AUTOCODE

ein vereinfachtes Codiersystem für den

MERCURY

Rechner

Instituto de Cálculo
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

November 1961

VORBEMERKUNG

Wir haben es für sinnvoll erachtet, die Grundregeln der Programmierung im AUTOCODE Codiersystem auf wenigen Seiten zusammenzufassen. Notwendig ist dies, da die Wissenschafts- und Industriezentren ihre eigenen Programme entwickeln müssen. Der Gebrauch des im Instituto de Cálculo der Universidad de Buenos Aires untergebrachten Ferranti – Mercury Rechners soll auf diese Weise erleichtert werden.

Wir fügen zwei Anhänge hinzu, die wir als sehr nützlich erachten. Sie ermöglichen auch jenen, die das konventionelle System nicht kennen, eine höhere Flexibilität.

Unser besonderer Dank gilt Frau Dr. Cicely M. Popplewell für das geduldige und sorgfältige Lesen unseres ersten Entwurfs sowie für ihre vielfältigen Korrekturen. Weiterhin bedanken wir uns bei Frau Dr. R. Ch. de Guber für ihre wertvolle Zusammenarbeit.

Buenos Aires, Oktober 1961

E. García Camerero

Instituto de Cálculo
Facultad der Ciencias
Exactas y Naturales

I. EINLEITUNG.

1. Allgemeines.

Das Autocode-System besteht wie alle automatischen Codiersysteme aus einem Übersetzungsprogramm *(Compiler)*, das eine symbolische Sprache (die Programmiersprache von Autocode), die der mathematischen Sprache sehr ähnelt, in reine Maschinensprache übersetzt. Nach der Übersetzung fängt die Ausführung des eigentlichen Programms an. Dazu gehört u.a. das Lesen der numerischen Daten des Problems und der Ausdruck der Ergebnisse.

Der Programmierer benötigt für Autocode weder genaue Kenntnisse der reinen Maschinensprache noch die genauen funktionellen Details des Mercury. Er muss lediglich die Autocode-Programmiersprache und die Funktionsweise der idealen Maschine kennen, mit der er arbeitet. Dies ermöglicht ihm, seine numerischen Berechnungen anzustellen, ohne auf den Rechner, der sich um die Auflösung kümmert, direkt zugreifen zu müssen.

2. Der Rechner.

Unser idealer Rechner besteht wie alle automatische Digitalrechner aus den folgenden Teilen (Abb. 1):

a) einem Speicher
b) einer arithmetischen Einheit
c) einer Steuereinheit
d) einem Eingabe-/Ausgabegerät

Die arithmetische Einheit unseres “idealen Rechners” ist sehr leistungsfähig, da sie neben den vier grundlegenden Operationen auch eine Reihe komplexer Funktionen durchführen kann.

Die Steuereinheit ist der Kern, der alle die Statements interpretiert und jene ausführt, die eine eindeutige Logik- und Kontrollstruktur besitzen.

Die Eingabe- und Ausgabegeräte erlauben, das Programm und die Daten in die Maschine einzugeben und die Ergebnisse in einem Lochstreifen bzw. ausgedruckt zu bekommen.

Der Speicher ist der Ort, an dem während der Ausführung des Programms, die Daten und die Ergebnisse abgespeichert werden.

Der Autocode-Programmierer muss weder den inneren Aufbau des Eingabe- und Ausgabegeräts noch der Steuer- und arithmetischen Einheiten kennen. Interessant sind für ihn Aspekte des Speichers.

Der Speicher ist der Ort, in dem Zahlen und Befehle gespeichert werden können. Er ist in zahlreiche Felder unterteilt, auf denen jeweils eine Zahl oder ein Befehl gespeichert werden kann. Um sie unterscheiden zu können sind sie nummeriert. Die Nummer entspricht einer Adresse. Folglich wird jedem Feld zugeordnet. Die eine Nummer entspricht der zuordnenden Adresse, die andere bestimmt den Inhalt des Feldes, der sowohl ein Befehl als auch ein numerischer Wert sein kann. Ein und dieselbe Adresse kann unterschiedliche Zahlen zu unterschiedlichen Zeitpunkte haben. Die Adressen wiederum nehmen eine ähnliche Rolle ein, wie die Variablen in der Algebra. Ebenso wie wir in der Algebra mit Buchstaben operieren, die, wenn wir numerische Ergebnisse suchen, durch Werte ersetzt werden, operieren wir mit den Adressen, wobei die Befehle deren Inhalte übernehmen.

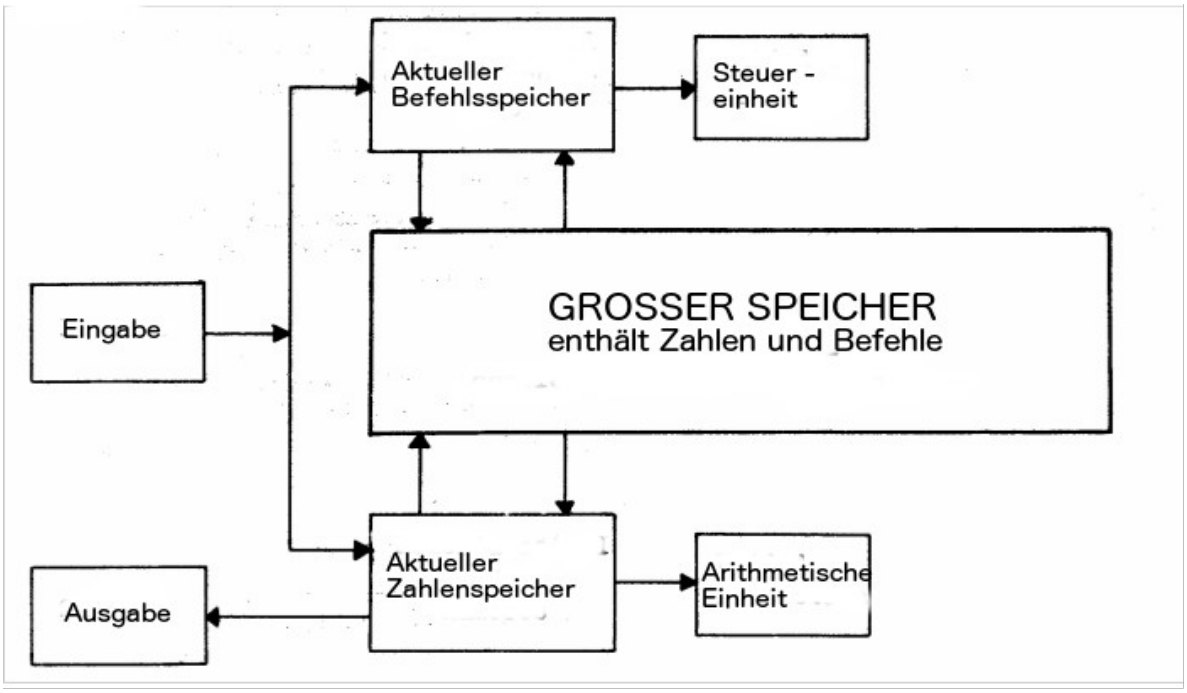


Abb. 1. – Funktionelles Diagramm des MERCURY-Rechners für die Wirkungsweise des AUTOCODE-Autoprogrammierungssystems

Um in Autocode zu programmieren, setzen wir voraus, dass der Speicher in drei Teile geteilt ist oder dass es drei unterschiedliche Speicher gibt: der aktuelle, in zwei Teile geteilte Speicher (ein Teil für Befehle, der andere für Zahlen) und der große Speicher.

Der aktuelle Befehlsspeicher setzt sich aus Ferritkernen zusammen, auf die sehr schnell zugegriffen werden kann. Wenn ein Programm sehr groß ist, muss es in sogenannte Kapitel aufgeteilt werden, deren Größe auf die Kapazität des aktuellen Befehlsspeichers begrenzt ist. Der aktuelle Zahlenspeicher, ebenfalls aus Ferritkernen, hat eine Kapazität von 521 Variablen, wie in Abb. 1 dargestellt ist (s. die Abschnitte über Variablen und Indizes).

Der große Speicher, der aus einer magnetischen Trommel mit langsamen Zugriff besteht, hat eine Kapazität, die den aktuellen Speicher mehrfach enthalten kann.

Nur die im aktuellen Speicher abgespeicherten Befehle werden ausgeführt. Deshalb ist es für Programme mit mehr als einem Kapitel notwendig, die Methode des schrittweisen Übertragens des Kapitels unseres Programms vom großen Speicher in den aktuellen Speicher unter Beachtung der dazugehörenden Variablen und Kostanten zu verwenden. Das rechtfertigt den Namen, den wir dem aktuellen Speicher gegeben haben.

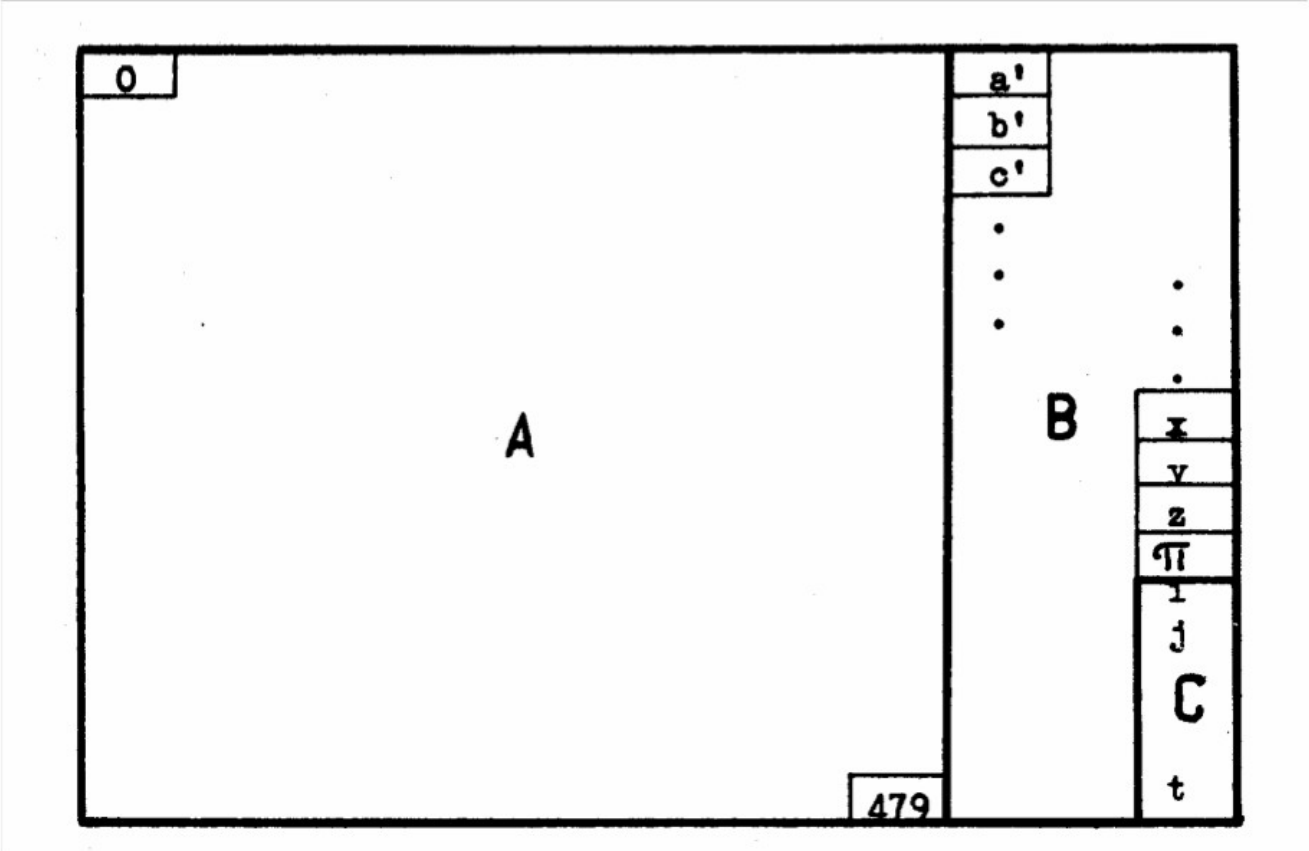


Abb. 2. – Aktueller Zahlenspeicher. Mit A, B und C sind die Bereiche für die Sondervariablen und Indizes gekennzeichnet (siehe genauer in Abb. 1).

3. Die Autocode Sprache.

Die Autocode Sprache besteht aus Statements. Der Rechner übersetzt die in jedem Statement enthaltenen Befehle und setzt diese, sobald das ganze Programm übersetzt ist, um. Die Statements bestehen aus einem Betriebsteil und einigen Operanden. Diese Statements, konkrete grammatikalische Sätze der Sprache mit der wir arbeiten, sind unterschiedlicher Art und sie bestehen aus unterschiedlichen Teilen.

Bezüglich ihrer Betriebsfunktion können sie folgendermaßen klassifiziert werden:
- Arithmetische Befehle
- Steuerbefehle
- Ein- und Ausgabebefehle
- Direktiven

Die grundlegende Teile eines Statements sind:
- Variablen
- Indizes
- Zahlen
- arithmetische Zeichen

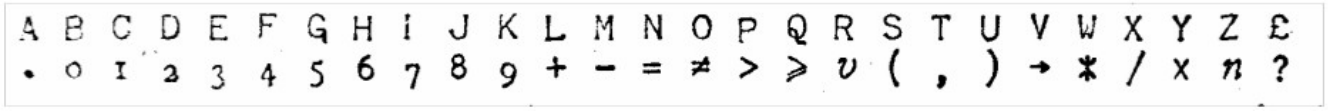
- Sonderzeichen
- Wörter

Die Grundelemente sind in allen Fällen Buchstaben, Ziffern und Zeichen; die Gesamtheit dieser Elemente bildet unser Alphabet, das das folgende ist:

a b c d e f g h I j k l m n o p q r s t u v w x y z π
0 1 2 3 4 5 6 7 8 9 + = ≠ > ≥ ≈ (,) → * / Ψ ' ?

Wir möchten darauf hinweisen, dass ebenso wie die Syntax unserer Sprache, also die Gesamtheit der Regeln, die wir befolgen müssen, wenn wir unserer Statements schreiben, auch die Orthographie wichtig ist. Daraus folgt, dass wir jeden Teil des Statements in seiner korrekten Form schreiben müssen, da das Auswechseln eines Buchstabens durch einen anderen oder das Auslassen eines Zeichen unseres Alphabets, sich von den gewünschten unterscheidende Ergebnisse oder aber das Anhalten der Maschine verursachen können, da diese das Statement nicht interpretieren kann.

HINWEIS: Die Zeichen, die zurzeit vom Instituto de Cálculo de la Universidad de Buenos Aires installiert sind, sind Folgende:



Zu beachten ist die spezielle Darstellung der Zeichen π ≈ ' Ψ.

II. ELEMENTE DER STATEMENTS

1. Variablen.

Wie schon ausgeführt, können die Felder des Speichers verschiedene Zahlen zu unterschiedlichen Zeitpunkten abspeichern und die Adressen dieser Felder spielen eine ähnliche Rolle wie die Variablen in der Algebra. Aus diesem Grund nennen wir in der Autocode-Sprache die Adressen der Felder, in denen man die verschiedenen Zahlen, die wir beim Rechnen benutzen, abspeichern kann, Variablen. Es gibt zwei Variablentypen: aktuelle Variablen und Hilfsvariablen. Erstere belegen Plätze des aktuellen Speichers und gehen bis Platz 509; letztere befinden sich im großen Speicher und reichen bis zu Platz 10.752.

1.1 Aktuelle Variablen.

Die 509 Variablen, die zu unserer Verfügung stehen, sind in zwei Gruppen geteilt: Haupt- und Sondervariablen.

a) Hauptvariablen.
Es gibt 480 Hauptvariablen, die durch einen der folgenden Buchstaben bestimmt werden können

a b c d e f g h u v w x y z π

ergänzt mit einem Subindex.
Beispiele: a_0 v_j h_{32} z_r

→

Bevor man die Hauptvariablen im Programm benutzt, müssen die Felder des Speichers reserviert werden. Das wird mit dem Symbol → folgendermaßen gemacht:

a → α
b → β
c → γ
……..

Damit werden die Speicherstellen 0 bis α für die Variablen a a_0 a_1 … a_α, die Stellen α + 1 bis α + β + 1 für die Variablen b_0 b_1 … b_β sowie die Stellen α + β + 2 bis α + β + γ + 2 für die Variablen c_0 c_1 … c_γ usw. reserviert.

Beispiele: Die Ausdrücke

x → 100
y → 36
z → 1

reservieren die Speicherstellen von 0 bis 100 für die Variablen x_0, x_1 … x_{100}, die Speicherstelle von 101 bis 137 für die Variablen y_1, y_2 … y_{36} und die Speicherstelle 138 und 139 für die Variablen z_0 und z_1. Wenn wir nur eine Variable benötigen, benutzen wir die Sondervariablen, die im folgenden Abschnitt behandelt werden.

b) Sondervariablen.
Es sind 29 Variablen, die durch folgende Buchstaben ohne Subindex dargestellt werden:

a' b' c' d' e ' f' g' h' u' v' w' x' y' z'
a b c d e f g h u v w x y z π

die den festen Stellen zwischen 480 und 508 des aktuellen Speichers umfassen. An der Stelle π=508 ist, solange man es nicht verändert, die Zahl 3.141 592... abgespeichert.
Die Variablen mit Akzent werden normalerweise für Sonderfälle reserviert. Es gibt keine Variable π'.

1.2. Hilfsvariablen.

Wenn die 509 Variablen, die wir im aktuellen Speicher zur Verfügung haben, nicht ausreichen, ist es notwendig, die Hilfsvariablen, bestimmte Stellen des großen Speichers, zu benutzen. Diese Variablen müssen in den aktuellen Speicher zu ihrer Anwendung verschoben werden. Jeder dieser Stellen wird

eine Nummer zwischen 0 und 10571 zugewiesen, die ihrer Adresse entspricht. Man kann nicht immer die 10752 verfügbaren Felder oder Hilfsvariablen benutzen, da die letzten 512n Felder von den Kapiteln 1, 2, 3...n unseres Programms besetzt sind. (s. IV.3 und Anhang 1).

