	⍉	

Fußnote: Diese Funktionen haben unterschiedliche Bedeutung, je nachdem, welche Koordinate eines Vektors oder einer Matrix angesprochen wird.

Stichwortverzeichnis

)CLEAR-Systemanweisung 107, 109
)CONTINUE-Systemanweisung 107
)FNS-Systemanweisung 106
)LOAD-Systemanweisung 107
)MARK-Systemanweisung 107
)VARS-Systemanweisung 106
[⍳] 45, 46, 48
[□] 78
[□n] 83
[n□] 83
[⍙n] 82
□ 95
□: 95
□CT 109
□IO 108
□PP 36, 109
⍞ 95
→□LC 76, 99
⍬-Zeichen 96, 102
∇-Zeichen 71
⌈-Funktion 29, 41
⌊-Funktion 29, 42
⍋-Funktion 30
⍒-Funktion 30
?-Funktion 31, 43
⍳-Funktion 22, 32
⍴-Funktion 32
<-Funktion 34
≤-Funktion 34
>-Funktion 34
≥-Funktion 34
=-Funktion 34
≠-Funktion 34
∧-Funktion 37, 38
∨-Funktion 37, 38
/-Funktion 44, 45
⍪-Funktion 47
⍟-Funktion 51

Abbrechen
einer Funktion während der Dateneingabe 96
Abrunden ⌊ 29
Absteigend sortieren ⍒ 30
Abwärtstaste 4, 7, 12, 72
Addition 14
Ändern einer Funktion 77
Ändern einer Kopfzeile 84
Aktiver Arbeitsbereich 104
Alphabetische Tasten 2
Alphanumerische Tasten 2
ALREADY MARKED 107
Alternierende Summe 56
Anforderung von Eingabedaten 95
Anweisungen 71
Anweisungsnamen 91
Anweisungsnummer 72
Anzeige Umkehrung 7

Anzeigen
  Anweisung 83, 84
  Daten 5
  Funktion 78
  Funktionsname 106
  Globale Variablennamen 106
APL-Sprache 1
APL-Systemanweisungen 4
Arbeitsbereich 107
Argumente 14, 28, 87
Arithmetische Operationen 14
ATTN-Taste 4, 12, 74, 98
Aufrunden ⌈ 29
Aufsteigend sortieren ⍋ 30
Aufstellungshinweise 1
Aufwärtstaste 4, 7, 12, 72
Ausführungsmodus 71
Ausführungsunterbrechung 75
Austauschen
  Anweisung 80
  Zeichen 8
Auswählen Elemente 44

BASIC/APL-Schalter 4
Bandkassette 104
Bandkassette einlegen 104
Bedienungseinrichtungen 1
Bedienungstasten 4
Bedingte Verzweigung
Bibliothek 104
Bildschirm 1, 5
Bildschirmanzeige 5, 6
- Spreizen 7

CLEAR WS 4
CMD-Taste 4, 9

Dateien 104
Dateneingabe 5
Dateiformate 107
Definitionsmodus 71
Dialog-Funktionen 96
Dimension zeigen ⍴ 32
Division 14
Durchschnitt 53

Einfügen
Anweisungen 81
Zeichen 10
Leerzeichen oder Nullen 45
Eingabe
Daten 5
Daten während Funktionsausführung 95
Eingabezeile 7
Einlegen Bandkassette 104
Eintastfehler 8
Elementarfunktionen 1, 14, 59
Ergebnis Explizites 88
Ergebnisvariable 72, 77, 88
EXECUTE-Taste 4, 6, 12
Expandieren \ 45
Expandieren Vektor oder Matrix 45
Explizites Ergebnis 88

Fehler in einer Funktion 101
Fehlermeldung 18, 74, 101
Fortlaufende Zahlen 31
Funktion
Ändern 77
Ausführungsstop 98
Definitionsmodus 71
Hängende 75, 98
Testen 73
Funktionsdefinition 71

Gleich = 34
Globale
Namen 89
Variablennamen 106
Größer als > 34
Größer gleich ≥ 34
Grundrechenarten 4

Hängende Funktionen 75, 98
Helligkeitssteuerung 7
Hinzufügen von Anweisungen 79
Hochkomma 26
HOLD-Taste 4, 103
HYP-Funktion 71

Index Anfang ⎕IO 108
Index [I] 45, 46, 48
Index eines Wertes 43
Indexvektor bilden 31
Indexzahlen 24
Index zeigen 43
Indiziert 30
Indizierung 24
Interne Genauigkeit 36
Interne Prüfungen 4

