Eurocomp

ACT V

Algebraischer Compiler und Translator für LGP 21

Herausgeber:
POOL EUROPA in der Benutzerorganisation POOL für die elektronischen Rechenanlagen LGP 21, LGP 30 u. RPC 4000 und EUROCOMP GmbH Elektronische Rechenanlagen
495 Minden/Westf. Schillerstraße 72

D 018/166/1-06

POOL EUROPA

in der Benutzerorganisation POOL

für die Rechenanlagen LGP 21, LGP 30 und RPC 4000

A C T V

Algebraischer Compiler und Translator

für den LGP-21

PROGRAMMIERUNGSANLEITUNG

INHALTSVERZEICHNIS

zur Programmierungsanleitung ACT V für LGP - 21

Einleitung

Der Lösungsablauf eines numerischen Problems mit Hilfe des Übersetzers ACT V wird in drei Phasen unterteilt:

1. Die Programmierungsphase

   Zuerst ist es nötig, den gewählten Algorithmus als Folge von Anweisungen in ACT V - Sprache zu schreiben. Das Programm wird auf einem Flexowriter, einer speziellen Schreibmaschine, auf Lochstreifen gestanzt. In ACT - Sprache geschriebene Programme nennen wir Quellenprogramme.

2. Die Compilierphase

   Das Quellenprogramm wird mit Hilfe des ACT V - Übersetzers in die Maschinensprache des LGP-21 übersetzt. Das entstehende Maschinenprogramm (= Objektprogramm) läßt man für spätere Verwendung vom Rechner ausstanzen.

3. Rechenphase

   Das Maschinenprogramm und ein Satz von Unterprogrammen bilden das Rechenprogramm. Das Rechenprogramm wird gestartet und gibt automatisch die Ergebnisse aus.

Die vorliegende Programmierungsanleitung enthält eine ausführliche Beschreibung der ACT V - Sprache und gibt an, wie Quellenprogramme aufzubauen sind.

Die Beschreibung der inneren Zusammenhänge und Vorgänge bei Ausführung der zweiten und dritten Phase sind einem ergänzenden Handbuch, der Bedienungsanleitung, vorbehalten.

K a p i t e l I

Zahlendarstellung

Man unterscheidet grundsätzlich zwei Darstellungsformen von Zahlen, Festkommazahlen und Gleitkommazahlen.

Bei den Festkommazahlen muß die Kommalage vom Programmierer jeweils notiert werden. Bei arithmetischen Operationen ist die Kommalage der Operanden genau zu beachten, damit ein richtiges Ergebnis mit der gewünschten Kommalage erhalten wird.

Bei den Gleitkommazahlen (Darstellung durch Mantisse und Exponent) wird die Kommalage durch den Exponenten vom Programm berücksichtigt. Der Programmierer braucht sich darum nicht zu kümmern.

Ein Sonderfall der Festkommazahlen sind die Ganzzahlen (Integer). Bei ihnen steht das (gedachte) Komma immer hinter der letzten gültigen Ziffer. Auch hier braucht die Kommalage vom Programmierer nicht berücksichtigt zu werden.

Im ACT V können die beiden Darstellungsformen Ganzzahlen und Gleitkommazahlen verwendet werden. Deshalb gibt es zwei Arten von Operatoren, Variablen und Konstanten. Die Umwandlung einer Zahl von einem Typ in den anderen geschieht nicht automatisch, sondern muß programmiert werden. Man muß als Programmierer also aufpassen, daß Operator, Variablen und Konstanten innerhalb jeder Anweisung von demselben Typ sind.

Ganzzahlen

Mit der Ganzzahldarstellung erreicht man die genaueste Darstellung einer Zahl; die Ganzzahl-Arithmetik ist im allgemeinen exakt, Ausnahmen werden bei der Besprechung spezieller Operatoren angemerkt. Der Größenbereich der Zahlen ist jedoch auf

$$-536\ 870\ 911 \leq y \leq 536\ 870\ 911$$

beschränkt.

Ganzzahlen werden hauptsächlich nur für Zähler und Indizes verwendet.

Gleitkommazahlen

Gleitkommazahlen werden in halblogarithmischer Darstellungsweise mit Mantisse a und Exponent m geschrieben:

$$y = a \cdot 10^{m}$$

Der Zahlenbereich der Gleitkommazahlen ist wesentlich größer als der für Ganzzahlen. Es gilt

$$0.10000002 \cdot 10^{-32} \leq y \leq .99999994 \cdot 10^{31}$$

Generell dürfen Zahlen, die größer als die obere Schranke sind, nicht vorkommen. Bei Überläufen in der Rechenphase hält der Rechner an. Zahlen, die kleiner als $0.1 \cdot 10^{-32}$ sind, werden durch 0 ersetzt.

Der Vorteil der Gleitkomma-Arithmetik besteht u.a. darin, daß ein viel größerer Zahlenbereich verfügbar ist. Allerdings büßt man aus den folgenden Gründen Genauigkeit ein:

1. Die Mantisse muß nach jeder Rechenoperation abgerundet werden, damit sie die richtige Wortlänge hat.

2. Wegen der Verwendung des Dualsystems in der Maschine treten kleine Abweichungen bei der Umwandlung der Zahlen auf, z.B. wird die Dezimalzahl 1.0 durch die Gleitkommazahl 0.99999994 dargestellt. Im allgemeinen ist der Umwandlungsfehler jedoch nicht größer als 3 Einheiten in der achten Stelle. Deshalb dürfen Gleitkommazahlen nicht zum Zählen benutzt werden und das Ergebnis von Gleitkommaoperationen sollte nicht auf Null getestet werden, höchstens nur, um Division durch Null zu verhindern. Bei einer Division durch Null in der Rechenphase hält der Rechner an.

K a p i t e l II

Elemente der ACT V - Sprache

Die ACT V - Sprache besteht aus Wörtern mit fünf oder weniger Zeichen des Flexowriters. Jedes Wort wird durch einen Stopcode abgeschlossen. Diese Wörter können Operatoren, Konstante, Marken oder Variable kennzeichnen. Ein Programm besteht aus einer Folge von Anweisungen, die ihrerseits aus mehreren Wörtern gebildet werden. Wir betrachten zunächst die in der Formelsprache des ACT V verfügbaren Zeichen. Ein Zeichen kann eine Ziffer, ein Buchstabe oder eins der folgenden Symbole sein:

\- ; . , / oder Leertaste

Da die Leertaste als Zeichen angesehen wird, ist bei ihrem Gebrauch Vorsicht geboten. Es ist zu beachten, daß der Buchstabe " L " und die Zahl " 1 " identisch sind. Außerdem muß beachtet werden, daß in der ACT V - Sprache nicht zwischen Groß- und Kleinschreibung unterschieden wird.

Aus den Schriftzeichen können Sprachelemente gebildet werden, die wir Wörter nennen. Ein Wort besteht aus 1 bis 5 Zeichen und einem Stopcode (') als Begrenzer gegen das nächste Wort. Es kann Operator, C-Wort, Konstante, Marke oder Variable sein. Eine Folge von Wörtern, die bestimmten Satzregeln untersteht, wird Anweisung genannt. Das Ende einer Anweisung wird durch einen zusätzlichen Stopcode gekennzeichnet. Die letzte Anweisung eines Quellenprogramms ist mit einem weiteren Stopcode zu versehen.

Operator

Einen Operator kann man als " Befehl " für den Rechner auffassen. Die einzelnen Operatoren werden weiter unten besprochen.

C - Wort

Ein C - Wort ist ein Wort, das entweder aus 1 bis 5 Ziffern oder aus " + " oder " . " und 1 bis 4 Ziffern besteht.

Konstante

Eine Konstante ist ein Wort oder ein Satz von Wörtern und stellt einen bestimmten Zahlenwert dar. Das erste Wort muß ein C - Wort sein. Eine im Quellenprogramm niedergeschriebene Konstante ist immer positiv. Es dürfen höchstens 63 verschiedene Konstanten innerhalb eines Programms verwendet werden. Je nach der verwendeten Arithmetik ist zu unterscheiden:

a) Ganzzahlkonstanten

Ganzzahlkonstanten können aus einem oder zwei Wörtern bestehen. Ganzzahlkonstanten bestehend aus fünf Ziffern oder weniger belegen ein Wort von höchstens fünf Ziffern gefolgt von einem Stopcode. Ganzzahlkonstanten von sechs bis neun Ziffern werden in zwei Wörtern dargestellt. Das erste Wort besteht aus einem Pluszeichen gefolgt von eins bis vier Ziffern; das zweite Wort kann eins bis fünf Ziffern haben.