2. Indizes.

Sie sind die Adressen der 12 letzten Stellen des aktuellen Zahlenspeichers. Wir markieren sie mit den Buchstaben

i j k l m n o p q r s t

Diese Buchstaben können nur ganze Zahlen, die zum Intervall (-512, 511) gehören, aufnehmen. Sie werden hauptsächlich als Subindex verwendet.
Beispiel: Haben wir die Variable a_i und der Inhalt des Registers i = 3, entspricht die vorherige Variable a_3.

3. Zahlen.

Die Zahlen sind der Rohstoff, durch die die Befehle wirken und über die diese in die von den Variablen reservierten Felder eingefügt oder explizit im Programm ausgedrückt werden. Die Zahlen, die explizit im Programm beinhaltet sind, nennen wir Konstanten und die, die während des Rechenvorgangs gelesen werden müssen, nennen wir Daten. Die Konstanten füllen Felder des aktuellen Befehlsspeichers aus. Der aktuelle Zahlenspeicher ist für die Daten, die Teilergebnisse und für die Lösungen vor ihrem Ausdruck reserviert. Obwohl der Rechner immer mit Gleitkommazahlen im Intervall $10^{-70} < | X | < 10^{+70}$ arbeitet, können die Daten als Festkommazahlen gelesen werden und werden fast immer auf dieser Weise ausgedruckt. Die Konstanten müssen immer als Festkomma geschrieben werden.

3.1 Festkomma.

<ZEICHEN> <VORKOMMATEIL> <KOMMA> <NACHKOMMATEIL>

Die Darstellung der Festkommazahl ist die übliche der Dezimalzahlen. Das heißt, sie bestehen aus dem Zeichen, den Vorkommastellen, dem Komma und den Dezimalstellen. In Autocode ist es möglich, das Pluszeichen, führende Nullen und das Komma einer Ganzzahl wegzulassen. Vor dem Komma kann es eine beliebige Anzahl von Stellen geben, aber der Nachkommateil darf maximal 24 Stellen enthalten. Der Rechner operiert in allen Fällen mit den zehn signifikantesten Ziffern.

Beispiele: +3.570 oder 3.57
-0.327 oder -3.27
+321.00 oder 321

3.2 Gleitkommazahl.

<MANTISSE> <KOMMA> <EXPONENT>

Die Gleitkommazahl ist die Kurzschreibweise von Ausdrücken der Art

$\pm\alpha \times 10^{\pm\beta}$

(α wird Mantisse und β Exponent genannt), die einfach durch

$\pm\alpha , \pm\beta$

dargestellt werden.
Der Exponent muss eine Ganzzahl und im Bereich -128 bis 127 liegen. In allen Fällen muss er sich auf den folgenden Bereich erstrecken:
2-256<X<.. 2256
Beispiele: -27, 3 drückt -27×10^3 oder -27000 aus
+ 0.5, -2 drückt 5×10^{-3} oder 0.005 aus

4. Arithmetische Zeichen.

Um die vier arithmetische Grundrechenarten auszudrücken, benutzt man folgende Zeichen oder Konventionen:

a) Produkt <Aneinanderreihung der Faktoren>
Um das Produkt mehrerer Faktoren auszudrücken, reiht man die unterschiedlichen Faktoren, die an dem Produkt beteiligt sind, aneinander, ohne dass sie durch ein Zeichen getrennt sind. Die Faktoren können Variablen, Indizes oder Konstanten, aber kein algebraischer Ausdruck sein.

Beispiele: 3abb 4.2xy $x_i x_i$ $u_i v_j wk$
5ay $2iz_j$ 6jz $6z_j$
ijk $mna_m b_n$

Folgende Ausdrücke sind nicht zulässig:

a (b + c)

Ausdrücke der Art 6zj werden, wie nachfolgend aufgezeigt wird, als $6z_j$ interpretiert.

b) Addition

\- +

Die Zeichen + oder – zwischen zwei Elementen bewirken die Addition oder Subtraktion dieser Elemente. Die Elemente können Variablen, Indizes und Konstanten, oder das Produkt mehrerer von ihnen sein.

Beispiele: $3+v_i$ x+j+k-4.5 jk + pq
ab+2 a'-b'b' $2x_i+.25y_i-327.6$

/

c) Quotient

Das Zeichen / zwischen dem Dividend und dem Divisor ergibt den Quotient des ersten durch den zweiten. Der Dividend muss eine Variable, ein Index, eine Konstante oder das Produkt von einigen von ihnen sein, aber kein Ausdruck, bei dem die Zeichen + oder – beteiligt sind. Der Divisor muss eine Variable, ein Index oder eine Konstante sein, aber kein Produkt oder Summe davon.

Beispiele:	a/b	3xy/z	$3if_k/f_j$
	5.21/m	v/q	a'b'/c'

Ausdrücke wie

a+b/c+d

werden durch das Autocode-System folgendermaßen interpretiert:

a+(b/c)+d.

5. Sonderzeichen und Wörter.

Außer den bis hierhin betrachteten Teilen der Statements existieren Sonderzeichen und Wörter, deren Aufgabe es ist, den Ausdruck des Statements zu erleichtern. Obwohl ihr Gebrauch fast immer der mathematischen Sprache ähnelt, ist das nicht immer der Fall. Z.B. bilden mehrere Summanden in Klammern nie einem Faktor, das Komma in einer Zahl stellt nie die Trennung zwischen Vorkomma- und Dezimalteil dar und der Punkt drückt nie ein Produkt aus. Ein Wort kann ein ganzes Statement ausdrücken, wie bei einigen Steuerbefehlen, Direktiven oder Namen einer Funktion.

Mercury-Konsole, in der sich das fotoelektrische Lesegerät und der Ausgabelocher erkennen lassen. Die Ergebnisse können in einem Fernschreiber gedruckt werden. Er liegt auf einem Schreibtisch, der rechts daneben steht.

III: DIE STATEMENTS

Die Statements der Autocode-Sprache geben den im vorhergehenden Absatz erläuterten Symbolen Sinn. Letztere allein (außer im Fall der Schlüsselwörter) haben keinerlei Bedeutung für den Rechner. Laut ihrer Aufgabe im Programm können die Statements (wie schon darauf hingewiesen wurde) so eingeordnet werden:
Arithmetische Befehle, Steuerbefehle, Ein-/Ausgabebefehle und Direktiven.

1. Arithmetische Befehle.

Die arithmetischen Befehle sind durch das Zeichnen = (≈) gekennzeichnet, das den erhaltenen bzw. enthaltenden Wert des rechten Elementes in der Adresse des linken Elementes abspeichert. Das erste Element kann nur eine Sonder- bzw. eine allgemeine Variable oder ein Index sein. Jeder dieser beiden Fälle bestimmt die Form des rechten Elementes. Ein Statement kann bis 68 Zeichen (inkl. Leerzeichen) lang sein.

1.1. Das erste Element ist eine Variable.

Wenn das erste Element eine Variable ist, kann das zweite Element eine der folgenden Formen annehmen:

a) das zweite Element ist ein algebraischer Ausdruck, an dem sich die vier arithmetischen Grundoperationen zwischen Variablen, Indizes und Konstanten beteiligen können.

Beispiele: $y = 2x+3.5z/2-i$
$w_s = 3.5jx-4.6v_0\ w_i/s-.03272+z_{(s-2)}+i$

Es ist zu anzumerken, dass ein Index oder eine ganze Zahl, die hinter einer Variable stehen, als Subindex zu betrachten sind. Unser Rechner liest nämlich alle Zeichen auf der gleichen Ebene. Deshalb wird das zweite vorherige Beispiel folgendermaßen im Rechner geschrieben:

```
ws =3.5jx-4.6v0 wi/s-.03272+z(s-2)+i
```

Dieses Beispiel zeigt die Notwendigkeit der Klammern bei arithmetischen Operationen mit Subindex. Es ist empfehlenswert, die Faktoren jedes Elementes eines algebraischen Ausdrucks in der folgenden Reihenfolge zu schreiben: Konstante, Indizes, Variablen.

b) das zweite Element ist eine einzelne der folgenden Funktionen einer Variable:

$y = \psi\ sqrt(x)$	$x > 0$
$y = \psi\ sin(x)$	x in rad
$y = \psi\ cos(x)$	x in rad
$y = \psi\ tan(x)$	x in rad
$y = \psi\ exp(x)$	e^x $x < 177$
$y = \psi\ log(x)$	$\log_e x$ $x > 0$

y = ψ mod(x)	|x|
y = ψ int pt(x)	Ganzteil von x
y = ψ fr pt(x)	Bruchteil von x
y = ψ sign(x)	y = ±1
y = ψ poly(x) a_0, n	a_0 ist die Addresse des ersten Koeffizientes und n der Grad des Polynom
y = ψ parity(n)	$y = (-1)^n$

Das Argument vorherigen Funktionen kann eine Variable, ein Index, eine Konstante oder es kann ein algebraischer Ausdruck unter ihnen sein. Eine Ausnahme ist die Parity-Funktion, deren Argument ein Index oder ein ganzer algebraischer Ausdruck zwischen Indizes und Ganzzahlen sein muss.

Beispiele: y = ψ sqrt(xi + 2yj)
y = ψ sin(π z/180)

c) Das zweite Element ist eine der nachstehenden Funktionen mit zwei Variablen:

z = ψ divide(x,y)	x/y
z = ψ arctan(x,y)	arctg(y/x)
z = ψ radius(x,y)	$\sqrt{x^2+y^2}$

beide Argumente können Variablen, Indizes, Konstanten oder auch algebraische Ausdrücke sein.

Beispiele: z = ψ divide(3ixj + 2iyj, 5i/j)
z = ψ radius(a + b, 6.28)

Nochmals möchten wir hier die Wichtigkeit der korrekten Schreibweise in Autocode betonen. Insbesondere für die Funktionen ist es erforderlich, den griechischen Buchstaben ψ direkt hinter das Zeichen = zu setzen, die Funktionen mit dem gleichen Buchstaben in der Liste zu benennen, sowie das Argument in Klammern setzen.
Es ist zu anzumerken, dass unser Rechner bei allen arithmetischen Befehlen, die durch das Zeichen = gekennzeichnet sind und deren erstes Element eine Variable ist, die Ergebnisse der vier Grundrechenarten der Arithmetik (Addition, Subtraktion, Multiplikation, und Division) rundet. Das Gleiche passiert beim Lesen der nicht ganzen Zahlen. Die Grundoperationen werden ohne Runden ausgeführt, wenn das Zeichen ≈ statt = in einer der oben genannten Befehle benutzt wird.
Es ist zu berücksichtigen, dass weder die Funktionen y = ψ int pt(x), y = ψ fr pt(x), y = ψ mod(x), y = ψ parity(n) und y = ψ sign(x) noch Ausdrücke der Art y = a jemals gerundet werden. Alle anderen Funktionen werden immer gerundet. Wenn das Argument der Funktion ein algebraischer Ausdruck ist, hängt das Runden davon ab, ob das Zeichen = oder ≈ benutzt wird.

1.2. Das erste Element ist ein Index.

Falls das erste Element ein Index ist, kann das zweite eine der folgenden Formen annehmen:

a) ein ganzes Polynom von Indizes und ganzen Zahlen im Intervall (-512, 511)

Beispiele: i = 2j + j
m = rs – st
j = i + 2

Folgende Ausdrücke sind nicht zulässig:

i = v + w
j = i + 3.121
k = m/n

b) eine der nachstehenden Funktionen

i = ψ int pt(x)	i-Ganzteil von x
i = ψ max(x_0,m,n)	m<n
i = ψ min(x_0,m,n)	m<n

Die x im ersten Beispiel kann irgendeine Variable oder ein algebraischer Ausdruck aus Variablen, Indizes und Konstanten sein. Im zweiten und dritten Fall drückt i den Subindex der größten oder kleinsten Zahl aus der Folge $x_m x_{m+1}...x...x_n$ aus, die aus $x_0 x_1...x_p$, $p \geq n$ entnommen ist. Falls es mehrere Maxima und Minima gibt, wird das mit dem kleinsten Index ausgewählt.

Beispiele: i = ψ int pt(3x – y)
i = ψ min(v_0, 7, 21)

In arithmetischen Befehlen mit Index als erstem Element wird nie gerundet.

2. Steuerbefehle.

Die Steuerbefehle steuern den Rechenverlauf, indem sie den logischen Ablauf des Programms und die Datenein- und -ausgabe organisieren oder die externe Steuerung des Programmablaufs ermöglichen. Dieser Abschnitt wird in fünf Teile geteilt: Schleifen, Sprünge, Sprünge zwischen Kapiteln, Datenein- und -ausgabe, externe Steuerbefehle.

2.1. Schleifen.

Anweisungsblöcke, die während des Berechnungsverfahrens mehrmals mit verschiedenen Daten wiederholt werden müssen, werden Schleifen genannt. Um eine Schleife auszuführen, besitzt das Autocode-System folgendes untrennbares Befehlspaar:

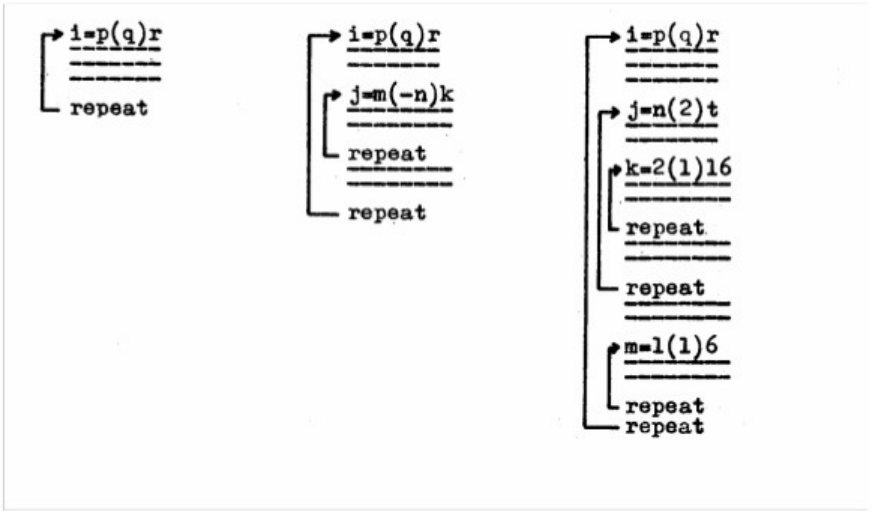
```
i = p (q) r
repeat
```

Der erste Befehl, der der “Kopf der Schleife” genannt werden kann, kann eine der folgenden Formen annehmen:

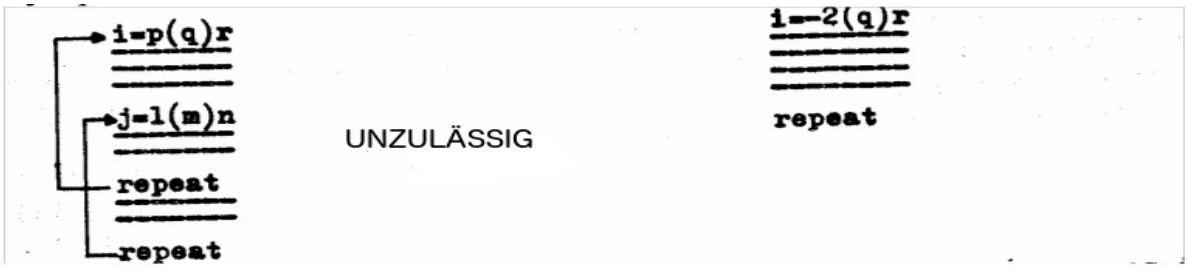
i = p (q) r oder i = p (-q) r

und zeigt, dass die Schleife, die im nächsten Befehl anfängt und mit dem Befehl repeat beendet, dadurch ausgeführt wird, dass dem Index i der Anfangswert p und der Endwert r gegeben wird, vom einem zum anderen über den Schritt q oder auch –q gehend. Wenn der Wert r erreicht wird, wird die Anweisung repeat ignoriert und das Programm läuft sequenziell. Die Buchstaben p, q, r können Indizes oder positive Ganzzahlen im Rang (0, 511) sein. Es ist sicher zu stellen, dass r – p ein Vielfaches von q ist. Die Zahlen p, q, r müssen positiv sein.

Innerhalb einer Schleife können andere Schleifen vorliegen, mit der Einschränkung, dass ein bestimmter Befehl maximal in acht Schleifen enthalten sein darf. Das bedeutet, dass nicht mehr als acht Schleifen geschachtelt werden können. Man muss auch beachten, dass jeder Kopf der Schleife zu einem einzigen repeat gehört. Die folgenden Schemata stellen die verschiedenen Fälle dar:



Wie man sehen kann wird im letzten Beispiel das Wort repeat zwei Mal verwendet, obwohl zwei Schleifen mit dem gleichen Befehl enden. Jeder Kopf der Schleife braucht sein jeweiliges repeat und umgekehrt. Folgende Beispiele sind unzulässig:



2.2. Sprünge.

Wenn man die Reihenfolge des Programms unterbricht, d. h. an einem bestimmten Punkt mit einem anderen als dem nachfolgenden Befehl fortsetzt, hat man einen Sprung gemacht. Zum Springen braucht man einen Sprungbefehl und eine Marke, die den Zielbefehl anzeigt. In diesem Abschnitt werden die Marken und die drei Typen von Sprungbefehle erklärt: unbedingter Sprung, bedingter Sprung und

mehrfacher Pfad.

a) Marken

n)

Wenn eine Anweisung durch einen anderen als den sequenziellen Pfad erreichbar ist, muss sie markiert werden. Marken sind positive Ganzzahlen im Rang (1, 127), gefolgt von einer geschlossenen Klammer. Sie (Nummer und Klammer) werden vor dem Befehl platziert.