Kaufmännisches Runden 29
Klammern 19
Kleiner als < 34
Komprimieren / 44
Koordinate 45, 46
Kopfzeile 71, 84, 88, 89
Korrektur eines Eintastfehlers 8

Lampen 1
Leerer Vektor 32, 50
Leerstelle 26
Löschen
Anweisung 82
Zeichen 9
Logarithmieren ⍟ 51
Logische Daten verknüpfen 37
Lokale
Lokale Namen 89
Lokale Variablen 91
L 32 64 R32-Schalter 4, 7

Maximieren ⌈ 41
Maximum-Reduktion 57
Matrix 22, 26
Matrizen 19
Mehrfachzeichen 28, 112, 113
Minimieren ⌊ 42
Minimum-Reduktion 57
Minus 15
Minuszeichen 15
Minus-Reduktion 56
Modus wechseln 77
Multiplikation

Negative Zahlen 15
Numerische Tasten 4
Numerische Eingabe während einer Funktion 95

Oder ∨ 39, 37
Oder-Reduktion 58

Plus-Reduktion 53
Positionsanzeiger 5
Potenzieren * 16
Protokoll 75

Radizieren 17
Rechentasten 4
Reduktion 53
Register-Anzeige 4
Reihenfolge der Ausführung 18, 19
Rücktaste 4, 8, 9, 10
Runden, kaufmännisch 29

SAFE-Schalter 104
Schalter 1, 4
Schleifenbildung 74, 94
Schreibmaschinentastatur 2
Schwarze Zeichen 7
Schwarzer Hintergrund 7
Semikolon 25
Skalar 26
Sortieren eines Vektors 30
Spalte 22
Speichern von Daten 17
Statusanzeiger 102
Steuerpult 1
Stichprobe nehmen 43
Stopeinrichtung 100
Strukturieren 22
Subtraktion 14
Systemanweisungen 4, 104
Systemvariable 108

Tastatur 1
Tasten 1
Tabellen 19
Testen von Funktionen 73
Testeinrichtung 75
Textdaten 26
Texteingabe während einer Funktion 95
Toleranz □CT 109

Übertragen
  aktiver Arbeitsbereiche 107
  gespeicherter Arbeitsbereiche 107
Umschalttaste 2
Unbedingte Verzweigung 92
Und-Funktion 1, 37, 38
Und-Reduktion 48
Ungleich ≠ 34
Unterbrechung der Verarbeitung 103
Unterbrechungstaste (siehe ATTN-Taste)

Variablen 17
Vektoren 19, 26
Verändern einer Anweisung 83
Vergleichen 34
Verketten 47
Verschieben von Bildschirmzeilen 7
Verzweigen 91
- bedingtes 92
- unbedingtes 92
Vorwärtstaste 4, 8, 9, 10

Weiße Zeichen 7
Weißer Hintergrund 7
Wert zuweisen 17
Wiederanlaufschalter 4
Wiederanlaufverfahren 4
Wurzel 17

Zahlen, negative 15
Zahlenlänge □PP 109
Zahlenreihen 19
Zehnertastatur 3
Zeichen für Texteingabe 26
Zeilen 22
Zeilen, verlorene 7
Zufallszahl 31, 43
Zuweisungszeichen 17

Anregungen

zur Verbesserung und Ergänzung von DV Fachliteratur nehmen wir dankbar entgegen. Benutzen Sie bitte diese Korrekturkarten, wenn Sie Fehler, ungenaue Darstellungen oder irgendwelche andere Mängel entdecken sollten.

IBM Deutschland GmbH
DV Literatur (0426)

Seite	Zeilen (von oben)		Empfohlene Verbesserung IBM-Form
	von	bis	Titel:

Seite	Zeilen (von oben)		Empfohlene Verbesserung IBM-Form
	von	bis	Titel:

Absender _____

_____

_____

_____

IBM Deutschland GmbH
DV Literatur (0426)
Adreßschlüssel 7032-86
Postfach 800880

7000 Stuttgart 80

Absender _____

_____

_____

_____

IBM Deutschland GmbH
DV Literatur (0426)
Adreßschlüssel 7032-86
Postfach 800880

7000 Stuttgart 80

Anfragen für weitere sachliche Auskünfte zum Inhalt dieser Veröffentlichung an:

IBM Deutschland GmbH
DV VF Literatur 0426

Januar 1977
Änderung des vorstehenden Textes bleibt vorbehalten

IBM Form SA12-2153-1

Gesetzt mit IBM Composer

IBM Deutschland GmbH

IBM Form SA12-2153-1