Beispiele:

1608' und +8441'5927'

b) Gleitkommakonstanten

Gleitkommakonstanten werden immer in vier Wörtern dargestellt. Die ersten zwei Wörter bilden die Mantisse. Das erste Wort besteht aus einem Punkt, dem 1 bis 4 Ziffern der Mantisse folgen. Das zweite Wort enthält die restlichen Ziffern der Mantisse und ist auf fünf Ziffern beschränkt. Wenn keine restlichen Ziffern angegeben werden, folgt nur ein Stopcode. Das dritte Wort besteht aus e' bei positivem und e-' bei negativem Exponenten. Das vierte Wort besteht aus dem Absolutwert des Exponenten. Es ist zu beachten, daß zwar 9 Ziffern für die Mantisse zulässig sind,aber intern schon die 8. Stelle mit einem Rundungsfehler behaftet ist.

Beispiele:

.2''e'1' gleich 2.0

.3141'5927'e'1' gleich 3.1415927

Anmerkung:

Die Eingabeformate für Zahlen als Daten werden bei der Besprechung der Operatoren angegeben.

Hinweis:

Die Konstante 1.0 schreibt man am besten als .9999'99999'e'0', da sie auf diese Weise am genauesten dargestellt wird.

Die Null wird sowohl für Ganzzahl- als auch für Gleitkommaarithmetik in der Form 0' geschrieben.

Marken

Marken werden dazu benutzt, Anweisungen für Sprungoperationen zu kennzeichnen. Das erste Zeichen einer Marke ist ein " s ", es folgen 1 bis 4 Ziffern, die Ganzzahlen von 1 bis 255 darstellen, z.B. s10', s0190'.

Variable

Variable sind numerische Größen, deren Wert eingelesen oder während des Programmablaufs errechnet wird. Jede Variable wird durch einen Namen bezeichnet. Dieser Name ist ein Wort aus 5 oder weniger Zeichen. Variablennamen dürfen nicht nur aus Ziffern bestehen. Außerdem dürfen sie nicht mit Operatoren oder Marken identisch sein.

Beispiele für Variablennamen:

ex', var', discr', phi'

Kommentar

Kommentar ist eine Folge von Zeichen und Steuerungssymbolen (außer Tabulator), der ein Stopcode folgt und außerdem folgende Bedingung erfüllt: Der Kommentar muß mindestens 6 Zeichen enthalten und als 6. Zeichen vor dem Stop muß einer der Buchstaben a, b, c, d, e, h, i, m, n, p, r, s, t, u, y, z stehen. Kommentare werden vom Übersetzer ignoriert und können an jeder beliebigen Stelle des Programms eingefügt werden.

Beispiele:

unter 1',inversion'

K a p i t e l III

Grundregeln der Satzlehre

Bei der Verknüpfung von Operanden mit Operatoren zu Anweisungen sind einige wenige Regeln zu beachten. Zunächst ist der Begriff " Typ " von Operanden und Operatoren einzuführen. Wie weiter oben klargestellt wurde, ist streng zu unterscheiden zwischen zwei Arten von Zahlen, Gleitkomma- und Ganzzahlen. Genauso werden die Operatoren unterschieden. Der Typ eines Operanden oder Operators gibt an, um welche Art von Information es sich handelt bzw. auf welche Ausdrücke er anwendbar ist. Für den Typ Gleitkomma (floating point) wird der Buchstabe F und für den Typ Ganzzahl (integer) der Buchstabe I eingeführt. Logische Operatoren werden durch den Buchstaben L gekennzeichnet.

1. Im allgemeinen sind Variable oder Konstante durch einen Operator mit weiteren Variablen oder Konstanten zu verknüpfen. Ausnahmen sind einige Namen mit Sonderbedeutung und indizierte Variable, sowie bestimmte Operatoren, die nur einen oder sogar keinen Operanden benötigen.

2. Der Typ des Operanden muß mit dem Typ des Operators übereinstimmen.

3. Die Reihenfolge der Ausführung der Operationen ist die gleiche wie in der Algebra.

Zuordnungszeichen

Das Gleichheitszeichen der Algebra drückt den Zustand der Gleichheit zweier Größen aus. Im ACT V wird das Gleichheitszeichen nicht benutzt sondern der Substitutions-Operator " ; ". Er hat die folgende Bedeutung: Der Wert des Ausdrucks, der links dieses Operators steht, wird der rechts stehenden Variablen zugeordnet und auf deren Speicherplatz gespeichert.

Beispiel:

```
a'x'a'+'b'x'b';'c''
sum'+'a'x'a';'sum''
```

Vorrang und Klammern

Die Vorrangstufen der algebraischen Operatoren entsprechen der gewöhnlichen Ordnungskonvention der Algebra. Bei Abwesenheit von Klammern werden die Funktionen (abs, sin, ln usw.) zuerst ausgeführt, haben also den höchsten Vorrang. Multiplikation und Division haben den nächst niedrigeren Vorrang, Addition und Subtraktion liegen eine weitere Stufe tiefer, und die Ein- und Ausgabeoperatoren sowie der Substitutions-Operator haben den niedrigsten Vorrang.

Klassifikation der Operatoren nach Vorrangstufe:

Operatoren	Vorrang
Standardfunktionen	3
Umwandlungsoperatoren	3
Multiplikation und Division	2
Addition und Subtraktion	1
Ein- und Ausgabe	0
Zuordnungszeichen	0

Wie in der Algebra kann durch das Setzen von Klammern die Reihenfolge der Ausführung der Operationen abgeändert werden. Die innersten Klammern haben den höchsten Vorrang, die äußersten den niedrigsten. In einem Klammerausdruck gelten die normalen Vorrangstufen. Im Quellenprogramm müssen eckige Klammern geschrieben werden. Es dürfen höchstens 7 sich öffnende Klammern aufeinander folgen. Die Anzahl der schließenden Klammern muß mit der der öffnenden Klammern übereinstimmen.

Beispiel:

```
['a'+'b'x'c'] '/'['d'-'e'];'y''
```

Reihenfolge in der Ausführung:

1. b * c
2. a + b * c
3. d - e
4. [a + b * c] / [d - e]
5. [a + b * c] / [d - e] ⇒ y

K a p i t e l IV

Arithmetische Operatoren

In diesem Kapitel werden die arithmetischen Operatoren und die im ACT V enthaltenen transzendenten Funktionen besprochen. Außerdem werden die Operatoren zur Umwandlung von Variablen von Gleitkomma in Ganzzahl und umgekehrt erklärt.

Operatoren der Gleitkomma-Arithmetik

a) Die arithmetischen Grundoperatoren.

Die Symbole für die Grundoperationen sind die gleichen wie in der Algebra.

;	z.B.	a';'b'.	Der Wert von a wird der Variablen b zugewiesen, dabei bleibt die Variable a unverändert. Der rechte Operand muß eine Variable sein.
+	z.B.	a'+'b'.	Der Wert von a wird zu dem Wert von b addiert.
-	z.B.	a'-'b'.	Der Wert von b wird von dem Wert von a subtrahiert.
x	z.B.	a'x'b'.	Multiplikation der Größen a und b.
/	z.B.	a'/'b'.	Division der Größen a und b.
0-	z.B.	0-'a'.	Das ist ein Operator mit nur einem Operanden, er kehrt das Vorzeichen seines Operanden um. Er hat den Vorrang 3, d.h. daß jeder zusammengesetzter Ausdruck, welcher Operand von " 0- " ist, in Klammern gesetzt werden muß.
	Beispiele:		a'x'0-'['b'+'c'x'e']'
			a'x'0-'b'
abs	z.B.	abs'a'.	Das ist ebenfalls ein Operator mit nur einem Operanden. Er bildet den Absolutwert seines Operanden und hat den Vorrang 3.

b) Potenzrechnung.

pwr	z.B.	a'pwr'b'.	Der Operator pwr erhebt den linken Operanden (Typ F und positiv) in die Potenz des rechten Operanden (Typ F). Es wird also a^b gebildet. Der Operator hat den Vorrang 3.

Beispiel: $w = a/(b-c^{-y^2})$

wird geschrieben als

```
a'/'['b'-'c'pwr'['0-'y'x'y']']';'w''
```

Potenzen von Gleitkommazahlen mit kleinen ganzzahligen positiven Exponenten werden besser durch Multiplikationen dargestellt.

x10p z.B. a'x10p'b'. Der Operator x10p bewirkt, daß der linke Operand a (Typ F) mit 10^b multipliziert wird, b ist vom Typ I. Der Operator hat Vorrang 3.

c) Allgemeine Funktionen.

Im ACT V können die folgenden Funktionen benutzt werden:

sqrt z.B. sqrt'a', Quadratwurzel aus a. Der rechte Operand muß größer oder gleich Null sein.

ln z.B. ln'a', natürlicher Logarithmus von a. Der rechte Operand muß größer als Null sein.

log z.B. log'a', Logarithmus zur Basis 10 von a. Der rechte Operand muß größer als Null sein.

exp z.B. exp'a', Exponentialfunktion von a zur Basis e (e-Funktion). Der rechte Operand muß kleiner als 13 897 423 sein.

sin z.B. sin'a', Sinus von a. Das Argument a muß in Bogenmaß angegeben werden und muß kleiner als 10^8 sein.

cos z.B. cos'a', Cosinus von a. Das Argument a muß in Bogenmaß angegeben werden und muß kleiner als 10^8 sein.

artan z.B. artan'a', Arcustangens von a. Das Argument a wird in Bogenmaß eingegeben. Als Ergebnis wird der Hauptwert ($-\pi/2 \leq y \leq +\pi/2$) ausgegeben.