Beispiele:
3) y=x+5b
32) i=6
14) z-radius(x,y)

b) Unbedingter Sprung

jump n

Der Sprungbefehl jump n, in dem n eine positive ganze Zahl kleiner als 128 ist, bewirkt den Sprung zur nummerierten Anweisung mit der Marke n). Dieser Sprung kann ohne Unterschied nach vorne oder nach hinten ausgeführt werden.

Beispiele:	jump 3	4)-------
	---------	---------
	---------	---------
	3)------	jump 4

c) Bedingter Sprung

jump n,α,σ,ß

Diese Anweisung bewirkt den Sprung auf die mit der Marke n) gekennzeichnete Anweisung, vorausgesetzt, dass die Beziehung σ zwischen α und β existiert.
Der Buchstabe σ drückt irgendeines der Symbole = ≠ > ≥ aus. Die Buchstaben α und β können beide Variablen, beide Indizes sowie Variable und Konstante, oder Index und Ganzzahl sein. Es ist jedoch unmöglich, eine Variable mit einem Index direkt zu vergleichen.

Beispiele:	jump 3,	$a' > a_i$
	jump 127,	$b \neq i$
	jump 42,	$a_i > 0.01$
	jump 71,	$0.001 = a_i$
	jump 26,	$n > k$
	jump 5,	$n \neq 1$
	jump 1,	$3 \geq k$
	jump 102,	$25 = n$

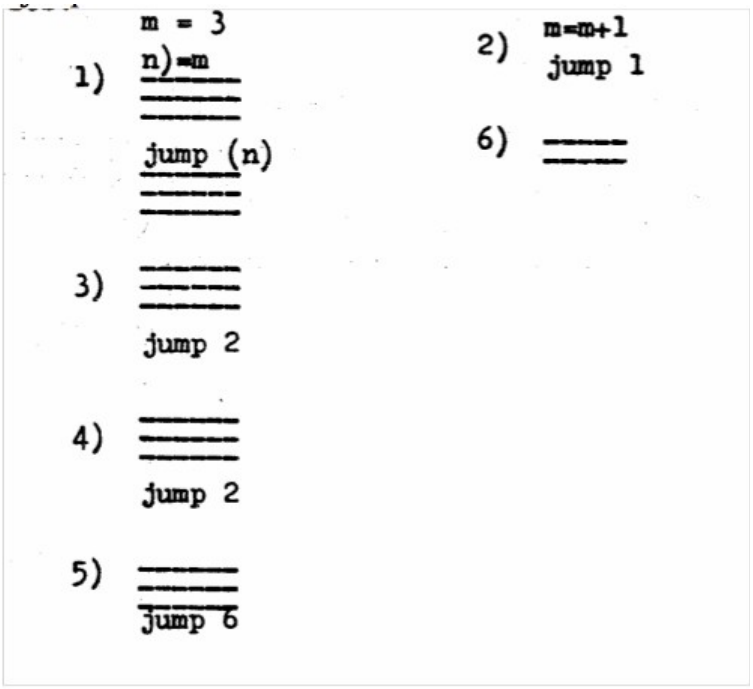
Dabei ist anzumerken, dass es nicht sinnvoll ist, einen Sprung durch den Ausdruck = oder ≠ zwischen zwei Variablen zu bedingen, da es sehr unwahrscheinlich ist, dass diese Gleichheit exakt während eines Berechnungsverfahrens vorkommt.

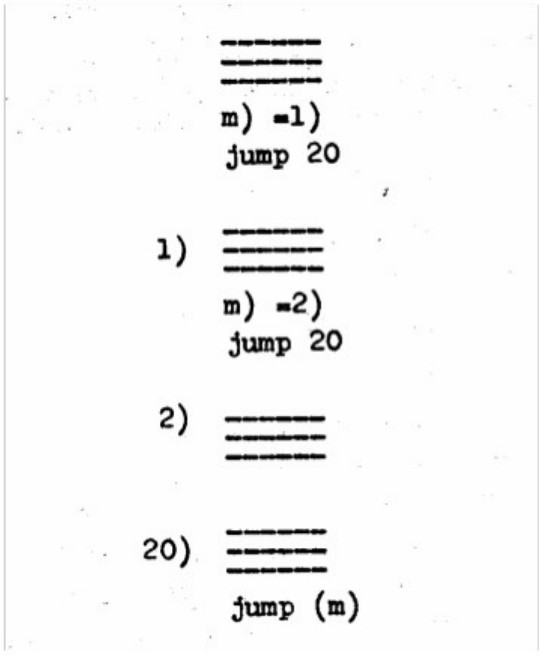
d) Mehrfacher Pfad

n) = m)
jump (n)

Der Sprungbefehl jump (n) erzeugt einen bedingten Sprung mit dem Markenwert n) (n ist immer ein Index), der zuvor mit dem Befehl n) = m) (m ist eine Ganzzahl) definiert worden sein muss.
Da der Befehl n) = m) sehr langsam ist (17 ms), sollte er nur in unerlässlichen Fälle verwendet werden.
Der Befehl n) = Ganzzahl) ist viel schneller (120 µs).

Beispiele:





2.3 Sprung zwischen Kapiteln.

Um vor dem Abschluss eines Kapitels zu einem Befehl in einem anderen Kapitel zu springen, muss man folgende spezielle Steueranweisungen benutzen: across m/c oder das untrennbare Paar down m/c, up.

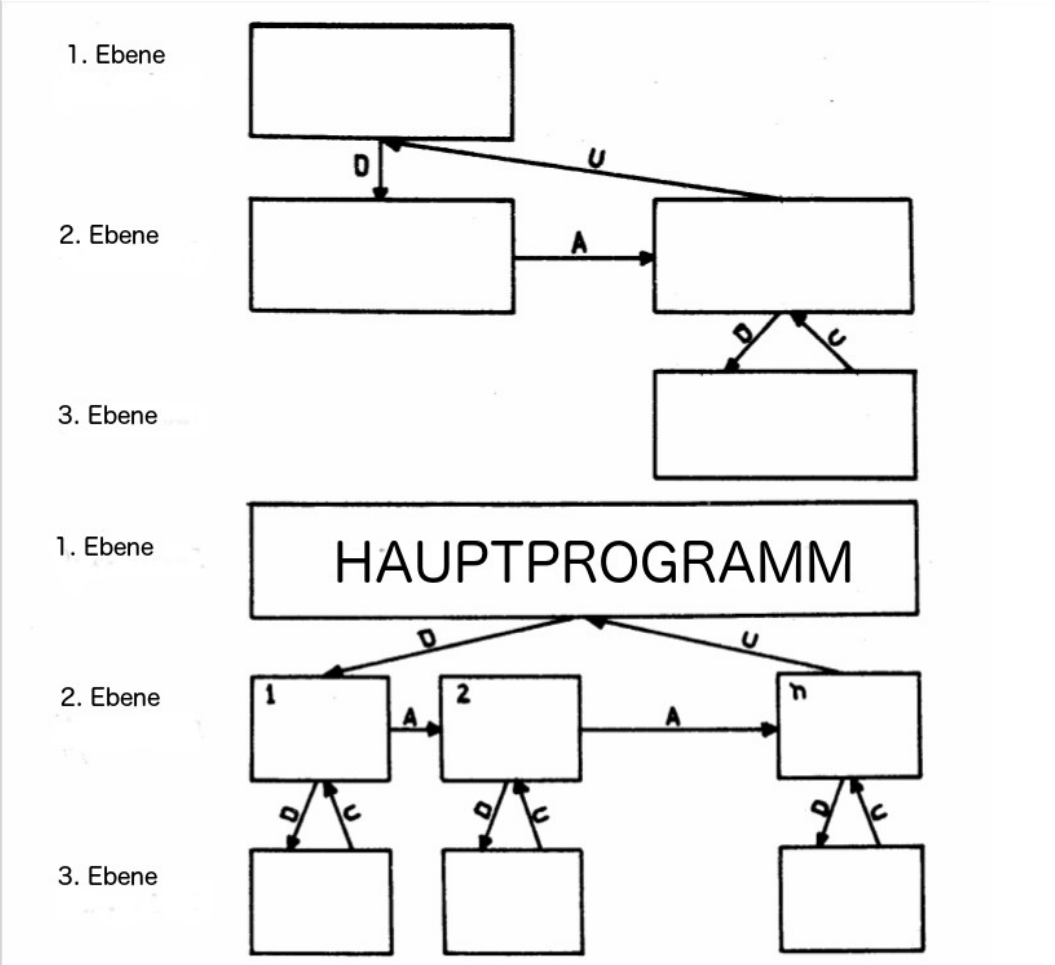
a) Hin (einfache Änderung) across m/c

Diese Anweisung setzt die Berechnung in der Anweisung der im Kapitel c) enthaltenen Marke m fort (s. Abschnitt Direktive). Dafür legt sie das Kapitel c im aktuellen Speicher ab und beginnt mit der Programmausführung auf der Marke m.

b) Rückkehrbefehl (Änderung mit Rückkehr) down m/c up

Sie sind zwei untrennbare Anweisungen. Die erste hat die gleiche Wirkung wie across m/c mit der Besonderheit, dass, wenn bei der Programmausführung die Anweisung up vorgefunden wird, automatisch die Anweisung down m/c (von der sie ausging) folgt.

Die folgenden Diagramme zeigen einige Möglichkeiten:



3. Externe Steuerbefehle.

So wird eine Reihe von Anweisungen genannt, deren Zweck ist, beim Steuern des Berechnungsverfahren durch Tonsignale zu helfen. Einige zeigen ungefähr an, in welchem Teil des Programms man sich befindet. Andere weisen darauf hin, wenn ein manueller Befehl notwendig ist oder wenn die Berechnungen abgeschlossen wurden. Die hier Beschriebenen sind: Hoot, Halt, End.

a) Signal

Hoot

Dieser Befehl erzeugt über den Lautsprecher ein Tonsignal, das eine Sekunde dauert. Auf diese Weise kann man ein Berechnungsverfahren markieren und weiß jederzeit, ob das Programm korrekt funktioniert.

b) Halt

Halt

Dieser Befehl bewirkt eine Schleife, die aus folgenden Teile besteht:

- Erzeugung des Tonsignales a
- Erzeugung des Tonsignales b
- Von der Konsole lesen
- Fortsetzung der Schleife, falls eine bestimmte Information in der Konsole fehlt
- Fortsetzung des Berechnungsverfahrens, falls es eine bestimmte Information in der Konsole gibt

Dieser Befehl weist darauf hin, dass ein Punkt erreicht wurde, an dem man manuell eingreifen muss (s. Anhang II).

c) Ende

end

Dieser Befehl am Ende des Programms macht durch ein Tonsignal deutlich, dass die Berechnungen beendet sind und es unmöglich ist, fortzusetzen.

4. Eingabe- und Ausgabebefehle.

Man braucht einige Befehle, um den Dateneingang und -ausgang zu steuern, also die Daten zu speichern oder die Ergebnisse auf einem Papier auszudrucken. Die Hauptbefehle sind read und print. Darüber hinaus gibt es eine andere durch **?** bezeichnete, bedingte Ausgabeanweisung und andere Hilfsanweisungen. newline und space ermöglichen es nämlich, die gedruckten Zahlen in Form einer Tabelle anzuordnen. rmp bietet die Gelegenheit, ein Programm während der Berechnungen zu lesen.

4.1 Eingabe. Es gibt zwei Anweisungen (read (α) und rmp), die das Lesen eines in einem Eingabeelement platzierten Lochstreifens ermöglichen, je nachdem, ob es sich um ein Daten- oder Programmband handelt.

a)Daten lesen

read(α)

Durch diesen Befehl wird die nächste Zahl des Lochstreifens in der Speicherstelle α abgespeichert, wobei α eine Variable oder ein Index sein kann. Diese Zahlen können als Fest- oder Gleitkommazahl gelocht werden, wenn α eine Variable ist. Falls es sich einen Index handelt, muss er als Ganzzahl im Intervall (-512, 511) gelocht werden. Wichtig ist, dass die Zahlen in der erforderten Reihenfolge gelocht werden.

Beispiel: Wenn die Zahl 2.2575 im Eingabeelement steht, bewirkt die Anweisung

read (a_0)

nach einer Umwandlung in eine Gleitkommazahl die Speicherung der Zahl 2.2575 auf der Speicherstelle a_0.

b) “Read more programm” rmp

Der Befehl rmp (read more programm) liest und übersetzt den Programmlochstreifen. Zur automatischen Fortsetzung des Berechnungsverfahrens muss das Programm, das gerade gelesen wurde, ein Kapitel 0 enthalten, der das bestehende Kapitel 0 zerstört.

4.2 Ausgabe. Die Befehlsausgabe ist durch folgende Befehle geregelt:

a) Ergebnisausdruck print(α) m, n

Im Befehl print (α) m, n, sind m und n ganze Zahle oder Indizes und α eine Variable, ein Index, eine Konstante oder einer der algebraischen Ausdrücke. Wenn m ≠ 0 wird α als Festkommazahl mit m Ganzstellen und n Dezimalstellen ausgedruckt. Wenn n = 0 wird der Dezimalteil nicht ausgedruckt. Wenn m = 0, $\alpha \geq 10^{14}$ wird α als Gleitkommazahl mit einem dreistelligen Exponent ausgedruckt. Wenn α ein Index ist, enthält der Ausdruck n Nullen als Dezimalteil. Enthält die Zahl α einen Dezimalteil, wird ihr Wert mit einer Abrundung in der letzten Ziffer ausgedruckt.

Beispiele: Sei a_0 = -3.27721675, wird die Anweisung

print (a_0) 2,5

-3.27722

drucken.

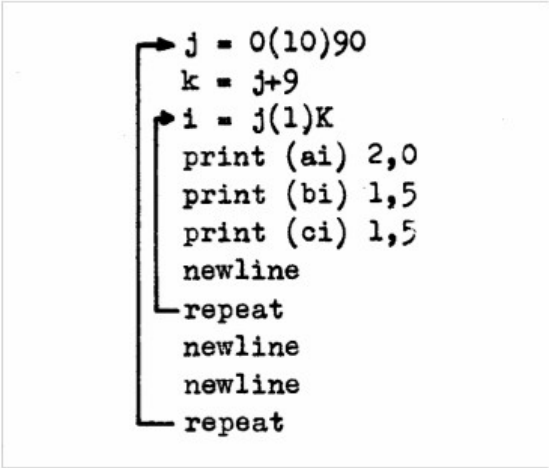
b) Tabs newline space

Durch die Wörter space und newline wird über den Fernschreiber auf dem Schriftstück ein Leerzeichen und einen Zeilenvorschub eingefügt. Man muss bei dem tabellarischen Ausdruck der Ergebnisse darauf beachten, dass die Fernschreibleitung nur 68 Stellen hat und der Rechner automatisch zwei Leerzeichnen nach jeder Zahl lässt. Von daher füllt jede Zahl im allgemeinen Fall m + n + 4 oder m + 3 wenn n = 0 aus. Wenn es als Gleitkommazahl geschrieben werden würde, würden n + 9 Stellen ausgefüllt. Wenn die Anzahl der Ganzstellen der zu druckenden Zahl größer als m ist, werden alle diese Ziffern ausgedruckt und der Dezimalpunkt wird nach rechts verschoben.

c) Teilausdruck ?

Das Zeichen ? druckt am Anfang oder Ende einer arithmetischen Anweisung die berechnete Zahl auf der linken Seite des Fernschreibers, wenn ein Schalter der Konsole sich während der Übersetzung in einer bestimmten Position befindet. Wenn die Zahl größer als 10^{14} ist, wird sie als Gleitkommazahl gedruckt. In allen anderen Fälle wird sie mit zehn Dezimalstellen gedruckt. Die Indizes werden als ganze Zahl ohne Komma oder Dezimalteil gedruckt. Es ist unzulässig, diesen Befehl mit Leseanweisungen zu benutzen. Dieser Befehl wird besonders während eines Programmtests verwendet.

Beispiel: Drucken ai, bi und ci (i= 0 (1) 99) in Zehnerblöcken durch eine Doppelleerzeile getrennt. Die ai sind zweistellige ganze Zahlen, bi und ci sollen mit einer ganzen Zahl und fünf Dezimalstellen vorkommen.



5. Direktiven.

Im Autocode System gibt es Statements, die während des Berechnungsverfahrens keine Wirkung haben. Ihre Aufgaben sind die Organisierung des Programms während der Übersetzung, die Platzierung von verschiedenen Kapiteln und andere Fähigkeiten, wie u. a. das Schreiben des Programmtitels oder das Ausdrucken der Anzahl der zu einem bestimmten Zeitpunkt im schnellen Speicher verfügbaren Felder. Schon bekannt ist eine der gebräuchlichsten Direktiven (durch → dargestellt), die für die Hauptvariablen die entsprechenden Speicherstellen reserviert. Im Folgenden werden die Direktiven title, chapter, close, variables, p s a betrachtet.

a) Titel

title

Die title-Direktive druckt auf das Papier des Fernschreibers die Wörter, Zahlen und Zeichen, die in die nächste Zeile geschrieben werden. Sie wird hauptsächlich benutzt, um den Titel des Programms, oder einige Kommentare dazu an die oberste Stelle der Ergebnisse zu schreiben. Falls der Titel mehr als eine Zeile ausfüllt, soll das Wort title jeder einzelnen Zeile vorangehen.