Diese Funktionen sind Gleitkommaoperatoren und haben den Vorrang 3.

d) Zufallszahlen.

randm Der Operator " randm " hat keine Operanden. Bei Ausführung der Operation wird eine Zufallszahl aus dem Intervall 0 bis 1 als Gleitkommazahl ermittelt. Bei mehrfacher Ausführung dieser Operation verteilen sich die erhaltenen Zufallszahlen gleichmäßig auf das bezeichnete Intervall. Der Operator randm hat den Vorrang 3.

Anwendungsbeispiele:

```
randm'/'randm';'y''
```

Der Quotient von zwei Zufallszahlen wird der Variablen y zugeordnet.

```
sin'['randm']';'y''
```

Der Sinus einer Zufallszahl zwischen 0 und 1 wird der Variablen y zugeordnet.

Operatoren der Ganzzahl-Arithmetik

Im allgemeinen werden Ganzzahlen für Indizes und Abzählvorgänge benutzt. Der Absolutwert von Ganzzahlen ist auf 536 870 911 beschränkt. Ganzzahl-Operatoren beginnen im allgemeinen mit dem Buchstaben " i ".

i+ z.B. a'i+'b', Addition

i- z.B. a'i-'b', Subtraktion

ix z.B. a'ix'b', Multiplikation mit Fehlerstop bei Überlauf.

nx z.B. a'nx'b', Multiplikation ohne Fehlerstop bei Überlauf. Wenn keine Gefahr von Überläufen besteht, ist dieser Operator vorzuziehen, da er viel schneller abläuft.

i/ z.B. a'i/'b', Division. Da der Quotient zweier ganzer Zahlen im allgemeinen keine ganze Zahl ist, wird folgende Übereinkunft getroffen: Das Ergebnis einer Ganzzahl-Division wird immer auf eine ganze Zahl abgerundet. Der Rest wird unter dem Namen "remdr" gespeichert und kann im Programm weiter verwendet werden.

Beispiele:

a'i/'b'		Quotient	Rest
10	5	2	0
11	5	2	1
-11	5	-3	4
-11	-5	2	-1
11	-5	-3	-4

iabs z.B. iabs'a', Absolutwert einer Ganzzahl. Der Operator hat den Vorrang 3.

ipwr z.B. a'ipwr'b', der Operator ipwr berechnet den Wert a^b, er hat den Vorrang 3. Bei Überlauf erfolgt Fehlerstop.

Operatoren zur Formatumwandlung

Die folgenden Operatoren werden benötigt, um Konstante und Variable von Ganzzahl in Gleitkomma und umgekehrt umzuwandeln. Sie haben den Vorrang 3.

flo z.B. a'flo'b'. Die Ganzzahl b wird in eine Gleitkommazahl umgewandelt. Der rechte Operand wird vor der Umwandlung durch 10^a dividiert. Sowohl linker wie rechter Operand sind vom Typ I.

Beispiel:

3'flo'1234' wandelt die Ganzzahl 1234 in die Gleitkommazahl 1.234 um.

unflo z.B. a'unflo'b'. Die Gleitkommazahl b wird in eine Ganzzahl umgewandelt. Der rechte Operand wird vor der Umwandlung mit 10^a multipliziert (a muß vom Typ I sein). Das Resultat wird gerundet.

Beispiel: 1'unflo'b' mit b = 1.357 ergibt die Ganzzahl 14.

fix z.B. a'fix'b'. Die Gleitkommazahl b wird in eine Ganzzahl umgewandelt. Der rechte Operand wird vor der Umwandlung mit 10^a multipliziert, das Resultat wird stets abgerundet.

Beispiel: 1'fix'b' mit b = 1.357 ergibt die Ganzzahl 13.

K a p i t e l V

Programmablauf

Die Anweisungen eines Programms werden normalerweise in der Reihenfolge ausgeführt, in der sie im Quellenprogramm niedergeschrieben worden sind. In diesem Kapitel werden die Operatoren besprochen, mit denen man den normalen Programmablauf abändern kann.

Marken und markierte Anweisungen

Wie im Kapitel II definiert wurde, bestehen Marken aus dem Buchstaben " S " gefolgt von 1 bis 4 Ziffern. Die Ziffern stellen ganze Zahlen dar, die kleiner als 255 und größer als 0 sein müssen. Der Rechner speichert die Anfangsadressen von Anweisungen, deren erstes Wort eine Marke ist, und setzt am Ende der Übersetzung in allen Anweisungen für die Marken die Adressen ein. Dadurch ist es möglich, von jeder Stelle eines Programms zu solchen markierten Anweisungen zu gelangen.

Sprungoperatoren

Es werden zwei Sprungoperatoren unterschieden: Der unbedingte Sprungoperator und der bedingte.

use z.B. use'S10'. Der Operator bewirkt einen unbedingten Sprung zu der durch S10 markierten Anweisung. Er hat den Vorrang 0.

if neg zero pos z.B. if'a'neg'S10'zero'S11'pos'S12''

Der Operator " if " bewirkt lediglich, daß sein rechter Operand im Akkumulator bereitgestellt wird. Er hat den Vorrang 0. Jeder der nachfolgenden Testoperatoren " neg ", " zero ", " pos " testet den Wert des linken Operanden und springt, wenn die Bedingung erfüllt ist, auf die im rechten Operanden angegebene Anweisung. Die Reihenfolge der Tests darf nicht vertauscht werden, aber es dürfen überflüssige Tests weggelassen werden. Wenn keine der angegebenen Bedingungen erfüllt ist, wird die unmittelbar folgende Anweisung ausgeführt. Die verwendeten Marken müssen im Programm vorkommen. Die Variable " a " darf Ganzzahl oder Gleitkommazahl sein.

Steht die zu testende Variable aus einer früheren Rechnung noch im Akkumulator, so dürfen der Operator " if " und sein Operand fehlen. Auf Gleitkommazahlen ist der " zero "- Test wegen der Umwandlungs- und Rundungsfehler im allgemeinen nicht anwendbar.

Verteiler

Bei Verteilern (=variabler Sprung) kann das Sprungziel durch spezielle Anweisungen abgeändert werden.

go to z.B. s20'go to's0''. Der Verteiler " go to " ist ein unbedingter Sprung mit variabler Sprungadresse. Der Operand " s0 " ist als Symbol für das variable Sprungziel anzusehen. Es ist eine spezielle Marke, die nur für diesen Zweck verwendet werden darf. Es ist zu beachten, daß " go to" mit Leertaste zu schreiben ist. Ferner ist die Schreibweise s0 zu beachten (nur eine Null!)

set z.B. set's20'to's15''. Diese Anweisung setzt in der Rechenphase in den mit s20 markierten Verteiler die Marke s15 ein. Jeder Verteiler muß, bevor er in der Rechenphase erreicht wird, gesetzt werden. Falls ein Verteiler nicht gesetzt wurde, erfolgt ein entsprechender Fehlerstop in der Rechenphase.

ret z.B. ret's20'use's10''. Der Rückkehroperator " ret " dient dazu, in Verteilern Rückkehradressen einzusetzen. Die als Beispiel gewählte Anweisung setzt in der Rechenphase das Sprungziel des mit s20 markierten Verteilers auf die der ret-use-Anweisung unmittelbar folgende Anweisung, die keine Marke zu tragen braucht. Anschließend erfolgt ein Sprung nach s10. Damit hat man die Möglichkeit, an verschiedenen Stellen eines Programms einen Block von Anweisungen, beginnend mit einer Marke z.B. s10' und endend mit einem Verteiler z.B. s20'go to's0'' als Unterprogramm in die Berechnung einzuschalten.

Schleifen

Sollen bestimmte Programmteile mehrmals, z.B. mit verschiedenen Elementen von indizierten Größen (s. Kapitel VII), durchlaufen werden, so benutzt man die " for " -Anweisung, z.B.

for'i'step'j'until'n'repeat's20''

Diese Anweisung erhöht den Wert der Variablen i um den Betrag j und vergleicht ihn anschließend mit dem Grenzwert n. Die drei Größen i, j und n müssen Ganzzahlen sein. Solange die Variable i kleiner oder gleich n ist, wird nach s20 gesprungen, im anderen Fall wird die nächste Anweisung ausgeführt.

Die " for " -Anweisung wird zur Programmierung von Schleifen benutzt. Es ist zu beachten, daß die Variable i nach Erreichen des Grenzwertes noch einmal erhöht wird. So ist z.B. nach Verlassen der Schleife

```
1';'i''
s20'read'a'i''
for'i'step'1'until'30'rpeat's20''
```

die Variable i = 31.

Es ist zu betonen, daß die " for "-Anweisung nur auf Ganzzahlen anwendbar ist. Schleifensteuerung mit Gleitkommazahlen hat über Gleitkommaoperatoren und Testoperatoren zu erfolgen.

Stop der Rechenmaschine

stop — Der " stop " -Operator hat weder linken noch rechten Operanden. Wenn dieser Operator in der Rechenphase erreicht wird, hält der Rechner an. Durch Drücken der Start-Taste kann man den Programmablauf mit der nächsten Anweisung fortsetzen.

wait — Der Operator " wait " hat nur in der Compilierphase eine Wirkung. Nach dem Einlesen von wait' wird das Compilieren unterbrochen und der Rechner hält an. Er hat weder linken noch rechten Operanden, und auf ihn darf nur ein Stopcode folgen. Nach Drücken der Start-Taste wird weiter compiliert.

K a p i t e l VI

Ein - und Ausgabe

Es werden jetzt die Operatoren zur Eingabe von Daten und zur Ausgabe von Ergebnissen besprochen.

Numerische Eingabe

1. Eingabe von Gleitkommazahlen

read — z.B. read'a''. Es wird eine Gleitkommazahl eingelesen und unter dem Namen gespeichert, der als rechter Operand angegeben ist. Der rechte Operand darf kein zusammengesetzter Ausdruck sein. Für den Operator " read " ist das folgende Eingabeformat vorgeschrieben: Mantisse bestehend aus Vorzeichen und 1 bis 7 Ziffern. Ein Dezimalpunkt wird nicht eingegeben, er steht definitionsgemäß hinter dem Vorzeichen. Die Mantisse wird durch einen Stopcode abgeschlossen. Anschließend folgt der Exponent mit Vorzeichen und 1 bis 2 Ziffern und ein Stopcode.

Beispiele:

+1'+1' entspricht $0.1 \cdot 10^{1} = 1.0$

-53'+0' entspricht $-0.53 \cdot 10^{0} = 0.53$

2. Eingabe von Ganzzahlen

iread

z.B. iread'a''. Es wird eine Ganzzahl eingelesen und unter dem Namen gespeichert, der als rechter Operand angegeben ist. Der rechte Operand darf kein zusammengesetzter Ausdruck sein. Für den Operator " iread " ist das folgende Eingabeformat vorgeschrieben: Vorzeichen und 1 bis 7 Ziffern gefolgt von einem Stopcode. Führungsnullen brauchen hinter dem Vorzeichen nicht abgelocht zu werden. Bei positiven Zahlen kann entweder das Pluszeichen oder eine Leertaste verwendet werden.

Beispiele:

```
     +4' entspricht der Zahl     4
    -12' entspricht der Zahl   -12
 +48753' entspricht der Zahl 48753
+0000023' entspricht der Zahl    23
-0000171' entspricht der Zahl  -171
```

3. Ganzzahleingabe mit anschließender Gleitkommaumwandlung

rdflo

z.B. n'rdflo'a''. Mit Hilfe des Operators " rdflo " können Ganzzahlen eingelesen und in Gleitkommazahlen umgewandelt werden. Die als Beispiel gewählte Anweisung hat dieselbe Wirkung wie die Anweisungen

```
iread'a''
n'flo'a';'a''
```

Der linke Operand darf ein Ganzzahl-Ausdruck und der rechte Operand muß eine Ganzzahl-Variable sein. Der Wert, der auf dem Platz von a gespeichert wird, hat Gleitkommaformat. Der Wert von " n " gibt an, um wieviel Stellen das Komma verschoben werden soll; für positives " n " erfolgt eine Verschiebung nach links, für negatives nach rechts.

Beispiele:

n'rdflo'a'' mit n = 1 und mit a = 12	ergibt die Gleitkommazahl 1.2
-2'rdflo'a'' mit a = 12	ergibt die Gleitkommazahl 1200
['n'i+'m']'rdflo'a'' mit n = 1, m = 1 und a = 12	ergibt die Gleitkommazahl 0.12

Die drei Eingabeoperatoren " read ", " iread " und " rdflo " haben alle Vorrang 0 und dürfen auch als Operanden benutzt werden.

Beispiele:

```
read'a';'b';'c''
```

Es wird eine Gleitkommazahl a eingelesen und unter a, b und c gespeichert.

```
['read'a']'x'b';'c''
```

Es wird eine Gleitkommazahl a eingelesen, unter a gespeichert, das Produkt a · b gebildet und der Variablen c zugeordnet.

4. Der Eingabe-Verteiler

Der normale Programmablauf in der Rechenphase geht nach Ausführung einer Eingabe-Anweisung zur folgenden Anweisung weiter. Falls jedoch ein Wort eingelesen wird, das nur aus einem Stopcode besteht (Leerwort), läuft das Programm auf den Eingabe-Verteiler. Dieser Verteiler ist vorher durch den Operator " rdxit " auf die gewünschte Sprungadresse zu setzen, z.B. rdxit's10''.

Der Eingabeverteiler kann dazu benutzt werden, die Eingabe von Datenblöcken zu vereinfachen. Es muß nicht angegeben werden, wie lang die einzelnen Datenblöcke sind, da das Ende von Datenblöcken durch den zusätzlichen Stopcode gekennzeichnet wird. Der Eingabe-Verteiler behält sein Sprungziel solange bei, bis es in der Rechenphase geändert wird.

Anmerkung: Das Datenwort 0' ist kein Leerwort.

Numerische Ausgabe

1. Formatsteuerung für numerische Ausgabe

Die Ausgabe-Operatoren benötigen alle als linken Operanden eine Ganzzahl n, die sich folgendermaßen zusammensetzt:

$$n = 100 \cdot c + f$$

Die positive ganze Zahl " c " gibt an, daß die auszugebende Zahl in einem Feld von insgesamt " c " Anschlägen gedruckt werden soll. Die positive ganze Zahl " f " gibt an, daß in dem Ausdruck " f " Stellen hinter dem Komma erscheinen sollen. Die gewünschte Gesamtlänge wird dadurch erreicht, daß das Feld durch Leertasten vor der eigentlichen Zahl aufgefüllt wird. Statt des Pluszeichens wird eine Leertaste ausgegeben.

2. Operatoren für numerische Ausgabe

iprt — z.B. 1204'iprt'a''. Bei Ausführung des Ganzzahloperators " iprt " wird der Wert des rechten Operanden in dem durch den linken Operanden angegebenen Format mit Vorzeichen und Dezimalpunkt gedruckt. Ist die Anzahl der Dezimalstellen f = 0, so wird kein Dezimalpunkt gedruckt. Der Wert von f darf höchstens gleich 8 sein. In dem gewählten Beispiel sei a = -13745, gedruckt wird -1.3745, nachdem 5 Leertasten ausgeführt worden sind.

print — z.B. 1606'print'a''. Der Operator " print " dient zur Ausgabe von Gleitkommazahlen in halblogarithmischer Darstellung. Die Mantisse wird gerundet. Der rechte Operand wird unter der Format-Steuerung des linken Operanden folgendermaßen ausgeschrieben:

a) Wenn die Feldgröße c größer als f + 7 ist, besteht die Ausschrift aus c-f-7 Leertasten, Vorzeichen, Dezimalpunkt, f Stellen der Mantisse, Leertaste, dem Buchstaben e, Vorzeichen und 2 Ziffern des Exponenten. In dem gewählten Beispiel sei a = 23.5177, gedruckt wird .235177 e 02.

b) Für $7 < c \leq f + 7$ wird im Gegensatz zu a) die Mantisse mit c - 7 Stellen ausgegeben. Die Feldgröße c muß größer als 7 sein.

dprt z.B. 1206'dprt'a''. Der Operator " dprt " dient zur Ausgabe von Gleitkommazahlen, in der üblichen Dezimalform. Der linke Operand dient der Formatsteuerung. Der Wert des Exponenten der Gleitkommazahl, die gedruckt werden soll, wird mit " e " bezeichnet.

a) Ist c größer als e + f + 1 und e größer als Null, besteht die Ausschrift aus c - e - f - 2 Leertasten, Vorzeichen, e Dezimalziffern, Dezimalpunkt und f Dezimalbruchziffern gerundet auf die letzte gedruckte Ziffer. Für e kleiner oder gleich Null ist in dem Druckformat e durch 0 zu ersetzen.

b) Wenn e größer als 0 ist und c kleiner als e + f + 2, ist in dem unter a) angegebenen Druckformat f durch c - e - 2, oder wenn dieser Ausdruck negativ ist, durch 0 zu ersetzen.

Die Operatoren " iprt ", " print " und " dprt " haben alle den Vorrang 0. Im allgemeinen darf nur ein solcher Operator in einer Anweisung enthalten sein. Nur wenn Ausdrücke gleichen Ranges eingefügt werden, dürfen mehrere solche Operatoren in einer Anweisung vorkommen.