Beispiel: Wenn man im Programm

```
title
tab coulombs Funktion
title
delta = 0.01     epsilon = 0.01
```

schreibt, erscheint in der Kopfzeile der Tabelle:

```
tab coulombs Funktion
```

delta = 0.01 epsilon = 0.01

b) Kapitel

chapter
close

Das untrennbare Direktivenpaar chapter α und close zeigt den Anfang und das Ende jedes Kapitels, in dem das Programm geteilt wurde, falls es zu lang war, an. Dies ermöglicht die Lokalisierung jedes Kapitels im großen Speicher folgendermaßen: wenn der Rechner chapter α liest, reserviert er einen bestimmten Bereich im großen Speicher; wenn er close liest, befiehlt er das Übertragen des aktuellen Kapitels in die reservierten Speicherstelle. Die Zahl α muss eine Ganzzahl im Bereich (0, 22) sein.
Im besonderen Fall des Kapitels 0 befiehlt die Direktive close nicht nur das aktuelle Kapitel in den großen Speicher zu übertragen, sondern auch dem Programm mit seinem ersten Befehl anzufangen. Deshalb muss dieses Kapitel als Letztes gelesen werden.

c) Reservierung von Variablen

w

Die Direktive → reserviert Bereiche im aktuellen Zahlenspeicher, in denen sich die Hauptvariablen befinden werden. Ihr Gebrauch wurde schon in II-1.1.-a erklärt.

d) Beibehaltung der Reservierung

variables

Wenn ein Kapitel mit den gleichen Variablen wie das vorige im Lochstreifen Kapitel α operiert, ist es nicht notwendig, den Bereich für die Variablen nochmal zu reservieren. Es reicht folglich aus, wenn man die Direktive variables α unter die Direktive des Kapitels schreibt. Es könnte außerdem eine andere neue Variable, die nicht in diesem Kapitel enthalten ist, verwenden. Diese neuen Variablen werden in gewohnter Form durch den Pfeil nach dem Schreiben der variables α reserviert.

e) Verfügbarer Platz

p s a

Aufgrund der unterschiedlichen Länge der Statements ist es unmöglich im Voraus zu wissen, wieviele von ihnen in ein Kapitel hineinpassen. Wenn man aber die genaue Größe des noch zur Verfügung stehenden Datensatzes wissen möchte, kann man die Direktive p s a benutzen. Die genannte Direktive, die an irgendeine Stelle des Programms eingefügt werden kann, druckt die Nummer des Kapitels und die Größe des noch verfügbaren Registers aus. Die maximale Anzahl von verwendbaren Maschinenbefehle pro Kapitel ist 832.

f) Quickies

ψ sqrt
ψ cos
ψ log
ψ tan
ψ radius
ψ sin
ψ exp
ψ arctan

Die vorhergehenden Namen der Funktionen, die vor der Direktive close enthalten sind, werden als Direktiven (Quickies) betrachtet. Sie beinhalten während des Übersetzungsprozesses die Berechnungsroutine dieser Funktionen, falls genügend verfügbarer Platz im Kapitel vorhanden ist.
Dadurch wird vermieden, dass, jedesmal wenn eine Funktion erscheint, diese Subroutine während des Berechnungsverfahren vom großen in den aktuellen Speicher überführt werden muss. Die Quickies müssen geordnet in die Liste eingetragen werden, indem die am häufigsten verwendeten an den Anfang gesetzt werden. Wenn während der Übersetzung ein Quickie nicht in das Kapitel überführt werden kann, weil er zu lang ist, prüft der Rechner, ob der folgende hineinpasst. Die Anzahl der in jedem Quickie verwendeten Anweisungen ist folgende:
ψ sq rt – 48; ψ cos – 36; ψ log – 42; ψ tan – 42; ψ radius – 48; ψ sin – 36; ψ exp – 50; ψ arctan – 58.
Die Quickies ψ sin und ψ cos ziehen sich auf die gleiche Subroutine auf, deswegen sollte nur einer von beiden in der Liste eingetragen werden. Das gleiches gilt für die Quickies ψ sqrt und ψ radius.

Beispiel: Im folgenden allgemeinen Schema eines Kapitels sieht man die Form und die Position, die die Quickies ausfüllen.

chapter 2
variables 1
v → 20
x → 5
Befehle
ψ sqrt
ψ arctan
close

IV. SPEICHER UND PROGRAMMIERUNG

Bisher lag die Aufmerksamkeit darauf, die verschiedenen Teile des aktuellen Zahlenspeichers (aktuelle Variablen, Indizes) sowie die Bestandteile eines Statements zu erklären. Von nun an beleuchten wir die allgemeine Organisation der Programme und die Erweiterungen, mit denen die Autocode-Sprache ausgestattet ist. Letztere fördern die Ähnlichkeit der mathematischen Sprache mit der im besagten System Verwendeten. Zu diesem Zweck beschäftigen wir uns zunächst mit der

funktionellen Anordnung des großen Speichers und der Organisation des aktuellen Befehlsspeichers und nehmen uns danach die oben erwähnten Erweiterungen und Programme vor.

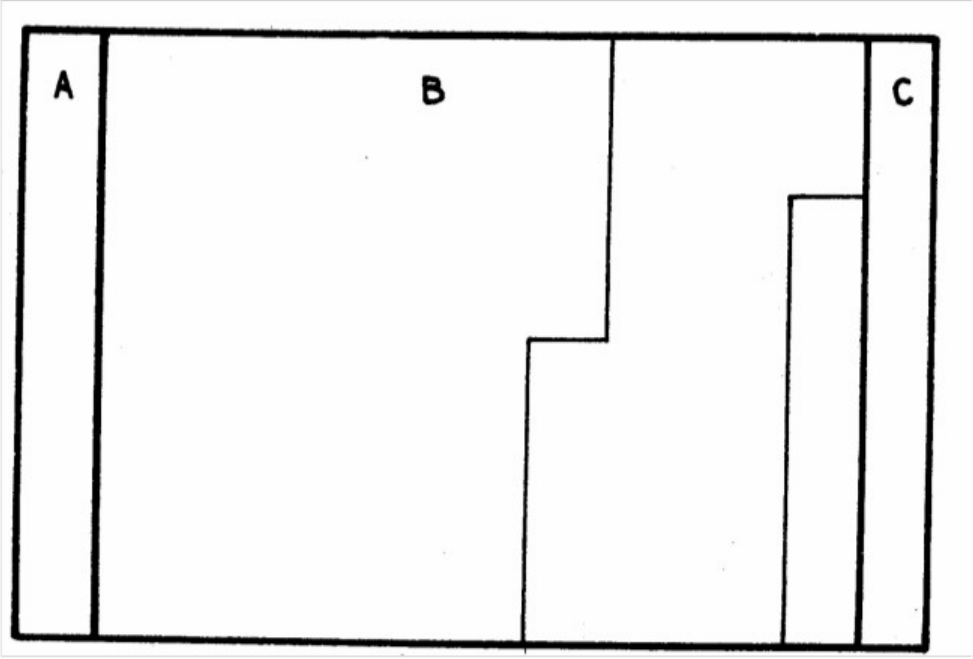


Abb. 3. – (Vgl. Anhang 1)

1. Aktueller Befehlsspeicher.

Der aktuelle Befehlsspeicher besteht aus drei Teilen, in denen sich die unterschiedlichen Teile des Programms befinden. In dem ersten sind einige Statements wie u. a. die Division und der Kapitelwechsel, einige Konstanten und verfügbarer Platz für die Verarbeitung der Subroutine. Der zweite und größte Teil enthält die Anweisungen und Konstanten, die das Programm bilden. Ist genug Platz, beinhaltet dieser Bereich auch die Subroutine, die die Quickies abruft. Im dritten Teil wird eine der von Autocode gebrauchten Funktionen nach Bedarf gespeichert (ψ sq rt, ψ sin, ψ exp, …,). (s. Abb. 3).

2. Großer Speicher.

Der große Speicher besteht aus einer Magnettrommel. Ihr Speichervermögen entspricht 13.824 Feldern, denen jeweils eine Zahl als Adresse zugeteilt wird. Diese Felder sind zwei großen Gruppen zugeteilt, je nachdem ob die entsprechende Zahl, die ihnen eine Adresse zuweist, positiv oder negativ ist. Jede Gruppe ist wiederum in verschiedene Bereiche mit unterschiedlichen Aufgaben unterteilt. In die erste Gruppe gehören Felder mit positiven Adressen von 0 bis +10751. Sie enthalten Hilfsvariablen und das völlig autokodierte Programm, welches sich in den letzten Feldern (10751, 10750, …) befindet. Weil jedes Kapitel 512 von diesen Feldern ausfüllt, wird der verfügbare Teil für die Variablen 10752-512n sein, wobei n die Anzahl der Kapitel ist. Zu den Feldern der zweiten Gruppen gehören die negativen Adressen, die Zahlen zwischen -1 und -3072 umfassend. Diese Felder bestehen aus folgenden Bereichen: ein Bereich für den Autocode-Übersetzer (-1 bis -1536); drei Bereiche für die zeitweilige Speicherung, den sogenannten Subkapitelspeicher (-1537 bis -2048), den Speicher des Hauptkapitels (-2049 bis -2560) und den Sonderspeicher (-2561 bis -3072). Die Verwendung dieser Bereiche wird

näher bei der Behandlung der Programme erklärt. Falls es notwendig ist, kann auch die Gruppe der Felder mit negativen Adressen verwendet werden, um die Hilfsvariablen zu speichern. (s. Anhang 1).

3. Hilfsvariablen.

Wenn die 509 aktuellen Variablen nicht ausreichen, kann man als Hilfsvariablen den Bereich der positiven Adressen im großen Speicher verwenden, der von den Anweisungen des Programms nicht benutzt wurde. Diese Variablen werden durch die Zahl ihrer Adressen gekennzeichnet. Da alle Zahlen durch den aktuellen Speicher hindurch müssen, um verarbeitet werden zu können, muss man über einige Funktionen verfügen, die die Übertragungen zwischen dem aktuellen und dem großen Speicher ermöglichen. Diese Funktionen sind folgende: ψ 6 (α) v, n; ψ 7 (α) v, n; preserve und restore.

Mit α verweist man auf eine Variable, einen Index, eine Konstante oder einen der algebraischen Ausdrücke. A nimmt immer einen ganzen Wert und rundet daher den Dezimalteil, wenn es einen gibt. Statt α werden normalerweise die Sondervariablen mit Akzent verwendet. Mit v wird eine allgemeine oder Sondervariable angezeigt. Schließlich zeigt n einen Index oder eine Ganzzahl an.

ψ 6(α) v, n

Die Funktion ψ 6 (α) v, n, überträgt den Inhalt der n Felder, die mit der Adresse α anfangen auf die aktuellen Variablen mit v als Anfangsspeicherplatz, vom großen in den aktuellen Speicher.

Beispiel: Der Befehl

ψ 6(10230)V_0, 5

überträgt den Inhalt der Positionen 10230, 10231,10232, 10233, 10234, auf die Variablen v_0, v_1, v_2, v_3, v_4.

Ausdrücke wie folgende sind möglich

6 (v_1 + 2v_j)w_i, h
6 (n) a, 1
6 (a') a_0, 16

ψ 7(α) v, n

Die Funktion 7 (α) v, n überträgt vom aktuellen in den großen Speicher den Inhalt der n aktuellen Variablen mit v als Anfangsspeicherplatz auf die vorzufindenden Hilfsvariablen mit den Adressen von α bis α + n – 1.

Beispiel: Der Befehl

ψ 7(10230)V_0, 5

überträgt den Inhalt von v_0, v_1, v_2, v_3, v_4 auf die Positionen des großen Speichers 10230, 10231,10232, 10233, 10234.

preserve
restore

Die Anweisung preserve speichert den Inhalt des ganzen aktuellen Zahlenspeichers in einem bestimmten Bereich des großen Speichers. Ihr untrennbare Befehl restore führt dem aktuellen Speicher das, was zeitweilig in einem bestimmten Bereich des großen Speichers ausgelagert war, wieder zu. Die Hauptanwendung dieses Befehlspaars ist die Verwendung von einem down Befehl zwischen beiden. Das Unterkapitel, das durch down aufgerufen wird, kann seinerseits durch einen weiteren down-Befehl ein Unter-Unterkapitel aufrufen. Daher kann das Paar preserve-restore verwendet werden, aber nicht in dem Unter-Unterkapitel. Der Bereich des großen Speichers, der für die zeitweilige Speicherung des preserve des Hauptkapitels verwendet wird, ist der Unterkapitelspeicher. Es ist zu anzumerken, dass die Sondervariable π nach jeder der Anweisungen wie preserve, restore, down, up und across die Konstante 3.141592... enthält.

V. ERWEITERUNGEN

Um der mathematischen Sprache ähnlicher zu sein, wird das Autocode-System mit einigen Schreibweisen und Funktionen, die Erweiterungen genannt werden, ausgestattet. Sie sind um einiges leistungsfähiger als die bisher vorgestellten Sonderanweisungen. An dieser Stelle sehen wir uns vier Erweiterungstypen genauer an:

1. Doppel-Wort-Arithmetik
2. Komplexe Algebra
3. Matrix-Algebra
4. Integration der Systeme von Differenzialgleichungen

1. Doppel-Wort-Arithmetik.

In den gewöhnlichen arithmetischen Statements operiert man mit Zahlen, die maximal neun signifikante Ziffern (als Gleitkommazahl oder Dezimalkommazahl betrachtet) enthalten. Manchmal erfordern die Berechnungen, dass man Ergebnisse mit mehr als neun Ziffern erhält. Dafür muss man die so genannte Double-Precision-Technik verwenden, die grundsätzlich die Ziffern von zwei verschiedenen Feldern als eine einzige Zahl betrachtet. Zum Beispiel, wenn man mit Double-Precision die Zahl 0.1237659380857364 ausdrücken möchte, muss man den signifikantesten Teil d.h. 0.12376593 in einem Feld und den weniger signifikanten Teil 80857364 in einem anderen Feld speichern. Bei einer beliebigen Zahl M nennen wir m1 den signifikanteren und m2 den weniger signifikanten Teil, deswegen könnte M durch das Paar (m_1 m_2) dargestellt werden. Die arithmetischen Operationen zwischen Double-Precision-Zahlen wären folgende:

Addition $M\,(m_1,m_2) + N\,(n_1,n_2) = S\,(s_1,s_2)$

in dem:

$s_1 = (m_1 + n_1)_1$

$s_1 = (m_2 + n_2)_1 + (m_1 + n_1)_2$

Produkt $M (m_1,m_2) N (n_1,n_2) = P (p_1,p_2)$
in dem:
$P_1 = (m_1 n_1)_1$
$P_1 = (m_1 n_2)_1 (m_2 n_1)_1 (m_1 n_1)_2$

Quotient $M (m_1,m_2)/N (n_1,n_2) = Q (q_1,q_2)$
in dem:
$q_1 = (m_1/n_1)_1$
$q_1 = ((m_2 n_1 - m_1 n_2)/n_1^2)_1 + (m_1/n_1)_2$

In den vorherigen Beispielen drückt der sich außerhalb der Klammer befindende Subindex (1 oder 2) aus, ob es sich um den signifikanteren oder weniger signifikanten Teil der sich in der Klammer angegebenen exakten Operation handelt. Man muss berücksichtigen, dass es manchmal zu einer Übertragung des weniger signifikanten Teil in den signifikantesten Teil der oben angeführten Operationen kommen kann. Es ist darüber hinaus offensichtlich, dass es sinnlos ist, die Double-Precision-Technik anzuwenden, wenn Rundungsfehler in dem signifikantesten Teil gemacht werden. Im Autocode-System wird durch eine Erweiterungsanweisung jede einzelne Grundrechenart mit der Anwendung der Double-Precision-Technik ausgedrückt. Dafür werden folgende Schreibweisen verwendet:

```
((A,B))=((X,Y))
((A,B))=((X,Y))+((U,V))
((A,B))=((X,Y))-((U,V))
((A,B))=1/((X,Y))
((A,B))=((X,Y))*((U,V))
((A,B))=((X,Y))/((U,V))
```

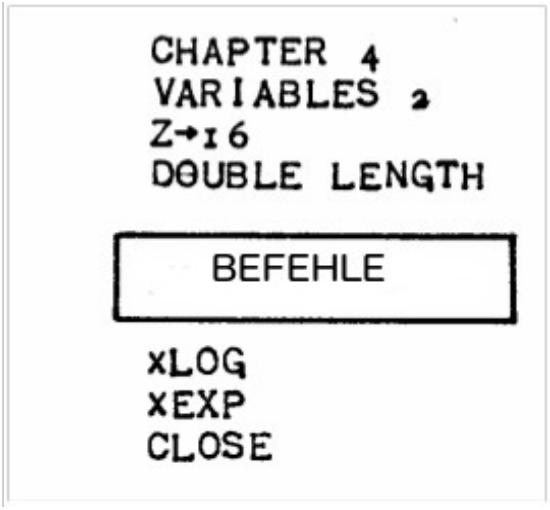
Man drückt symbolisch durch ein Aktuellenvariablen-Paar in doppelten Klammern und durch ein Komma getrennt ((a, b)) eine Double-Precision-Zahl aus, in der der signifikanteste Teil in der aktuellen Variable a und der weniger signifikante Teil in der Variable b gespeichert sind. Das erste Zeichen zeigt an, wo die Ergebnisse des zweiten Zeichens gespeichert werden müssen. Aus diesem Grund können a und b nur aktuelle Variablen sein, während x, u, y, v k alle Variablen sein können bzw. x, u können Ganzzahlen sein, wenn die entsprechenden Variablen y, v 0 sind.
Der weniger signifikante Teil ist immer positiv.
Die Variablen, die die beiden Teile ausdrücken, müssen nicht unbedingt konsekutiv sein.
In den Double-Precision-Operationen hat das Zeichen ? keine Wirkung.

double length

Wenn man eine der Double-Precision-Erweiterungen in einem Kapitel verwendet, muss man die double-length-Direktive zwischen den Anfangsdirektiven des Kapitels einschließen. Die genannte

Direktive beinhaltet die Double-Precision-Subroutinen am Anfang des Kapitels.

Beispiel: Im folgenden allgemeinen Schema eines Kapitels wird gezeigt, wo die double-length-Direktive positioniert ist:



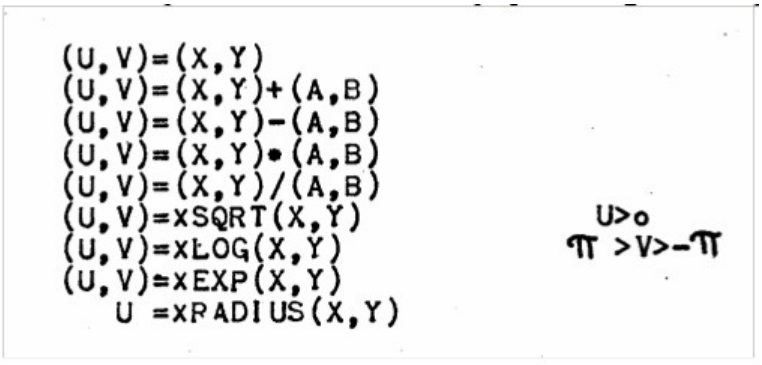
2. Komplexe Algebra. Andere im Autocode-System enthaltene Erweiterungen sind die Grundoperationen und einige Funktionen mit komplexen Zahlen durch einen einfachen Befehl, wie im Folgenden gezeigt wird:

(α,ß)

Eine komplexe Zahl wird im Autocode-System als ein Variablen- oder Konstantenpaar, die in Klammern durch ein Komma getrennt sind, definiert. Der erste Teil α (Variable oder Konstante) stellt den Realteil dar und der zweite Teil β (Variable oder Konstante) stellt den Imaginärteil dar.

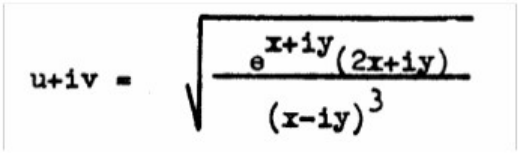
Beispiel: (A, B) (3.241, 6.742) (X (I + 1), X(I + 2))
(AI, BI) (AI, -3.94) (AI, A(I + 1))

Die Autocode-Sprache ermöglicht folgende Operationen mit komplexen Zahlen:



Es muss wieder die Wichtigkeit der korrekten Schreibweise festgestellt werden. Es ist auch wichtig anzumerken, dass bei der Multiplikation das Sternchen verwendet werden muss. Bei komplexen Operationen hat das Zeichen ? keine Wirkung. Die komplexen Funktionen sqrt, log, exp verwenden die reellen Funktionen für ihre Berechnung: sqrt; sin und cos für exp; arctan für log. Aus diesem Grund müssen bei der Verwendung einer komplexen Funktion die entsprechenden reellen Funktionen als Quickies hinzugefügt werden.

Beispiel: Ausdrucksberechnung



Dafür wird folgendermaßen vorgegangen:

```
£1=2X
£2=-Y
(£3,£4)=XEXP (X,Y)
(£5,£6)=(£3,£4)*(£1,Y)
(£7,£8)=(£5,£6)/(X,£2)
(£9,£10)=XSQRT (£7,£8)
(U,V)=(£9,£10)/(X,£2)
```

Am Ende des Kapitels, das dieses Statement enthalten würde, sollten folgende Quickies eingefügt werden:

ψ exp
ψ sin
ψ sqrt

3. Matrizenrechnung. Die Matrizen werden als eine einzige Spalte gespeichert, indem eine Gruppe Hilfsvariablen benutzt wird. Die erste Stelle einer Matrix A wird durch den Inhalt einer Sondervariable, normalerweise a', angezeigt. Um das Element a_{ij} in einer Matrix der Ordnung m x n zu finden, wird folgender Ausdruck verwendet:

$$a_{ij} = a' (n (i - 1) + (j - 1))$$

Das zweite Element drückt aus, dass ab der Anfangsstelle a' (i-1) Blöcke, mit (i-1) Zeilen und n-Elementen pro Zeile durchlaufen wurden. Im Block i (also Zeile i) wurden j-1 Elemente durchlaufen. Daraus folgt, dass das aktuelle Element a_{ij} ist.

Im Autocode-System erfolgt diese Suche sowie die Hauptoperationen zwischen Matrizen wie z. B. das Lesen und Schreiben, automatisch. Hierfür ist es mit einigen Funktionen, die von mehreren Parameter abhängig sind, ausgestattet. Ein Merkmal der Parameter ist, dass es Ganzzahlen sind, deren Wert zwischen 8 und 28 liegen kann, je nach Aufgabe, die sie durchführen sollen.
Die Parameter sind:

- eine Sondervariable oder eine Ganzzahl, um die Position des ersten Elements der Matrizen, die von der Funktion bearbeitet werden, im großen Speicher anzuzeigen
- eine Sondervariable oder eine Ganzzahl, um die Gesamtanzahl der Elemente oder die Ordnung anzuzeigen
- ein Index oder eine Ganzzahl, um die Anzahl der Ganzen- oder Dezimalziffern, die wir drucken wollen, anzuzeigen. (Ein Index oder eine Ganzzahl wird auch verwendet, um die Ordnung einer Matrix bei der Division anzuzeigen).
- eine Sondervariable, um einen Skalar anzuzeigen

  Normalerweise wird eine Variable ohne Akzent benutzt.

Hiernach werden die Funktionsgruppen, die folgende Aufgabe ausführen, behandelt:

- eine Matrix als Gleit- oder Festkommazahl drucken
- eine Matrix oder einen Vektor lesen
- Linearkombination zweier Matrizen, auch bei den Einzelfällen der Diagonal- und Einheitsmatrix
- Transposition einer Matrix
- die Determinanten einer Matrix finden
- Multiplikation und Division von Matrizen

Drucken

ψ 8 (a', u, v, m, n)
ψ 9 (a', u, v, n)

Die Funktion ψ 8 (a', u, v, m, n) druckt die Matrix A, die im großen Speicher ab der Speicherstelle a' gespeichert ist, folgendermaßen:

- jede Zeile der Matrix füllt auf dem Papier eine Spalte v Elementen aus
- nach den v Elementen einer Zeile folgt eine Leerzeile
- nach der Leerzeile folgt die nächste Zeile der Matrix, getrennt von der vorigen durch einen Tab
- jedes Element wird als Festkommazahl mit Vorzeichen (falls negativ), m Ganzziffern und n Dezimalziffern gedruckt. Sie werden als Gleitkommazahl geschrieben, wenn ihr Wert größer als 10^{16} oder wenn m = 0 ist.

Beispiele: Angenommen, dass die Matrix

```
        |2,512  2,753   -3,104|
A=      |                     |
        |5,215  -7,126  9,627|
```

in den Feldern des großen Speichers mit den Adressen 100, 101, 102, 103, 104, 105 und a' in Nummer 100 gespeichert ist; die Funktion ψ 8 (a', 2, 3, 1, 2) druckt die vorige Matrix folgendermaßen aus:

```
2, 51
2,75
-3,10

5, 21
-7,13
9,63
```

Die gleiche Wirkung hätte die Funktion ψ 8 (100, 3, 2, 1, 2).

Die Funktion ψ 9 (a', u, v, n) hat die gleiche Wirkung, aber die Zahlen werden als Gleitkommazahl mit n = 10 Ziffern in der Mantisse und mit einem Exponent kleiner als 70 gedruckt. Sie hat genau die gleiche Wirkung wie die Funktion ψ 8 mit m = 0.

ψ 10(a', w)

Lesen

Diese Funktion liest die w gelochten Zahlen auf dem Lochstreifen als Fest- oder Gleitkommazahl und speichert sie in den Feldern des großen Speichers, dessen erste Stelle mit der Sondervariablen a' registriert ist. Wenn w = u.v kann diese Menge an Zahlen als eine Matrix der Ordnung (u x v) betrachtet werden. In allen Fälle kann es als ein w-dimensionaler Vektor betrachtet werden.

Linearkombination

Im Folgenden werden die Funktionen, die die Berechnung der Linearkombination zweier Matrizen ermöglichen, angegeben. Wegen ihrer verschiedenen Besonderheiten werden sie in drei Gruppe geteilt, je nachdem, ob es sich um die Linearkombination zweier beliebiger Matrizen, einer beliebigen Matrix und der Einheitsmatrix, oder einer beliebigen Matrix und der Diagonalmatrix handelt. Natürlich müssen die Matrizen in den zwei letzen Beispielen quadratisch sein.

- zwei beliebige Matrizen

a' = ψ 11(b', c', w)	A = B + C	Ordnung(u,v)
a' = ψ 12(b', c', w)	A = B - C	Ordnung(u,v)
a' = ψ 13(b', x, c', w)	A = B + xC	Ordnung(u,v)
a' = ψ 14(b', x, c',w)	A = B - xC	Ordnung(u,v)
a' = ψ 15(b',w)	A = B	Ordnung(u,v)

Die Sondervariablen a', b', c' enthalten die Adressen des ersten Feldes der Bereiche, die die jeweiligen Matrizen A, B, C ausfüllen. Die Sondervariable x stellt ein Skalar dar. Die drei Matrizen müssen der

gleichen Ordnung sein und w enthält der Anzahl ihrer Elemente, d. h. w = u.v.

Beispiele: Seien die Matrizen

B= |1 2 2| C= |2 1 1|
|2 1 1| |1 2 2|

in den Speicherstellen des großen Speichers mit den Nummern 20, 21, 22, 23, 24, 25 und 26, 27, 28, 29, 30, 31 gespeichert. Wenn die Sondervariablen b', c' und a' die jeweiligen Zahlen 20, 26, 32 enthalten, wird die Funktion

a' = ψ 12 (b', c', 6)

in den Speicherstellen 32, 33, 34, 35, 36 folgende Matrix speichern:

A= |-1 1 1|
|1 -1 -1|

Angenommen, seien die vorigen Bedingungen und sei 0,5 in a gespeichert, dann speichert die Funktion

a' = ψ 13 (b', a, c', u)

die Matrix

A= |2 2,5 2,5|
|2,5 2 2|

in den Speicherstellen 32, 33, 34, 35, 36, 37.

- eine beliebige Matrix und die Einheitsmatrix

a' = ψ 17(b', u)	A = B + I	Ordnung(u,u)
a' = ψ 18(b', u)	A = B - I	Ordnung(u,u)
a' = ψ 19(b', x, u)	A = B + xI	Ordnung(u,u)
a' = ψ 20(b', x, u)	A = B - xI	Ordnung(u,u)

In diesem Fall sind die Matrizen quadratisch und u stellt die Ordnung anstatt der Anzahl der Elemente dar.

- eine beliebige Matrix und eine Diagonalmatrix

a' = ψ 21(b', d', u)	A = B + D
a' = ψ 22(b', d', u)	A = B - D
a' = ψ 23(b', x, d', u)	A = B + xD

a' = ψ 24(b', x, d', u) A = B - xD

Auch in diesem Fall sind die Matrizen quadratisch mit der Ordnung u. Die Diagonalmatrix wird als ein u-elementiger Vektor in einem Bereich des großen Speichers, dessen erste Adresse sich in d' befindet, gespeichert.

Transponierte Matrix

a' = ψ 16(b', u, v)

Diese Funktion speichert die transponierte Matrix in b' auf die durch a' gezeigten Speicherstellen. Daraus folgt, dass die Ordnung der zweiten v x u ist, wenn die erste u x v ist.

Determinante

x = ψ 25(a', u)

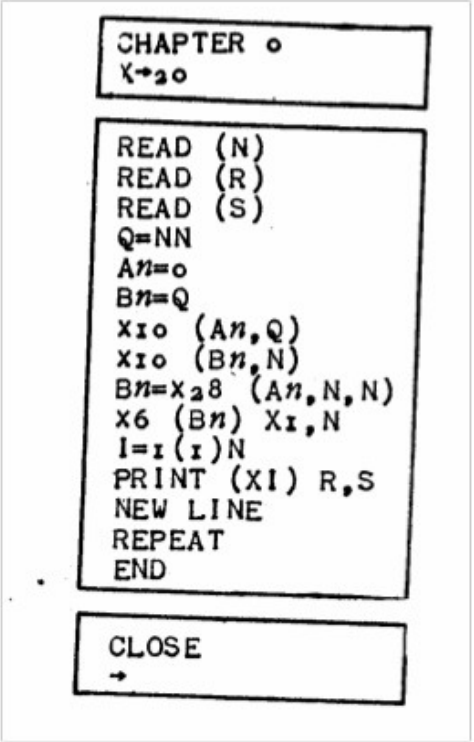
Durch diese Funktion wird in x der Wert der Determinanten der Matrix mit der Ordnung u, die sich im Bereich a' befindet, gespeichert.

Multiplikation und Division

a' = ψ 26(b', c', u, v, w)	A(u,v) = B(u,w)C(w,v)
a' = ψ 27(b', c', u, v, w)	A(u,v) = B(u,w)C'(w,v)
a' = ψ 28(b', m, n)	$A(m,n) = B^{-1}(m,m)A(m,n)$

C' stellt das transponierte C dar. Es ist zu bemerken, dass hier die Ordnung der Matrizen mit m, n statt u, v (wie bisher) gezeigt wird, denn dafür werden Indizes oder Ganzzahlen kleiner als 511 verwendet. Wichtig zu wissen ist auch, dass die Division die Matrix B zerstört.

Beispiel: Wenn man ein Programm entwickeln möchte, um ein lineares Gleichungssystem $a_i x_i = b_i$ mit Ordnung $n \leq 20$ zu lösen und die Lösungen mit r Ganzziffern und s Dezimalziffern spaltenweise zu drucken, würde man folgendermaßen vorgehen:



4. Integration der Systeme von Differenzialgleichungen. Das Autocode-System ist mit einer Erweiterung ausgestattet, die durch die Runge-Kutta-Methode die Integration eines Systems von Differenzialgleichungen ermöglicht, indem die Anweisung (int step (m)) benutzt wird. Dafür muss man zuerst die Variablen reservieren, die Parameter (Anzahl von Gleichungen, Länge des Schrittes, Anfangsbedingungen) festlegen und die Gleichungen definieren, wie später genauer erklärt wird.

Es integriert das System von Differenzialgleichungen

int step (m)

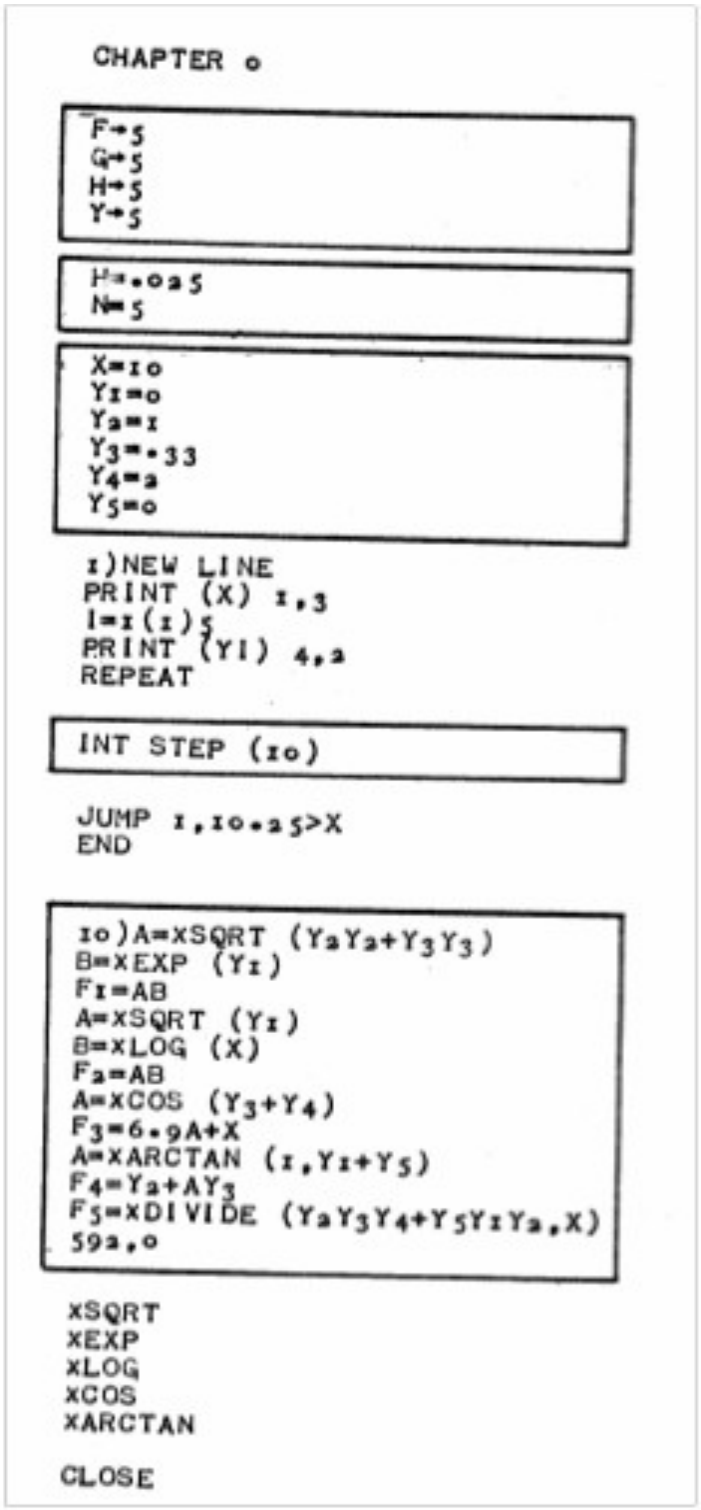
$$f_i = dy_i / dx = f_i (x, y_1, y_2, \ldots y_n)$$

in dem i = (1, 2, 3, ...n). Vorher muss man die Hilfsvariablen f_i, y_i, g_i, h_i (die zwei letzte als Arbeitsspeicher) reservieren, die Werte der Indizes n (Anzahl von Gleichungen) und h (Länge des Schrittes) definieren, die Anfangswerte von x, y_1, y_2, … y_n festlegen und die Gleichungen zwischen der Marke m und der Anweisung 592,0 bestimmen.