Alphanumerische Ein- und Ausgabe

daprt z.B. daprt's't'o'p''. Durch diesen Operator hat man die Möglichkeit, alle Schreibfunktionen des Flexowriters vom Programm zu steuern. In dem obigen Beispiel werden alle Maschinenbefehle erzeugt, damit in der Rechenphase das Wort "stop" gedruckt wird. Es können beliebig viele Zeichen in einer einzigen " daprt "-Anweisung zusammengefaßt werden. Jedes Zeichen muß durch einen Stopcode abgeschlossen werden.

Für die Steuerungssymbole gelten die folgenden Codes:

Umschaltung auf Kleinschreibung	lc1'
Umschaltung auf Großschreibung	uc2'
Farbumschaltung	color'
Wagenrücklauf	cr4'
Rücktaste	bs5'
Bedingter Stop	stop'
Apostroph	ap'
Tabulatorsprung	tab6'

Die beiden folgenden Codes haben eine besondere Bedeutung:

Bedingter Stop	stopf'
Unbedingter Stop	stop9'

Die ihnen entsprechenden Zeichen werden in der Rechenphase nur über Schnellstanzer ausgegeben. Bei Ausgabe über Flexowriter werden diese Symbole ignoriert. Wenn ein so gestanzter Lochstreifen vom Flexowriter wieder eingelesen wird, halten diese Codes den Flexowriter an, ohne daß etwas ausgedruckt wird. Das Symbol " stopf " hält den Flexowriter nur an, wenn COND. STOP am Flexowriter nicht gedrückt ist.

aread — z.B. aread'a''. Es werden 5 alphanumerische Zeichen, die durch Stopcodes voneinander getrennt sind (wie bei " daprt "), vom Datenlochstreifen eingelesen und unter dem Namen " a " gespeichert.

Steuerungssymbole können genau so codiert werden wie bei " daprt " angegeben. Dann ist die Eingabe des Datenlochstreifens auch über den Flexowriter möglich.

Eine unmittelbare Eingabe der Steuerungssymbole in einem aread-Wort ist nur über den Schnell-Leser möglich.

Die 5 Zeichen werden im 6-Bit-Modus in den Bitpositionen 0 . . . 29 gespeichert und in Bitposition 30 wird eine 1 als Schlußzeichen hinzugefügt.

aprt — z.B. aprt'a''. Der Operator bewirkt das Ausdrucken von Bit 0 . . . 29 des rechten Operanden in 5 Zeichen zu je 6 Bits.

Die Variable a wird im allgemeinen durch aread eingegeben. Es können aber auch 5 beliebige alphanumerische Zeichen in den Bitpositionen 0 . . . 29 von Hand codiert werden. Es ist dann aber darauf zu achten, daß immer eine 1 @ 30 zugefügt werden muß, damit das Wort ordnungsgemäß über " aprt " ausgegeben werden kann.

Das so entstandene hexadezimale Wort kann dann mit dem Operator " rdhex " (s.u.) eingelesen werden.

Ausgabe in wieder einlesbarer Form

Oft ist es wünschenswert, bestimmte Ausgabewerte des einen Programms als Eingabe für ein anderes zu benutzen. Hierfür stehen mehrere Operatoren zur Verfügung. Die weiter oben besprochenen Ausgabeformen stören solange nicht, wie keine Stopcodes mit " daprt " ausgegeben werden.

punch — z.B. punch'a''. Der Operator " punch " ist ein Gleitkommaoperator und stanzt den Wert von a in der folgenden Form aus: Die Mantisse wird mit Vorzeichen, 7 Ziffern (gerundet) und Stopcode ausgegeben. Es folgt der Exponent mit Vorzeichen, zwei Ziffern und Stopcode. Diese Ausgabeform ist mit " read " wieder einzulesen.

ipch — z.B. ipch'a''. Der Operator "ipch " ist ein Ganzzahloperator und stanzt den Wert von a mit Vorzeichen, 7 Ziffern (mit Führungsnullen) und Stopcode aus. Diese Ausgabeform ist mit " iread " wieder einzulesen.

iprt — kann in der speziellen Form O'iprt'a'daprt'stop'' verwendet werden. " iprt " ist ein Ganzzahl-Operator. Bei der Ausführung dieser Anweisung wird der Wert von a mit Vorzeichen und ohne Führungsnullen und einem Stopcode gestanzt. Diese Ausgabeform kann mit " iread " wieder eingelesen werden. Es ist zu beachten, daß eine solche Ausgabe nur auf die in diesem Abschnitt besprochenen Ausgabeoperatoren folgen darf, da vorhergehende andere Ausgabeoperatoren das korrekte Einlesen über iread stören können.

hxpch — z.B. hxpch'a''. Der Operator " hxpch " stanzt den Wert von a als hexadezimales Wort mit 8 Zeichen und Stopcode aus.

rdhex — z.B. rdhex'a''. Der Operator " rdhex " liest ein hexadezimales Wort ein und speichert es unter dem Namen, der als rechter Operand angegeben ist. Dieser Operator wird zur Eingabe von Streifen benutzt, die mit " hxpch " ausgegeben worden sind. Der Eingabe-Verteiler ist bei "rdhex" unwirksam.

Die Operatoren des Ausgabe / Eingabe - Systems haben den Vorrang 0.

K a p i t e l VII

Indizierte Größen

Vektoren und Matrizen sind die wichtigsten Beispiele für indizierte Größen. Es wird eine Vielzahl von Elementen mit einem einzigen Namen bezeichnet. Die einzelnen Elemente selbst werden fortlaufend numeriert. In ACT V können einfach- und doppelindizierte Größen verwendet werden. Der Speicherplatz für solche Größen ist durch den speziellen Operator " dim " (für Dimension) zu reservieren.

Die Anweisung

```
dim'a'10'b'55''
```

reserviert ein Feld von 10 Elementen unter dem Namen " a " und ein Feld von 55 Elementen unter dem Namen " b ". Im Quellenprogramm können mehrere " dim "-Anweisungen verwendet werden. Wichtig ist, daß die " dim "-Anweisung für indizierte Größen vor den Anweisungen steht, in denen mit ihnen gerechnet wird.

Einfach - Index

In der obigen " dim "-Anweisung werden für die Größe a zehn Speicherplätze festgelegt, die durch a'0', a'1', ..., a'9' aufgerufen werden können. Es ist zu beachten, daß die Numerierung bei Null beginnt. Falls " a " ohne Index erscheint, wird es als a'0' identifiziert.

Das Element a'10' bezieht sich auf den ersten Speicherplatz nach a'9', das ist im obigen Beispiel b'0'. Es besteht die Möglichkeit, indizierte Größen als Elemente von vorher definierten anderen Feldvariablen aufzurufen, z.B. b'5' durch a'15'.

Neben festen Indizes können auch variable Indizes verwendet werden, z.B. a'i' oder b'j'. Dem variablen Index ist vorher ein fester Wert zuzuweisen, z.B.

```
1';'i''
s1'read'a'i''
for'i'step'1'until'20'rpeat's1''
```

Doppel - Index

Doppelindizierte Variable werden in der Form a'i'j' geschrieben, wobei i der Zeilenindex und j der Spaltenindex ist.

Da der Rechner nur eindimensionale Felder speichern kann, rechnet er doppelt indizierte Größen in eindimensionale Variable um. Deshalb muß die Spaltenzahl n der doppelt indizierten Größen auf dem ersten Speicherplatz des Feldes, z.B. a'0' gespeichert werden.

Beispiel: Einlesen einer Matrix mit m Zeilen und n Spalten.

```
dim'a'401''
iread'm'
iread'n''n';'a'0''
1';'i''
s1'1';'j''
s2'read'a'i'j''
for'j'step'1'until'n'rpeat's2''
for'i'step'1'until'm'rpeat's1''
stop'''
```

Die Elemente a'i'j' der Matrix werden hier zeilenweise eingelesen, zuerst die 1. Zeile, dann die 2. usw. bis zur m-ten Zeile.

Indizierte Marken

Der Markenbegriff kann durch den Gebrauch von Indizes erweitert werden. Das Wortpaar s1'i' mit dem Index i wird als variable Marke behandelt. Die Anweisung unmittelbar vor s1' muß aus einer Reihe von " use "-Operatoren bestehen, die als Transfer-Vektor bezeichnet werden.

Beispiel:

```
:
0';'i''
:
use's5'use's20'use's3'use's75''
s1'
:
i'i+'1';'i''
use's1'i''
:
```

Die Marke s1'i' wird für die Werte i = 0,1,2,3,4 den Marken s1, s75, s3, s20, s5 gleichgesetzt. Es ist zu beachten, daß die Elemente des Transfer-Vektors von rechts nach links numeriert werden. Falls der Indexwert null ist, bezieht sich die indizierte Marke auf die Anweisung die dem Transfer-Vektor folgt. Indizierte Marken dürfen in Verteilern, " ret-use " und " set-to" - Anweisungen nicht auftreten.