Beispiel: Integration des Systems

$$f_1 = dy_1 / dx = (y_2 + y_3)^{1/2}\ e^{y1}$$
$$f_2 = dy_2 / dx = y_1^{1/2} \log_e x$$
$$f_3 = dy_3 / dx = 6.9 \cos (y_3 + y_4) + x$$
$$f_4 = dy_4 / dx = y_2 + y_3 \arctan (y_1 + y_5)$$
$$f_5 = dy_5 / dx = (y_2\, y_3\, y_4 + y_5\, y_1\, y_2) / x$$

im Intervall (10, 10.25) mit einem 0.025 Schritt und mit Anfangswerte $y_1(10) = 0$; $y_2(10) = 0.33$; $y_4(10) = 2$; $y_5(10) = 0$. Das Programm zur Integration dieses System wäre folgendes:



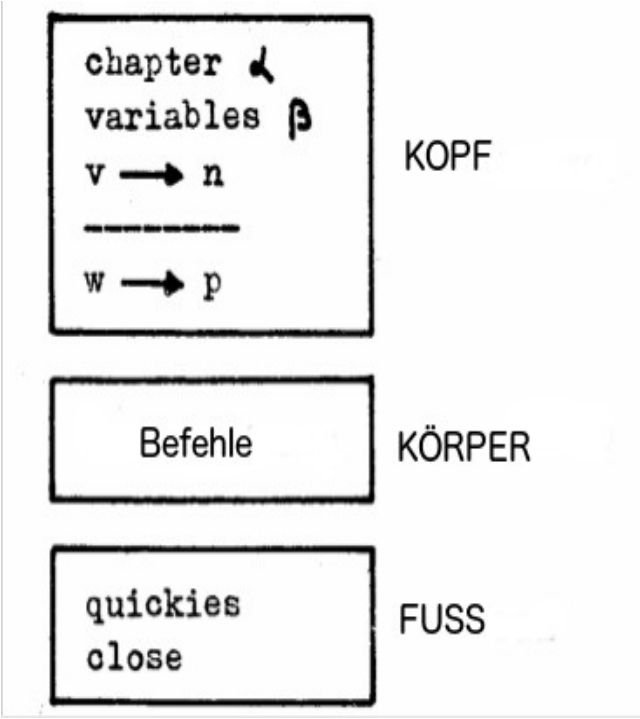
```
CHAPTER o

F→5
G→5
H→5
Y→5

H=.025
N=5

X=10
Y1=0
Y2=1
Y3=.33
Y4=2
Y5=0

1)NEW LINE
PRINT (X) 1,3
I=1(1)5
PRINT (YI) 4,2
REPEAT

INT STEP (10)

JUMP 1,10.25>X
END

10)A=XSQRT (Y2Y2+Y3Y3)
B=XEXP (Y1)
F1=AB
A=XSQRT (Y1)
B=XLOG (X)
F2=AB
A=XCOS (Y3+Y4)
F3=6.9A+X
A=XARCTAN (1,Y1+Y5)
F4=Y2+AY3
F5=XDIVIDE (Y2Y3Y4+Y5Y1Y2,X)
592,0

xSQRT
xEXP
xLOG
xCOS
xARCTAN

CLOSE
```

VI. DIE PROGRAMME

Ein ganzes Programm enthält verschiedene Teile, die ihrerseits in kleinere Einheiten unterteilt sind. Das Gesamtprogramm, das aus Unterprogrammen besteht, wird Makroprogramm genannt. Jedes Unterprogramm setzt sich aus einem oder mehreren Kapiteln zusammen. Die einzelnen Unterprogramme eines Makroprogramms sind durch ein Unterprogramm - das Hauptprogramm - miteinander verknüpft. Im Folgenden werden die formalen Konventionen dargestellt, die die Kapitel, Unterprogramme (oder Programme, falls sie sich nicht auf ein Makroprogramm beziehen) oder die allgemeine Organisation eines Makroprogramms bei der Programmierung in Autocode ausführen.

1. Die Kapitel.

Die Kapitel sind mit drei wesentlichen Teilen ausgestattet: Kopf, Körper und Fuß. Der Kopf und der Fuß bestehen aus Direktiven und werden benutzt, um die Übersetzung unserer Befehle zu organisieren. Der Körper ist die Menge der Befehle, die für die Programmausführung benutzt werden. Folgendes Schema zeigt die allgemeine Struktur eines Kapitels:



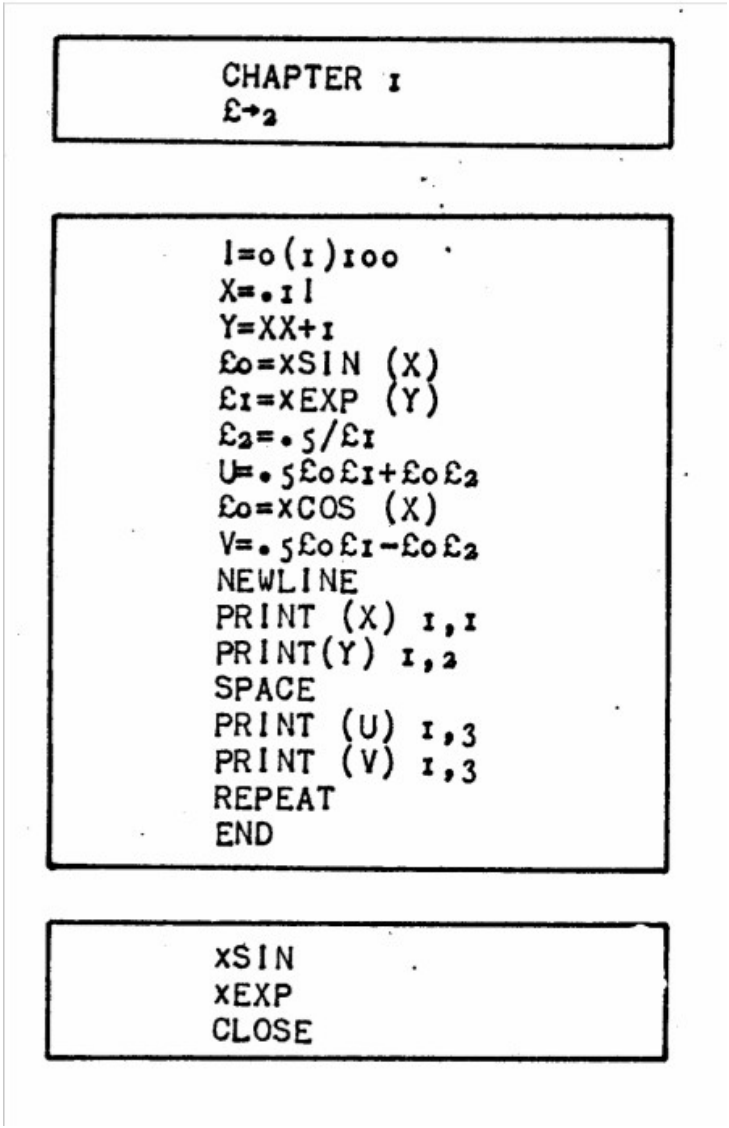
In manchen Fällen können die Direktiven variables, → und Quickies ausgelassen werden. Wenn sie jedoch verwendet werden, ist die im oben genannten Schema angezeigte Reihenfolge zu beachten. In allen Fällen benötigt man die Direktiven chapter α und close.

Beispiele:

a) Tabellieren der Funktion mit komplexer Variable

$$w = \sin z$$

(in der $w = u + iv$ und $z = x + iy$) in den Punkten der Fläche x,y, die auf der Kurve $y = x^2 + 1$ mit einer Abszisse im Intervall (0,1) und Schritt 0,01 liegen.



```
CHAPTER 1
£→2
```

```
I=0(1)100
X=.11
Y=XX+1
£0=xSIN (X)
£1=xEXP (Y)
£2=.5/£1
U=.5£0£1+£0£2
£0=xCOS (X)
V=.5£0£1-£0£2
NEWLINE
PRINT (X) 1,1
PRINT(Y) 1,2
SPACE
PRINT (U) 1,3
PRINT (V) 1,3
REPEAT
END
```

```
xSIN
xEXP
CLOSE
```

b) Lösung des Integrals $\int_{-1,1} e^x dx$ mit Hilfe der Simpsonregel und mit einem Fehler kleiner als D.

```
READ (D)
H=2
F=1
G=0
N=1
A=XEXP (1)
E=A+1/A
B=E+4
B=B/3
1)N=2N
H=.5H
A=.5H-1
G=G+F
F=0
I=1(1)N
X=XEXP (A)
F=F+X
A=A+H
REPEAT
C=E+4F+2G
C=HC/6
Y=XMOD (B-C)
B=C
JUMP 1,Y≥D
PRINT (C)1,9
END
```

BEMERKUNG. Es ist zu beachten, dass die geraden Punkte einer Unterteilung mit der geraden und ungeraden der vorigen Unterteilung zusammenfallen. Deswegen braucht man nur die ungeraden Punkte jeder neuen Unterteilung.

Wir möchten wieder daran erinnern, dass die maximale Anzahl der Maschinenbefehle, die ein Kapitel ausfüllen kann, 832 ist. Um die Zahlen der benutzten Befehle zu kontrollieren, stehen die Direktive psa (print space available) zur Verfügung. Falls es mehr als 832 Befehle gibt, zeigt psa den Fehler an, in dem die Kapitelnummer sowie die wegen Platzmangel nicht enthaltenden Befehle gedruckt werden. Jedes Kapitel füllt, je nach seiner Nummer, einen festen Bereich im großen Speicher aus. Die gleichen Marken können in verschiedenen Kapiteln verwendet werden. Der erste Befehl eines Kapitels muss immer markiert werden. Jedes Mal wenn man von einem Kapitel zum nächsten wechselt, wird in der Sondervariablen die Zahl 3,1415... wieder eingesetzt. In ein und demselben Kapitel müssen sich immer die Schleifengrenze i = p(q)r, repeat und die n) = 4) oder n) = m), jump(n) befinden.

2. Die Programme.

Ein Programm kann ein oder mehrere Kapitel beinhalten. Diese Kapitel dienen dazu, bestimmte Phasen der Berechnung auszuführen. Es gibt zwei verschiedene Typen: die Unterprogramme und das Hauptprogramm. Zur Verfügung steht eine große Anzahl von allgemeingültigen Unterprogrammen, die von Nutzern des Autocode-Systems entwickelt wurden und die eine Bibliothek bilden.

2.1 Die Unterprogramme

Die Unterprogramme sind unabhängige Recheneinheiten und können im Allgemeinen in sehr unterschiedlichen Makroprogrammen benutzt werden. Ein Unterprogramm ist z. B. die Menge der Kapitel, die für die Berechnung der Eigenvektoren und Eigenwerte einer beliebigen Matrix notwendig sind. Um ein Programm benutzen zu können, braucht es einen Namen, der ermöglicht es aufzurufen. Darüber hinaus muss es möglich sein darauf hinzuweisen, wo sich die Daten befinden und wohin die Ergebnisse zu speichern sind. Die erste Voraussetzung wird durch den Titel, die anderen beiden durch die Parameter erfüllt.

Titel programme - α

Um ein Programm von einem anderen zu unterscheiden und sie in einem Makroprogramm zusammenfügen zu können, ist für jedes Programm die Direktive programme-α vorgesehen, in der α eine zwischen 1-1023 enthaltende Ganzzahl sein muss. Die Zahlen größer als 500 werden konventionell reserviert, um die Programme der Bibliothek zu identifizieren. Die Benutzung des Bindestriches zwischen programme und α ist unerlässlich.

Die Parameter

Die Parameter sind Variablen oder Indizes, deren Werte von dem Fall abhängen, auf den das Programm angewendet wird. Der “Name” der Variable oder des Index wird vom Programm festgelegt und der in der besagten Variablen bzw. im Index gespeicherte Zahlenwert ist durch die konkrete Anwendung des Programms bedingt. Dafür werden laut folgender Konvention die Sondervariablen benutzt:

1. - Die individuellen Parameter, die Bereichsdimensionen, usw. werden durch Sondervariablen ohne Akzent oder Indizes ausgedrückt.

2. - Die Zahlenbereiche (Vektoren, Matrizen, …) werden durch eine Sondervariable mit Akzent, die die Adresse des ersten Elementes des Bereiches enthält, gekennzeichnet.

3. - Die Ergebnisse werden auf analoge Weise ausgedrückt, je nachdem, ob sie skalar oder vektoriell sind.
Es ist zu berücksichtigen, dass man die Direktive preserve benutzen soll, bevor die Parameter die Werte annehmen. Ebenso ist es angemessen, restore sofort nach dem Befehl, der das Unterprogramm ausführt, zu verwenden. Auf diese Weise bleibt der Inhalt des aktuellen Speichers erhalten und die Zerstörung seines Inhalts während der Ausführung des Unterprogramms wird verhindert.
Beispiel: Angenommen, dass das programme-506 die reellen und komplexen Wurzeln eines Polynoms mit reellem oder komplexem Koeffizient berechnet

$$\alpha_0 z^n + \alpha_1 z^{n+1} + \ldots + \alpha_{n-1} z + \alpha_n$$

Es ist offensichtlich, dass man Folgendes kennen muss, bevor das Unterprogramm anfängt

- den Grad der Gleichung
- den reellen Teil der Koeffizienten
- den Imaginärteil der Koeffizienten

Um das Programm zu beenden, muss man wissen, wo sich Folgendes befindet:

- der reelle Teil der Wurzeln
- der Imaginärteil der Wurzeln
- die Präzision der Wurzeln

Wenn man annimmt, dass in n der Grad der Gleichung, in a' die Adresse des reellen Teils erstes Element, in b' die Adresse des Imaginärteils des ersten Koeffizientes, in e die Präzision und in c' und d' die Adresse des reellen und imaginären Teils der Wurzeln gespeichert sind, sind dann die Variablen und Indizes n, e a', b', c', d' die Parameter des programme-506. Beim konkreten Fall einer Gleichung sechsten Grades muss man Folgendes schreiben, bevor das Unterprogramm aufgerufen wird

```
----------
preserve
n = 6
c = 0.001
a' = 100
b' = 106
c' = 112
d' = 118
down 1/1-506
restore
----------
----------
```

in dem die Koeffizienten in den Zellen 100-101 des großen Speichers und die Wurzeln in den Zellen 112-113 des gleichen Speichers abgespeichert werden.

2.2 Das Hauptprogramm

Das Hauptprogramm verknüpft alle Unterprogramme, die an unserem Berechnungsverfahren beteiligt sind, d. h. es “bildet” die Parameter und befiehlt die Sprünge. Außer den Verknüpfungsoperationen führt es diejenigen aus, die wegen ihrer Kürze und Einfachheit kein Sonderprogramm brauchen. Das Hauptprogramm braucht normalerweise keinen Titel, außer wenn es durch einen across-Befehl ein Unterprogramm aufruft.

2.3 Die Bibliothek.

Das Computing Machine Laboratory der Manchester Universität hat in Zusammenarbeit mit Ferranti und allen Benutzern des Mercury Rechners(*) eine Bibliothek entwickelt. Sie enthält allgemeine Programme, die in sehr verschiedenen Berechnungsverfahren nützlich sind und angewendet werden können. Irgendeines dieser Programme kann in unser Makroprogramm zu den gleichen Bedingungen wie alle anderen Unterprogramme eingefügt werden. Manche Programme der Bibliothek sind nicht vollständig, wie z. B. das Programm für die Berechnung des Integrals einer Funktion, dem eine Routine zum Generieren des Betreffenden hinzugefügt werden muss. Um die Programme der Bibliothek nutzen zu können, muss man folglich ihre Spezifikationen also die Parameter, Variablen oder Indizes sowie ihre Anwendung kennen.
Auf den folgenden Seiten fügen wir den Bogen mit den Spezifikationen, der dem Programm für die Berechnung der Wurzeln eines Polynoms beiliegt, hinzu.

(*) Computing Machine Laboratory—Manchester University
Computing Center. Oxford University
London University Computing Group
United Kingdom Atomic Energy Authority
United Kingdom Atomic Energy Risley (Industrial Group)
United Kingdom Atomic Energy Harwell
United Winfrith Heath
Royal Aircraft Establishment. Farmboroug
Meteorological Office
Imperial Chemical Industries Ltd.
Associated Electrical Industries Ltd. (AEI)
General Electrical Company Ltd. (GEC)
Shell Petroleum Ltd.
BP Petroleum Ltd.
Comisión Francesa de Energía Atómica. Saclay
Comisión Belga de Energía Atómica
Comisión Sueca de Energía Atómica
Instituto Noruego de Investigaciones de la Defensa
CERN. Ginebra
Instituto de Cálculo. Universidad de Buenos Aires.

Titel: Programme-506 Lösung algebraischer Gleichungen.
Ziel: Dieses Programm berechnet die Nullen des Polynoms

A -506

$$a_0 z^n + a_1 z^{n-1} + \ldots + a_n = 0$$

wobei a_i reell oder komplex sein kann.

Beschreibung: Die verwendete Methode ist von D. E. Muller in M.T.A.C., Oktober 1956 (S. 208) beschrieben.

Jede Wurzel wird durch eine iterative Methode, gefolgt durch die Entfernung des entsprechenden Faktors der Gleichung, bestimmt. Die erforderte Relativpräzision der Wurzeln wird durch einen Programmparameter e angegeben, der folgendermaßen als Konvergenzkriterium verwendet wird

$$\left| \frac{x^{(m+1)} - x^{(m)}}{\frac{1}{2}x^{(m+1)} + \frac{1}{2}x^{(m)}} \right| < e$$

wobei $x^{(m)}$, $x^{(m+1)}$ zwei sukzessive Iterationen einer Wurzel x sind.
Um vier signifikante Ziffern zu bekommen, muss e=0.001 sein.

Parameter:

n	Ordnung der Gleichung (≤ 116)
e	Relativpräzision der Wurzeln
a'	Die reellen Teile von a_i sind in a', a'+1, … a'+n abgespeichert.
b'	Die imaginären Teile von a_i sind in b', b'+1, … b'+n abgespeichert.
c'	Die reellen Teile der Wurzeln sind in c', c'+1, … c'+n-1 abgespeichert.
d'	Die imaginären Teile der Wurzeln sind in d', d'+1, … d'+n-1 abgespeichert.

Eingabe: Das Programm besteht aus einem einzigen Kapitel, das so aufgerufen wird:

1. Durch “down 1/1 – 506”, wenn a_i komplex sind
2. Durch “down 2/1 – 506”, wenn a_i reell sind (in diesem Fall ist es nicht notwendig, b' anzugeben)

Zeit: 1 Min. für n=25, 4 Min. für n=50 und 16 Min. für n=100.

Präzision: Siehe Beschreibung oben.