K a p i t e l VIII

Ergänzungen

Spezielle Operatoren

,, z.B. read'a',,'b',,'c',,'d''. Der Operator " ,, " ist ein Wiederholungsoperator. Die gewählte Anweisung ist identisch mit read'a' read'b' read'c' read'd''. Der Operator kann angewendet werden auf: read', iread', punch', ipch', aread', aprt', rdhex', hxpch', use', print', iprt', dprt' und arg'. Er darf nicht auf rdflo' angewendet werden. Bei den Operatoren print', iprt' und dprt' erfolgt die Wiederholung mit dem ursprünglichen Format. Z.B.

1608'print'a',,'b',,'c''

a, b und c werden im Format 1608 ausgegeben.

machn z.B. machn'r2830u2800'z0416'mt1234''. Der Operator " machn " ermöglicht es, in ein Quellenprogramm Maschinenbefehle mit fester Adresse einzufügen. Wenn der Maschinenbefehl negatives Vorzeichen hat, ist statt des Minuszeichens ein " m " zu schreiben. Die obige Anweisung erzeugt in dem Objektprogramm die Befehle: r2830, u2800, z0416, -t1234. Der Operator " machn " soll mit seinen Operanden immer eine Anweisung für sich bilden.

prev Dieser Operator hat keinen Operanden, besitzt den Vorrang 3 und ist vom gleichen Typ wie das Ergebnis der vorhergehenden Anweisung. Er macht den Akkumulatorinhalt sofort verfügbar und verkürzt die Rechenzeit eines Programmes. Es ist zu beachten, daß " prev " der erste Operator einer Anweisung sein muß, z.B.

prev'+'a'3';'a'2''

Der Operator " prev " kann beim Eingang in eine Prozedur oder in einer Anweisung, die verschiedene Programmzweige wieder vereinigt, erfolgreich angewendet werden.

// z.B. a'k'//'20''. Der Operator " // " dient zur Erhöhung des Index von einfach-indizierten Größen.

In dem gewählten Beispiel wird die Anfangsfeldadresse der Feldvariablen a schon bei der Compilierung um 20 Schritte erhöht, so daß der Index der Größe a in der Rechenphase immer k + 20 ist.

Deshalb ist bei einem konstanten einfachen Index die Schreibweise a'//'7' wirkungsvoller als die Form a'7'. Bei der ersteren wird die richtige Adresse des Feldelementes bereits in der Compilierphase eingesetzt, bei der letzteren muß sie in der Rechenphase ermittelt werden, was zusätzlich Zeit kostet.

Anmerkung: Die Schreibweise a'k'//'0'' ist nicht erlaubt.

Bei Doppelindizes darf der Operator //' nicht verwendet werden. So ist z.B. $a_{3,4}$ zu schreiben:

a'3'4'

Wenn ein Prozedurname als indizierte Größe verwendet wird (s. Seite 26), so muß die Form

name'//'3'

im Quellenprogramm geschrieben werden. Die Schreibweise name'3' ist falsch.

bkp4
bkp8
bkp16
bkp32

z.B. bkp4'use's7''. Diese Operatoren ermöglichen eine einfache Programmierung von Programmverzweigungen. Bei gedrückter PS - Taste wird der nächste Befehl, im anderen Fall der übernächste Befehl ausgeführt.

Programmierung im Maschinencode

Die ACT V - Sprache enthält Befehlssymbole für die meisten Maschinenbefehle. In der folgenden Tabelle werden diese Operatoren zusammengefaßt:

Operator	Maschinenbefehl
stop'	z0000
bring'a'	b(a)
stadd's1'	y(s1)
ret's1'	r(s1)
div'a'	d(a)
mult'a'	m(a)
extrt'a'	e(a)
use's1'	u(s1)
trn's1'	t(s1)
hold'a'	h(a)
clear'a'	c(a)
add'a'	a(a)
subtr'a'	s(a)

Diese Operatoren haben den Vorrang 0. Sie ermöglichen eine symbolische Programmierung (keine Angabe von festen Adressen).

Interne Zahlendarstellung

Ganzzahlen, Gleitkommazahlen und alphanumerische Informationen können in dem vorgeschriebenen Format codiert und über " rdhex " eingelesen werden.

1. Ganzzahlen stehen bei q = 29. Das Bit bei q = 30 ist null.

2. Die Mantisse von Gleitkommazahlen wird in den Bitstellen 0 bis 24 einschließlich bei q = 0 gehalten. Die Mantisse wird auf die letzte Stelle gerundet. In den Stellen 25 bis 30 steht der um 32 erhöhte Exponent bei q = 30.

3. Worte, die durch " aprt " gedruckt werden sollen, bestehen aus 5 Zeichen von je 6 Bit Länge, die in den Bits 0...29 stehen. Bit 30 enthält zusätzlich eine 1. So wird z.B. für 579G6566 ausgegeben: daten.

K a p i t e l IX

Prozeduren

In der ACT V - Sprache werden zwei Arten von Unterprogrammen unterschieden, " ret-use "- Unterprogramme und Prozeduren.

Die Organisation von Unterprogrammen mit " ret-use " wurde in Kapitel V besprochen. In vielen Fällen wirkt sich bei diesen Unterprogrammen nachteilig aus, daß die Bezeichnungen der Variablen und die Marken dem aufrufenden Hauptprogramm entsprechend gewählt werden müssen. Diese Methode ist außerdem sehr unhandlich, wenn umfangreiche Datenblöcke zu verarbeiten sind. Oft läßt sich eine Umspeicherung nicht vermeiden.

Bei Verwendung von Prozeduren treten diese Nachteile nicht auf. Allerdings sind Prozeduren etwas umständlicher aufzurufen. Prozeduren können unabhängig vom Programm geschrieben und compiliert werden. Die in ihnen verwendeten Variablennamen und Marken sind nur innerhalb der Prozedur definiert.

Für die Variablen und Marken, welche die Prozedur mit dem aufrufenden Programm verbinden, werden innerhalb der Prozedur formale Symbole verwendet. Durch das aufrufende Programm müssen diesen formalen Symbolen die tatsächlichen Namen zugeordnet werden. Prozeduren können von anderen Prozeduren aufgerufen werden. Dieser Vorgang darf mehrfach in die Tiefe gestaffelt werden.

Prozedur-Operatoren

Eine Prozedur - z.B. mit dem Namen " NAME " - wird durch eine "call"-Anweisung aufgerufen:

```
call'name'arg'ex'arg'y''  (oder  call'name'arg'ex',,'y'')
```

Dem Prozedur-Namen folgt eine Liste der Variablen, welche die tatsächlichen Namen enthält, die für die formalen Symbole einzusetzen sind. Vor jeden Variablennamen ist das Wort " arg " zu setzen. Diese Liste von Variablen wird "tatsächliche Parameterliste" genannt.

Jede Prozedur beginnt mit einer " enter " -Anweisung, die den Namen der Prozedur und die Liste der formalen Parameter enthält, z.B.

```
enter'name'a'b''
```

Die " enter "-Anweisung kennzeichnet den Anfang einer Prozedur und ersetzt in dem gewählten Beispiel die formalen Parameter a und b durch die Variablen ex und y des Hauptprogramms. In einer " enter "-Anweisung dürfen bis zu 31 formale Parameter verwendet werden.

Die Elemente der " arg " - Liste können einfache oder indizierte Variable oder Marken sein. Die Anzahl und der Typ der Variablen in beiden Listen muß gleich sein. Sie dürfen keine Verteiler oder zusammengesetzte Ausdrücke sein, die Operatoren enthalten.

Der Name einer Prozedur darf erst erscheinen, nachdem er in einer " enter " - Anweisung definiert wurde. Die formalen Parameter müssen in der " enter " - Anweisung ohne Indizes angegeben werden. Innerhalb der Prozedur müssen jedoch alle als formale Parameter eingeführten Variablen stets mit Index geschrieben werden. Das bedeutet z.B., daß der formale Parameter " a ", auch wenn er sich nicht auf ein ganzes Feld, sondern nur auf eine einzelne Variable (oder Marke) bezieht, durch a'0' aufzurufen ist.

Der Ausgang aus einer Prozedur wird durch die Anweisung

```
exit''
```

festgelegt. Das Programm läuft mit der Anweisung weiter, die der " call " - Anweisung folgt.

Die Anweisung

```
end''
```

kennzeichnet das Ende des Prozedur-Quellenprogramms. Beim Compilieren dieser Anweisung wird die Tabelle der Marken und Variablen gelöscht.

Prozedur-Quellenprogramm

Das Prozedur-Quellenprogramm besteht aus einer " enter " - Anweisung, den Arbeitsanweisungen der Prozedur, einer " exit " - Anweisung und einer " end " - Anweisung.