Autor: R. H. Kerr, Universität Manchester
Datum: 27. April 1959

Während den Programmen der Bibliothek eine Nummer zwischen 501 und 1023 zugewiesen wird, wird unseren Programmen eine Nummer kleiner als 501 zugewiesen. Damit ist es möglich, sie angesichts jedes konkreten Problems zu unterscheiden. Anschließend stellen wir eine Liste mit den aktuellen Programmen der Bibliothek vor. Allerdings kann sich ihr Inhalt ändern, indem sich die Anzahl der Programme erhöht bzw. die Vorhandenen verbessert werden.
Die Liste zeigt die Programmtitel sowie ihren Zweck.

Programme-501 Grenzwerte einer Folge bestimmen
Gegeben seien die (n+1) Elemente einer Folge

$a_0\ a_1\ a_2\ \ldots\ a_n$.

Dieses Programm berechnet seinen Grenzwert für $n \to \infty$.

Programme-502 Einfache Quadratur
Dieses Programm berechnet den Wert des Integrals

$\int_{a,b} f(x)\,dx$,

wobei f(x) keine Besonderheiten im reellen Intervall (a, b) aufweist.

Programme-503 Quadratur
Dieses Programm berechnet

$\int_{a,b} (b-x)^u\,(x-a)^v\ f(x)\,dx$,

wobei u>-1, v>-1 und f(x) keine Besonderheiten im reellen Intervall (a, b) aufweist.

Programme-504 Quadratur von unendlichen Integralen
Dieses Programm berechnet den Wert des Integrals

$\int_{c,\infty} f(x)\,dx$,

wobei f(x) keine Besonderheiten im reellen Intervall (c, ∞) aufweist.

Programme-505 Harmonische Analyse
Gegeben seien für die Periode L die Werte einer periodischen Funktion f(x) = f(x+L) in den gleich getrennten 2n + 1 Punkten

$x_r = x_0 + (r\ L/2n),\ r = 0\ (l)\ 2n.$

Das Programm berechnet den Koeffizient der harmonischen Näherung.

$$f(x) = a_0 + \sum_{1}^{n} \left\{ a_r \cos \frac{2\pi r}{L} (x-x_0) + b_r \operatorname{sen} \frac{2\pi r}{L} (x-x_0) \right\}$$

Programme-506 Lösung algebraischer Gleichungen

Dieses Programm berechnet die Nullen des Polynoms

$a_0 z^n + a_1 z^{n-1} + \ldots\ldots + a_n = 0$,

wobei a_i reell oder komplex sein kann.

Programme-507 Autokorrelation und Kreuzkorrelation

Gegeben seien zwei Folgen (die identisch sein können):

$x_0 \quad x_1 \ldots\ldots x_n \qquad\qquad y_0 \quad y_1 \ldots\ldots y_n$

Das Programm berechnet die Summen

$$Z_s = \frac{A_s - \frac{B_s - D_s}{n-s}}{\sqrt{\left(C_s - \frac{B_s^2}{n-s}\right)\left(E_s - \frac{D_s^2}{n-s}\right)}} \qquad \text{für } S = 0(1)\ p$$

$$A_s = \sum_{t=1}^{n-s} x_t y_{(t+s)} \; ; \; B_s = \sum_{t=1}^{n-s} x_t \; ; \; C_s = \sum_{t=1}^{n-s} x_t^2 \; ; \; D_s = \sum_{t=s+1}^{n} y_t \; ; \; E_s = \sum_{t=s+1}^{n} y_t^2$$

Programme-508 Lösung und Tabellierung eines Systems von Differentialgleichungen

Das Programm löst und tabelliert das System von Differentialgleichungen

$dy_i / dx \; f_i (y_1, y_2, y_3 \ldots. y_n ; x) \qquad i = 1(l)\ n$

mit den Anfangsbedingungen $y_i (x_0)$ für $x = x_0$ in einem einheitlichen Intervall d.

Programme-509 Tabellierung

Dieses Programm tabelliert in zwei Dimensionen die Matrix a_{ij}, i=0(l)m, j=0(l)n. (Wenn nicht alle Säulen in eine Papierbreite hineinpassen, werden die überzähligen Säulen unter die vorherigen Säulen geschrieben).

Programme-510 Lösung von Normalgleichungen kleinster Quadrate

Gegeben sei ein System von m linearen Gleichungen mit n Unbekannten (m>n)

$$A\,x = b,$$

wobei A = (a ij), x = (xj), und b = (bí), i = l(l)m, j = l(l)n. Dieses Programm bildet und berechnet die Normalgleichungen

$$A'Ax = A'b.$$

Programme-511 Eigenvektoren und Eigenwerte einer allgemeinen reellen Matrix

Dieses Programm berechnet die Eigenwerte und Eigenvektoren einer Matrix A, die im großen Speicher abgespeichert ist.

Programme-512 Ein-/Ausgabe von Double-Precision-Zahlen

Dieses Programm liest Gleitkommazahlen aus dem Lochstreifen ab und speichert sie in aufeinanderfolgenden Zellenpaaren des großen Speichers ab. Es locht auch Double-Precision-Zahlen mit Festpunkt, die mit gerundetem Dezimalteil in aufeinanderfolgenden Zellenpaaren des großen Speichers abgespeichert sind.

Programme-515 Lösung eines linearen Gleichungssystems mit Double-Precision
Dieses Programm löst ein lineares Gleichungssystem

$$a_{ij}\, x_j = b_i \qquad i = l(l)n,$$

wobei a_{ij} und b_i mit Double-Precision gegeben sind. Die Gleichungen werden zeilenweise abgespeichert, indem die b_i nach den a_{in} gestellt werden.

Programme-516 Tabellierung der Lösung eines System von Differentialgleichungen
Dieses Programm ist eine geänderte Version des programme-508.

Programme-518 Division von Matrizen
Dieses Programm berechnet das Gleiche wie die Funktion ψ 28, d. h. es prüft

$$A = B^{-1} A.$$

Der einzige Vorteil des Programms liegt darin, dass nicht alle Subroutinen von Erweiterungen für Matrizen gelesen werden müssen.

Programme-519 Reelle uneigentliche Quadratur
Dieses Programm berechnet das uneigentliche Integral I, wie den Grenzwert

$$I = \int_a^b f(x)dx = \lim_{n\to\infty} \sum_{r=0}^{n} \int_{x_r}^{x_{r+1}} f(x)dx$$

wobei $x_n = 2^{-n}(a-b) + b$.

Programme-521 Komplexe uneigentliche Quadratur
Das ist eine Version des Programme-519, das komplexe Werte des Integranden, komplexe Grenzwerte und eine komplexe Definitionslücke/Singularität akzeptiert. Die Toleranz wird auf das Modulo der Ergebnisse angewendet.

Programme-522 Texte Ein-/Ausgabe
Dieses Programm liest Texte von einem Lochstreifen, speichert sie ab und locht sie in gewünschter Form.

Programme-523 Kleinste-Quadrate-Polynom
Dieses Kapitel passt durch die Methode der kleinsten Quadrate ein Polynom mit Grad k an n Punkte an, die nicht unbedingt das gleiche Gewicht haben oder auf gleiche Weise getrennt sind.

3. Makroprogramme.

Als Zusammenfassung behandeln wir schließlich die allgemeine Organisation eines Berechnungsverfahrens, das mehrere Unterprogramme enthält, d. h. ein Makroprogramm. Dafür weisen wir darauf hin, dass es in folgenden vier Schritten auszuführen ist:

a – Ein Blocksdiagramm erstellen, in dem man die wesentlichen Teile des Berechnungsverfahrens, die auszuführen sind, angibt

b – Die Teile, die sich mit Hilfe der Programmbibliothek behandeln lassen, erkennen und ihre Spezifikationen verstehen

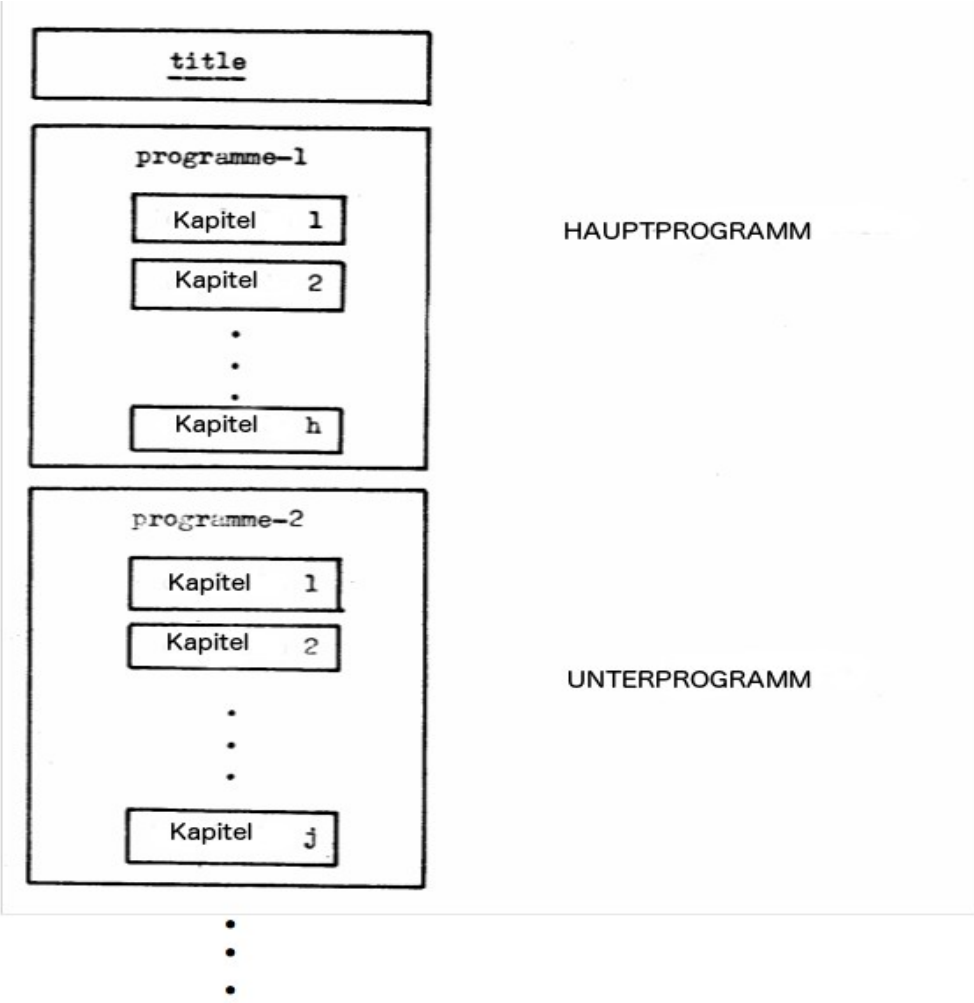
c – Programme für die anderen Teile, für die es kein geeignetes Programm in der Bibliothek gibt, schreiben. Dafür ist es sinnvoll, ein Detail-Diagramm zu erstellen.

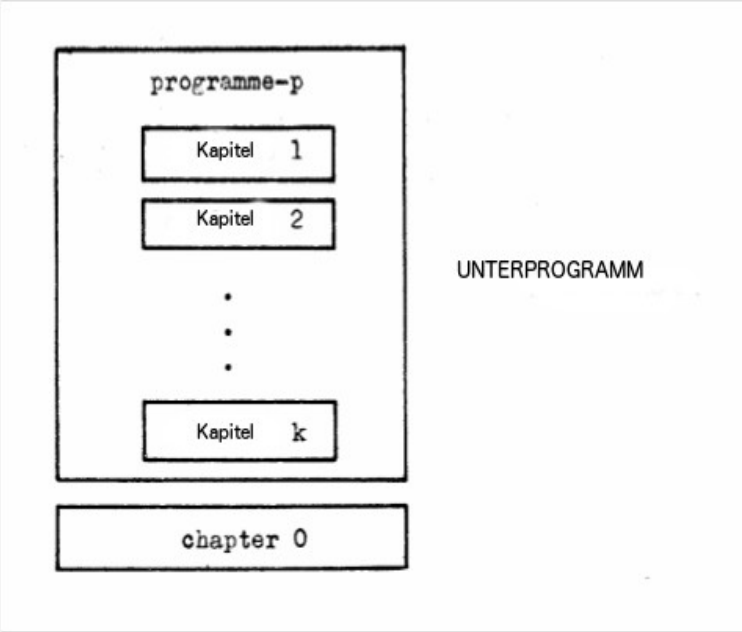
d - Das Verknüpfungsprogramm bzw. das Hauptprogramm schreiben und dabei beachten, wie die Dateneingabe/-ausgabe umgesetzt werden muss. Eigentlich entspricht das Hauptprogramm dem in a) erwähnten Blocksdiagramm.

Einige Ausnahmen bei der Benutzung einiger Direktiven und Sprungbefehle in den Makroprogrammen müssen beachtet werden:

- die Direktive variables kann sich nur auf in einem gleichen Unterprogramm enthaltende Kapitel beziehen
- die Sprungbefehle zwischen Kapiteln, die zu verschiedenen Unterprogrammen gehören, müssen drei Parameter haben: die Marke, das Kapitel, das Programm. Sie nehmen die Form across M/C – P, down M/C – P an.

Die funktionelle Struktur eines Unterprogrammes ist im Allgemeinen wie im folgenden Schema gezeigt:





Im Folgenden wird durch ein Beispiel gezeigt, wie ein unvollständiges Unterprogramm aus der Bibliothek zu benutzen ist. Es geht darum, durch das Programm A-502 die Funktion $y = \sin^2 x$ im Intervall $(0, \pi/2)$ zu integrieren. Dafür muss man die Funktion spezifizieren, die man am Ende des Programm-502 - das aus diesem Zweck nicht abgeschlossen ist - integrieren will. Vor der Programmausführung müssen seine Parameter definiert werden. Laut den Spezifikationen müssen in A und B die Integrationsgrenzen, in E die angeforderte oder erreichte Präzision, in u' die Adresse des großen Speichers, in dem der Wert des Integrals abzuspeichern ist, gestellt werden. Der Aufruf wird durch den Befehl down 1/1 – 502 ausgeführt. Da wir in diesem Fall annehmen, dass unser Ziel nur die Berechnung des Integralwerts ist, fügen wir zusätzlich Ausgabebefehle hinzu. Allerdings ist der Integralwert bei allgemeineren Problemen ein Zwischenwert und deswegen ist sein Ausdruck nicht nötig. Folgende Befehle zeigen schematisch, wie wir vorgehen würden:

```
TITLE
INTEGRAL SINUS-QUADRAT


PROGRAMME - 502
CHAPTER 1

1)----------
----------
----------
----------
101)--------
----------
----------
----------
----------
UP
----------
----------
120)570,0
X=xSIN(X)
X=XX
JUMP101


CLOSE




CHAPTER 0
X→1


A=0
B=£/2
E=.0001
Un=0
DOWN 1/1-502
X6 (Un) X0,2
PRINT (X0) 2,4
PRINT (X1) 0,4


CLOSE
```

ANHANG I

ANMERKUNGEN ZU DEN SPEICHERN

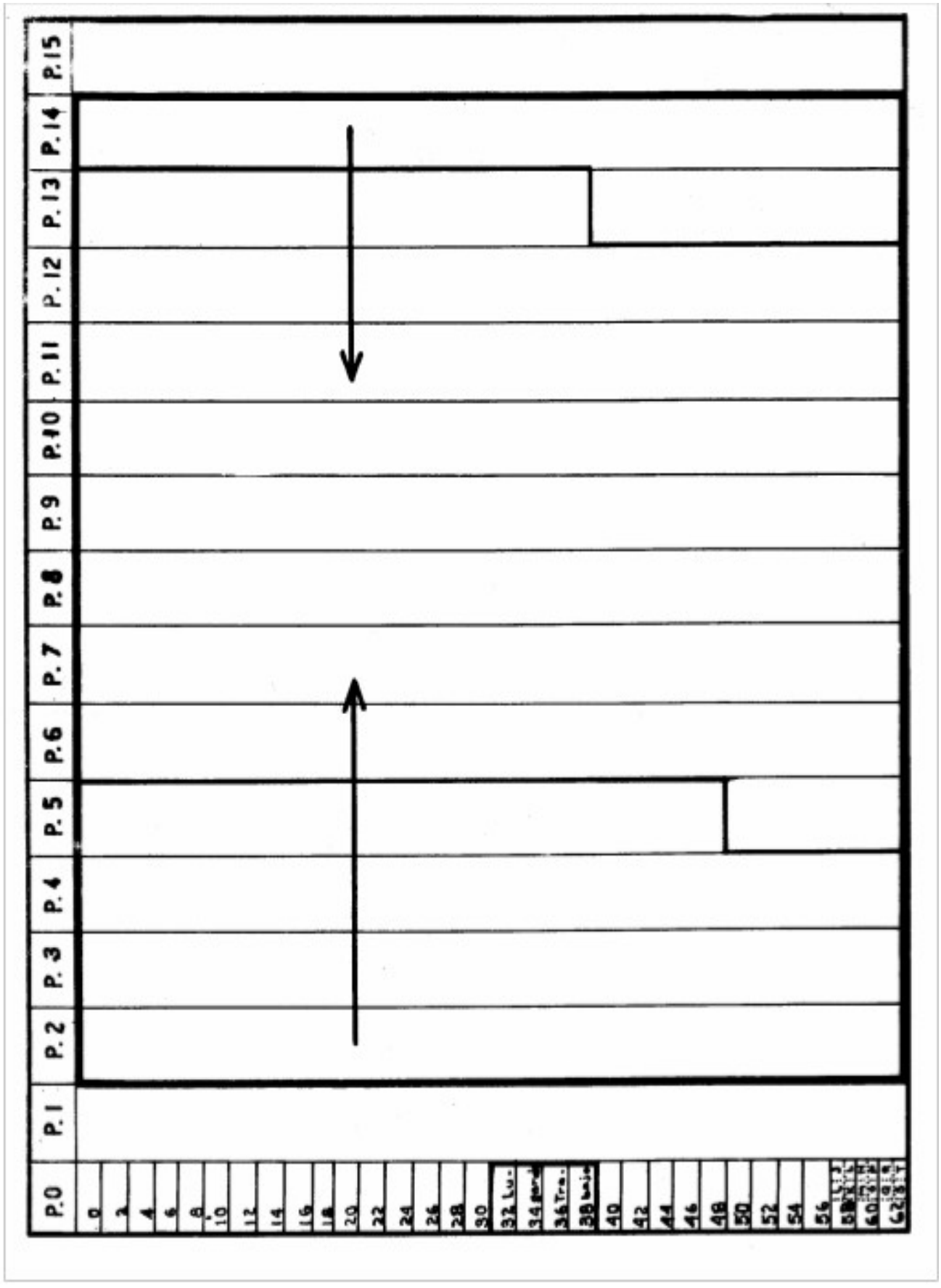
1. DER SCHNELLE SPEICHER.