Beispiel: Zeilenweises Einlesen der Elemente einer Matrix.

```
enter'mread'mat'zeil'spalt''
prev';'mat'0''
1';'i''
s1'1';'j''
s2'read'mat'i'j''
for'j'step'1'until'spalt'0'rpeat's2''
for'i'step'1'until'zeil'0'rpeat's1''
exit''
end''
wait'
```

Durch das Compilieren der " enter " - Anweisung wird der Name der Prozedur als Marke definiert. In manchen Fällen kann der Name auch als Verteiler benutzt werden. Der Prozedurname darf keinesfalls verwendet werden, bevor er definiert wurde. Das bedeutet, daß die Prozedur B vor der Prozedur A compiliert werden muß, wenn B von A aufgerufen wird. Außerdem müssen alle Prozeduren, die in einem Programm benutzt werden, vor der ersten Anweisung des Hauptprogramms compiliert werden. Die " dim " - Anweisung ist dabei ausgenommen. Prozeduren, die sich direkt oder indirekt selbst aufrufen, sind verboten.

Verwendung des Prozedurnamens als indizierte Größe

Das Feld von Speicherplätzen unmittelbar vor einer " enter "- Anweisung kann durch eine spezielle Anweisung reserviert und unter dem Namen der Prozedur, versehen mit einem Index, aufgerufen werden. Diese spezielle Anweisung besteht aus den Ausdrücken " stop " und " use'0' ", es darf auch nur " stop " und nur " use'0' " verwendet werden. Jeder Ausdruck reserviert einen Speicherplatz. Die Speicherplätze werden, beginnend mit 1, von der " enter " - Anweisung aus nach vorn gezählt. Außer der Ziffer für den Index muß der Operator " // " angegeben werden (s. Seite 22). Wenn ein " use'0' " verwendet wird, ist der zugehörige indizierte Prozedurname ein Verteiler, im anderen Fall ist er eine Variable. Wenn ein Speicherplatz im Feld eines Prozedurnamens ein " use's4' " enthält, ist der zugehörige indizierte Prozedurname eine Marke.

Variable im Feld des Prozedurnamens sind sehr nützlich, um einzelne Daten vom Hauptprogramm auf die Prozedur und umgekehrt zu übertragen. Dieser Weg ist im allgemeinen hinsichtlich der Rechenzeit und des Speicherplatzbedarfs günstiger, als einzelne Daten über die Parameterliste einzugeben.

Verteiler im Feld des Prozedurnamens können für verschiedene Aus- und Eingänge der Prozedur verwendet werden. Der erste Eingang in eine Prozedur muß allerdings über das " call-enter " - System gehen.

Wenn der Name einer Prozedur ohne Index als Marke benutzt wird, bezieht sie sich auf den normalen Ausgang (exit) der Prozedur. Die " exit " - Anweisung in der Prozedur " name " ist äquivalent dem Ausdruck use'name'', " name " ist als Marke anzusehen und enthält die Rückkehradresse der Prozedur in das Hauptprogramm. Die Anweisung

call'name'//'2''

ist äquivalent mit

ret'name'//'2'use'name'//'1''

Anwendungsbeispiel: Angenommen name'//'1' und name'//'2' seien zwei Verteiler in dem Feld des Prozedurnamens, dann überträgt die Anweisung

call'name'//'1''

innerhalb einer Prozedur den Programmablauf auf die Anweisung, die dem Aufruf call'name'' folgt. Gleichzeitig wird der Verteiler in name'//'1' für eine Rückkehr in die Prozedur gesetzt. Wenn nun im weiteren Ablauf des aufrufenden Programms eine Anweisung

call'name'//'2''

erreicht wird, geht der Programmablauf auf die Prozedur über, beginnend bei der Anweisung, die auf call'name'//'1'' folgt. Der Verteiler name'//'2' wird auf die Rückkehr ins aufrufende Programm gesetzt und zwar auf die dem Aufruf call'name'//'2' folgende Anweisung. Auf diese Weise wird der Block im aufrufenden Programm zwischen call'name'' und call'name'//2''zu einem Unterprogramm, das beliebig oft von der Prozedur aus durch call'name'//'1'' aufgerufen werden kann.

Prozeduraufruf als Operand

Wenn eine Prozedur einen einzelnen Zahlenwert als Ergebnis liefert, so kann dieser Wert beim Ausgang aus der Prozedur im Akkumulator gespeichert werden. Durch Ergänzung der " exit " - Anweisung auf

```
if'a'exit''
```

wobei a für das Ergebnis der Prozedur steht, wird a in den Akkumulator gebracht. Das Ergebnis der letzten Anweisung vor dem Operator " exit " steht beim Ausgang aus der Prozedur ohnehin im Akkumulator.

Falls eine Prozedur einen einzelnen Zahlenwert als Eingabe benötigt, kann dieser in ähnlicher Weise in den Akkumulator gebracht werden. Das geschieht entweder als Ergebnis der Anweisung vor der " call " - Anweisung oder mit dem Operator " if " vor dem Aufruf der Prozedur.

Beispiel für den Prozeduraufruf als Operand:

```
if'z'call'polyn'arg'v1';'v2''
```

Hierbei repräsentiert z einen Gleitkommaausdruck, dessen Wert das Argument eines Polynoms sein soll. Die Koeffizienten des Polynoms stehen in dem Feld v1 und das Ergebnis wird auf dem Speicherplatz v2 abgelegt. Der Wert von z wird durch den Operator " prev " in die Prozedur übertragen.

Prozedur zur Berechnung von Polynomwerten:

```
enter'polyn'a''
prev';'ex''
a'0';'i''
if'a'i''
s1'prev';'y''
i'i-'1';'i'until'1'neg's2''   *)
if'ex'x'y'+'a'i'use's1''
s2'if'y'exit''
end''
wait'
```

Die mit einem Stern gekennzeichnete Anweisung ist wesentlich wirkungsvoller - drei Befehle weniger und weniger Rechenzeit - als die entsprechende Anweisung

```
for'i'step'j'until'1'rpeat's2''
```

mit i = n und j = -1.

Beispiele zur Verwendung von Prozeduren befinden sich im Anhang C.

A n h a n g A

Ganzzahl - und Umwandlungsoperatoren

;'a'	h (a)
a'i+'b'	b (a) a (b)
a'i-'b'	b (a) s (b)
a'ix'b'	b (a) h5336 b (b) r4815 u4200
a'i/'b'	b (a) h5336 b (b) r4815 u4300
a'nx'b'	b (a) n (b) m5952
a'flo'b'	b (a) h5336 b (b) r4815 u4528
a'unflo'b'	b (a) h5336 b (b) r4815 u4414
a'fix'b'	b (a) h5336 b (b) r4815 u4207
a'iabs'b'	b (a) r4815 t5740
a'ipwr'b'	b (a) h5336 b (b) r4815 u4500

Gleitkomma - Operatoren

```
;'a'        h (a)
a'+'b'      b (a) h5336 b (b) r4815 u4600
a'-'b'      b (a) h5336 b (b) r4815 u4835
a'x'b'      b (a) h5336 b (b) r4815 u4807
a'/'b'      b (a) h5336 b (b) r4815 u4707
a'pwr'b'    b (a) h5336 b (b) r4815 u4513
a'x10p'b'   b (a) h5336 b (b) r4815 u4510
0-'a'       b (a) r4815 u5161
abs'a'      b (a) r4815 u4641
sqrt'a'     b (a) r4815 u4203
ln'a'       b (a) r4815 u4210
log'a'      b (a) r4815 u4303
exp'a'      b (a) r4815 u4503
sin'a'      b (a) r4815 u4710
cos'a'      b (a) r4815 u4603
artan'a'    b (a) r4815 u4810
```

Ein- und Ausgabeoperatoren

read'a'	r4815 u4900 h (a)
iread'a'	r4815 u5619 h (a)
a'rdflo'b'	b (a) r4815 u5538 h (b)
rdhex'a'	r4815 u5711 h (a)
aread'a'	r4815 u5606 h (a)
a'print'b'	b (a) h5336 b (b) r4815 u5130
a'iprt'b'	b (a) h5336 b (b) r4815 u5552
a'dprt'b'	b (a) h5336 b (b) r4815 u5201
punch'a'	b (a) r4815 u5137
ipch'a'	b (a) r4815 u6000
hxpch'a'	b (a) r4815 u6037
aprt'a'	b (a) r4815 u5955
rdxit's1'	r5408 u ( ) u (s1)
daprt'cr4'	r5513 u5513 40000002'
daprt'tab6'	r5513 u5513 60000002'
daprt'a'b'c'd'e'f'g'	r5513 u5513 Q45K5594'FFQ00002'

Logische Operatoren

use's1'	u (s1)
if'a'	b (a)
neg's1'	t (s1)
zero's1'	t ( ) s6018 (s1)
pos's1'	m5963 t (s1)
go to's0'	u ( )
set's1'to's2'	r (s1) u ( ) u (s2)
ret's1'	r (s1)
for'a'step'b' until'c'rpeat's1''	b (b) h5739 a (a) h (a) s (c) n5739 s5217 t (s1)
bkp4'use's10''	z0400 u (s10)

A n h a n g B

Liste der Zeichen und Steuerungssymbole

Klein-schaltg.	Groß-schaltg.	"daprt" Code	"aprt" Binär-Code	Klein-schaltg.	Groß-schaltg.	"daprt" Code	"aprt" Binär-Code
a	A	a'	111001	0	)	0'	000010
b	B	b'	000101	1	L	1'	000110
c	C	c'	110101	2	*	2'	001010
d	D	d'	010101	3	"	3'	001110
e	E	e'	100101	4	△	4'	010010
f	F	f'	101010	5	%	5'	010110
g	G	g'	101110	6	$	6'	011010
h	H	h'	110001	7	π	7'	011110
i	I	i'	010001	8	Σ	8'	100010
j	J	j'	110010	9	(	9'	100110
k	K	k'	110110	L e e r t a s t e		'	000011
l	L	l'	000110	-	_	-'	000111
m	M	m'	011101	+	=	+'	001011
n	N	n'	011001	;	:	;'	001111
o	O	o'	100011	/	?	/'	010011
p	P	p'	100001	.	]	.'	010111
q	Q	q'	111010	,	[	,'	011011
r	R	r'	001101				
s	S	s'	111101				
t	T	t'	101101				
u	U	u'	101001				
v	V	v'	011111				
w	W	w'	111110				
x	X	x'	100111				
y	Y	y'	001001				
z	Z	z'	000001				

Steuerungsfunktionen

Umschaltung auf kleine Buchstaben	1c1'	000100
Umschaltung auf große Buchstaben	uc2'	001000
Farbbandumschaltung	color	001100
Wagenrücklauf	cr4'	010000
Rücktaste	bs5'	010100
Bedingter Stop (')	stop'	100000
Tabulator	tab6'	011000
Bedingter Stop**	stopf'	101000
Unbedingter Stop**	stop9'	100100

** nur für Schnellstanzer.

Anhang C

1. Zeilenweise Ein- und Ausgabe von Matrizen:

```
enter'mread'mat'zeil'spalt''
prev';'mat'0''
1';'i''
s1'1';'j''
s2'read'mat'i'j''
for'j'step'1'until'spalt'0'rpeat's2''
for'i'step'1'until'zeil'0'rpeat's1''
exit''
end''
wait'

enter'matex'mat'zeil'spalt''
daprt'cr4'cr4''
0';'kend''1';'i''set's3'to's5'set's10'to's5''
s1'if','kend'i+'6'i-'spalt'0'.'pos's2'use's4''
s2'spalt'0';'kend''
s3'go to's1''
s4'kend'i+'6';'kend''s10'go to's1''
s5'1';'k''
s6'1608'dprt'mat'i'k''
for'k'step'1'until'kend'rpeat's6''
if','spalt'0'i-'kend'.'zero's7'use's8''
s7'daprt'cr4'cr4''0';'kend''set's3'to's5''set's10'to's5''
for'i'step'1'until'zeil'0'rpeat's1''
use's9''
s8'daprt'cr4''kend'i+'1';'k''
set's3'to's6''set's10'to's6''use's1''
s9'exit''
end''
wait'

dim'a'401''
s1'iread'm'iread'n''
call'mread'arg'a'arg'm'arg'n''
call'matex'arg'a'arg'm'arg'n''
stop''
use's1'''
```

```
s000 0618      s001 0600

   n 3540         m 3541         a 4158      matex 0418

mread 0333
```

Datenbeispiel

Eingabe

```
+2'+11'+1'+1'+2'+1'+3'+1'+4'+1'+5'+1'+6'+1'+1'+1'+2'+1'+3'+1'+4'+1'+5'+1'+6'
+1'+7'+1'+8'+1'+9'+1'+10'+2'+11'+2'+13'+2'+14'+2'+15'+2'+16'+2'+17'+2'
```

Ausgabe

```
  .99999994    1.99999992    3.00000016    3.99999980    5.00000004    6.00000028
  .99999994    1.99999992    3.00000016    3.99999980    5.00000004

 6.00000028    6.99999992    8.00000017    8.99999981    9.99999945   11.00000187
12.99999997   13.99999901   14.99999806   16.00000306   17.00000211
```

2. Loesung von linearen Gleichungssystemen:

```
        enter'simeq'array''

        ['['array'0';'n']'i+'1';'n+1']'i+'1';'n+2''
        1';'j''

s1'     j';'m''1';'k''

s2'     j';'h''
        if'k'i-'m'zero's4''
        array'k'h'/'array'm'h';'t''

        h'i+'1';'h''
s3'     array'k'h'-'t'x'array'm'h';'array'k'h''
        h'i+'1';'h'until'n+2'trn's3''

s4'     k'i+'1';'k'until'n+1'trn's2''
        j'i+'1';'j'until'n+1'trn's1''

        1';'m''n'i+'1';'h''

s5'     array'm'h'/'array'm'm';'array'm''
        m'i+'1';'m'until'n+1'trn's5''
        exit''
        end''
        wait'


        dim'a'111''

s1'     iread'a''
        prev'i+'1';'a+1''
        1';'j''

s2'     1';'k''daprt'cr4''

s3'     read'a'j'k''
        for'k'step'1'until'a+1'rpeat's3''
        for'j'step'1'until'a'rpeat's2''
        daprt'cr4'cr4''
```

```
s4'     call'simeq'arg'a''

        1';'k''
s5'     1606'print'a'k''
        for'k'step'1'until'a'rpeat's5''
        stop''use's1'''
```

```
      s000 0607       s001 0511       s002 0518       s003 0523

      s004 0548       s005 0553

        k 4006          j 4007        a+1 4008          a 4155

     simeq 0333
```

Datenbeispiel

Eingabe

```
+4'
-2468925' 02'-1115906' 02'+1593902' 02'+1653652' 01'-6593574' 02'
+3265821' 01'+2068144' 02'-3660459' 01'-2984554' 01'-1108003' 03'
-1865248' 02'+2026418' 02'-2189296' 02'+1714671' 02'-1630398' 03'
-1303823' 02'-7848847' 01'-5361969' 01'+1084467' 02'+7880835' 01'
```

Ausgabe

```
    .499999 e 01   -.600001 e 01   -.800013 e 00    .199998 e 01
```

3. Berechnung von Mittelwert,Varianz,Standardabweichung
und Standardabweichung des Mittelwertes
von IANZ Messwerten mit Hilfe der Prozedur stats '

```
        stop'stop'stop'stop'stop''
        enter'stats''

        0';'sumx';'sumx2''
        1';'i''
s1'     read'ex''
        sumx'+'ex';'sumx''
        sumx2'+'ex'x'ex';'sumx2''
        for'i'step'1'until'stats'//'5'rpeat's1''
        0'flo'['stats'//'5']';'zn''
        sumx'/'zn';'stats'//'4''
        ['sumx2'-'sumx'x'stats'//'4']'/'['zn'-'.9999'99999'e'0']';'stats'//'3''
        sqrt'['stats'//'3']';'stats'//'2''
        prev'/'sqrt'zn';'stats'//'1''
        exit''
        end''wait'
```

```
s1'     iread'ianz''
        prev';'stats'//'5''
        call'stats''
        daprt'cr4'cr4'uc2'm'i't't'e'l'w'e'r't' 'x'm' 'v'a'r'i'a'n'z' 'v' '
        s't'a'n'd'a'r'd'a'b'w'lcl'.' 'uc2's' ' 'u'n'd' 's't'a'n'd'a'r'd'a'
        b'w'lcl'.'uc2'd'lcl'.'uc2'm'i't't'e'l'w'lcl'.' 'uc2's'x'm'cr4'a'
        u's' 'lcl''

        500'iprt'ianz''

        daprt'uc2' 'm'e's's'u'n'g'e'n'cr4'cr4'x'm' '+'lcl''

        1608'print'stats'//'4''
        daprt'cr4'uc2'v' ' '+'lcl''
        1608'print'stats'//'3''
        daprt'uc2' ' ' 's' '+'lcl''
        1608'print'stats'//'2''
        daprt'cr4'uc2's'x'm'+'lcl''
        1608'print'stats'//'1''
        stop'use's1'''
```

```
s000 0603       s001 0460

ianz 4158       stats 0338
```

Zahlenbeispiel. (Erforderlich Funktions-U.P. SQRT)

```
.0000332'

+13'

+442'+1'+447'+1'+470'+1'+472'+1'+453'+1'+455'+1'+460'+1'
+464'+1'+429'+1'+452'+1'+457'+1'+458'+1'+466'+1'
```

```
MITTELWERT XM VARIANZ V STANDARDABW. S  UND STANDARDABW.D.MITTELW. SXM
AUS     13 MESSUNGEN

XM =   .45576918 e 01
V  =   .13896882 e-01   S =   .11788505 e 00
SXM=   .32695425 e-01
```