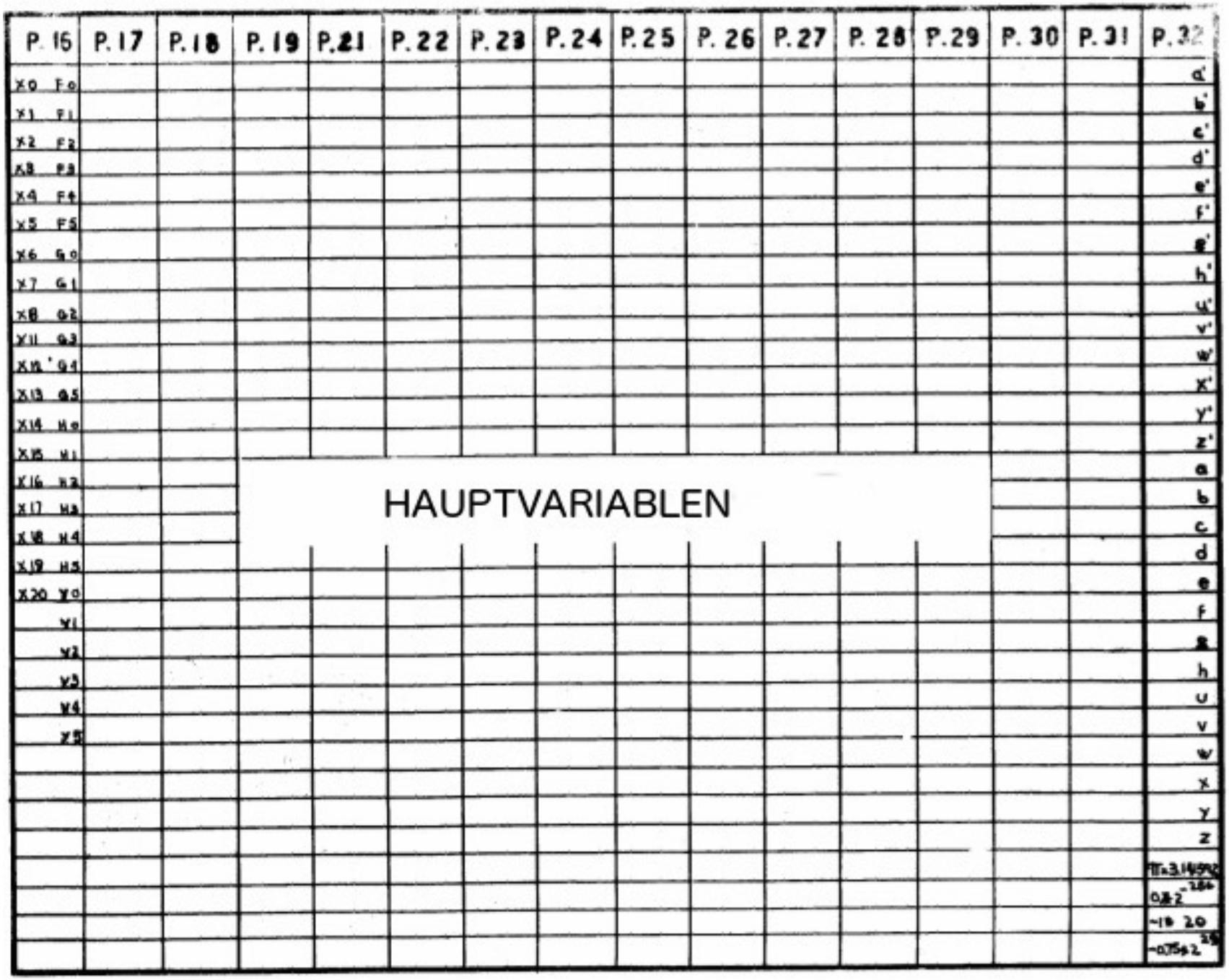
Wir haben vorher gesehen, dass der aktuelle oder schnelle Speicher in zwei verschiedenen Teilen – einen für die Befehle und einen für die Zahlen- betrachtet werden kann. Jetzt schauen wir im Detail an, wie beide physisch geteilt sind.
Der Befehlsspeicher besteht aus 16 gleichen Teilen, die Seiten genannt werden und von 0 bis 15 nummeriert sind. Jede Seite setzt sich folgendermaßen zusammen:

- aus 32 Registern, deren Speichervermögen einer Zahl aus 10 Dezimalziffern (40 Binärziffern) entspricht, die mit geraden Zahlen von 0 bis 62 nummeriert sind
- aus 64 Registern mit Speichervermögen für 20 Binärziffern, die einzeln von 0 bis 63 nummeriert sind
- aus 128 Registern mit 10 Binärziffern, die durch die Symbole 0, 0+, 1, 1+,... , 63, 63+ aufsteigend nummeriert sind.

Daher kann man sagen, dass die Indizes die Register 58, 58+, 59, 59+, 60, 60+, 61, 61+, 62, 62+, 63, 63+ der Seite 0 ausfüllen, wie in der Abbildung 4 dargestellt ist. Das gleiche Bild zeigt außerdem, dass die Register 4-31 die Divisionbefehle enthalten und die Register 32, 34, 36, 38 als Arbeitsplatz benutzt werden können (s. Benutzung von konventionellen Befehlen in Autocode). Die Seite 1 ist für Sonderübertragungen für das Ziehen einer komplexen Quadratwurzel reserviert. Die Seite 15 wird benutzt, um eine Funktion, die kein Quickie ist, abzuspeichern. Auf den Seiten 2 bis 14 wird die einem unserer Kapitel entsprechende übersetzte Information wie folgt abgespeichert*:* ab dem Register 0 der 2. Seite, die Befehle; ab dem Register 62 der Seite 14 rückwärts, die Konstanten, die ausdrücklich in unserem Kapitel dargestellt sind. Falls es genügend Platz geben würde, befinden sich Quickies-Funktionen im restlichen Platz zwischen dem letzten Befehl und der letzten Konstante.

Der Zahlenspeicher besteht aus 16 Seiten, die von 16 bis 31 nummeriert sind. Die Sondervariablen haben auf der Seite 31 ihren festen Platz. Die Register 0, 2, 4, … 54 entsprechen den Variablen a' b' … z. Wenn sie nicht verändert wird, enthält die Variable π die Konstante 3.141592 und ist im Register 56 abgespeichert. In den Registern 58, 60, 62 werden die Konstante 0, -1, -4.10^8 abgespeichert.
Folgendes ist anzumerken: Wenn ein preserve Befehl gezeigt wird, werden die Indizes die Register 58, 58+, … 63+ der Seite 31 ausfüllen, bevor der ganze Inhalt vom Zahlenspeicher zum großen Speicher übertragen wird.
Die Seiten 16 bis 30 enthalten die Sondervariablen in der Reihenfolge, die vor den Befehlen jedes Kapitels durch die Direktive → bestimmt wurde. In der Abbildung 5 sind links vom Register der Seite 16 die Stellen dargestellt, die den Variablen des Beispiels auf Seite 33 entsprechen. Rechts kann man erkennen, dass die gleichen Stellen den ersten zwanzig reservierten Variablen des Beispiels auf Seite 35 entsprechen.


P.0
P.1
P.2
P.3
P.4
P.5
P.6
P.7
P.8
P.9
P.10
P.11
P.12
P.13
P.14
P.15
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32 Lu.
34 Ma.
36 Tra.
38 Mer.
40
42
44
46
48
50
52
54
56


P. 16
P. 17
P. 18
P. 19
P. 21
P. 22
P. 23
P. 24
P. 25
P. 26
P. 27
P. 28
P. 29
P. 30
P. 31
P. 32
X0 F0
X1 F1
X2 F2
X3 F3
X4 F4
X5 F5
X6 G0
X7 G1
X8 G2
X11 G3
X12 G4
X13 G5
X14 H0
X15 H1
X16 H2
X17 H3
X18 H4
X19 H5
X20 Y0
Y1
Y2
Y3
Y4
Y5
a'
b'
c'
d'
e'
f'
g'
h'
u'
v'
w'
x'
y'
z'
a
b
c
d
e
f
g
h
u
v
w
x
y
z
HAUPTVARIABLEN

2. DER GROSSE SPEICHER.

In I.2., in II.1.2 und in IV wurde der große Speicher im Rahmen der grundsätzlichen Nutzung von Hilfsvariablen ausführlich behandelt. In diesem Anhang werden die verschiedenen Teile genauer erläutert, um ihn besser zu gebrauchen, sowie um bestimmte manuelle Operationen (wie in Anhang 2) zu erleichtern. Darüber hinaus wird der Programmierer auf diese Weise informiert, wie der Speicher genau benutzt wird.

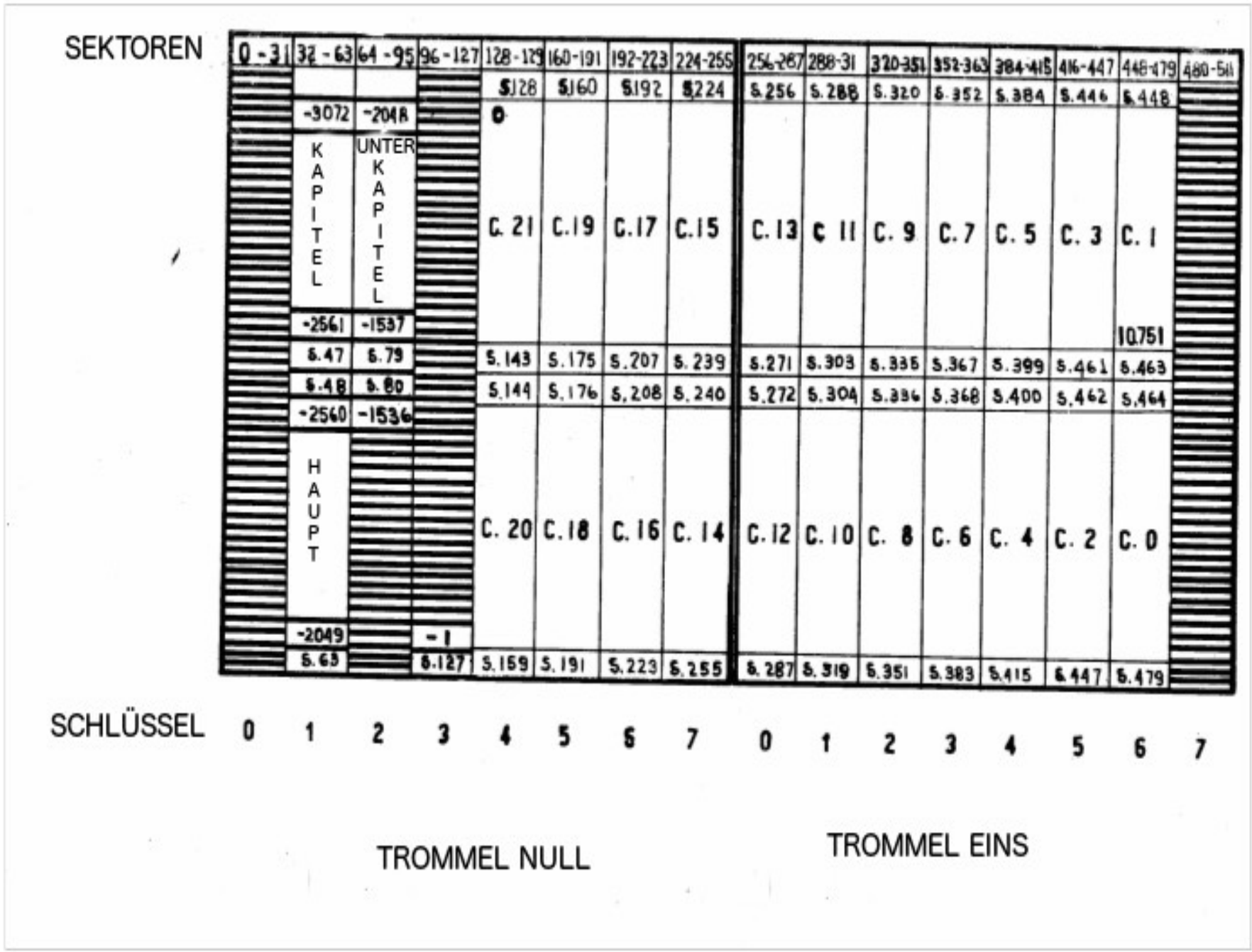
1. - Physische Beschreibung: Der große Speicher besteht aus zwei Magnettrommeln, die Trommel 0 (Drum 0) und Trommel 1 (Drum 1) genannt werden. Jede Trommel ist in acht Bereiche (von 0 bis 7) eingeteilt und jeder Bereich besteht aus 32 Sektoren. Die Sektoren sind vom Bereich 0 der Trommel 0 bis zum Bereich 7 der Trommel 1 von 0 bis 511 durchnummeriert. Die Sektoren des Bereichs 2 der Trommel 1 würden sich zum Beispiel zwischen dem Sektor 320 und dem Sektor 351 befinden.

Jeder Bereich kann durch einen Isolationsschlüssel, der die gleiche Zahl des Bereichs hat (s. Abb. 6), isoliert werden. Diese Isolation verhindert eine Übertragung vom schnellen Speicher zum isolierten Bereich des großen Speichers. Eine Übertragung in die umgekehrte Richtung ist trotzdem möglich. Damit wird sichergestellt, dass der isolierte Bereich gegen Programmierungs- bzw. Ausführungsfehler geschützt wird. Individuelle Sektoren können nicht isoliert werden. In jeder Trommel gibt es eine Glimmlampe, das bei einer Übertragung angeschaltet und nicht ausgeschaltet wird, solange keine Übertragung über die andere Trommel erfolgt. Dadurch erkennt man, welche Trommel als letzte operiert hat.

2. - Funktion. Der große Speicher hat drei Hauptfunktionen:

1. - Abspeicherung des Übersetzungsprogramms AUTOCODE INPUT
2. - Abspeicherung der verschiedenen Kapitel des übersetzten Programms
3. - Abspeicherung der Zwischenwerte der Berechnung

2.1. Das Übersetzungsprogramm besteht aus zwei Teilen: eine wird nur während der Übersetzungsphase und der andere auch während der Ausführungsphase (z. B. die Subroutine für die Berechnung der Elementarfunktionen) gebraucht. Der erste Teil ist in den Sektoren 80 bis 127 und der zweite in den Sektoren 0 bis 31 gespeichert. Beide füllen zusammen die Bereiche 0 und 3 und den Bereich 2 nur partiell aus. Die Routine für die Erweiterung für Matrizen, die in den Sektoren 480 bis 511 gespeichert werden, können als Teil des AUTOCODE INPUTS betrachtet werden.


SEKTOREN
0 - 31
32 - 63
64 - 95
96 - 127
128 - 129
160 - 191
192 - 223
224 - 255
256 - 287
288 - 31
320 - 351
352 - 363
384 - 415
416 - 447
448 - 479
480 - 511
S.128
S.160
S.192
S.224
S.256
S.288
S.320
S.352
S.384
S.416
S.448
-3072
-2048
0
KAPITEL
UNTER KAPITEL
C. 21
C. 19
C. 17
C. 15
C. 13
C. 11
C. 9
C. 7
C. 5
C. 3
C. 1
-2561
-1537
10751
S. 47
S. 79
S. 143
S. 175
S. 207
S. 239
S. 271
S. 303
S. 335
S. 367
S. 399
S. 461
S. 463
S. 48
S. 80
S. 144
S. 176
S. 208
S. 240
S. 272
S. 304
S. 336
S. 368
S. 400
S. 462
S. 464
-2560
-1536
HAUPT
C. 20
C. 18
C. 16
C. 14
C. 12
C. 10
C. 8
C. 6
C. 4
C. 2
C. 0
-2049
-1
S. 63
S. 127
S. 159
S. 191
S. 223
S. 255
S. 287
S. 319
S. 351
S. 383
S. 415
S. 447
S. 479
SCHLÜSSEL
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
TROMMEL NULL
TROMMEL EINS

2.2. Die übersetzten Kapitel werden im großen Speicher ihrer Nummer gemäß abgespeichert. Auf diese Weise entsprechen dem Kapitel 0 die Sektoren 464 bis 479, dem Kapitel 1 die Sektoren 448 bis 463 und dem Kapitel 21 (am möglichen Maximum) die Sektoren 128 bis 143. Unabhängig von der Lesereihenfolge der Kapitel werden sie immer an der zu ihrer Nummer passenden Stelle abgespeichert. Wegen der Einzigartigkeit des Kapitels 0, dessen close den Beginn der Berechnungen auslöst, muss dieses immer an der letzten Stelle gelesen werden. Die Makros, die aus mehreren Programmen bestehen, können mehrere Kapitel mit der gleichen Nummer enthalten. Die Stelle der Kapitel des ersten Programms entspricht genau der oben genannten Beschreibung. Die Stelle des nächsten Programms wird berechnet, indem man das Kapitel 1 als folgendes zu dem zuletzt gelesenen Kapitel des vorigen Programms betrachtet. Es existiert nur ein Kapitel 0, das immer die gleiche Stelle ausfüllt. Der große Speicher kann maximal 22 Kapitel (inkl. Kapitel 0) enthalten.

2.3. Die Stelle der Hilfsvariablen interessiert uns am meisten, um unser Programm besser organisieren zu können und die Zerstörung einiger Kapitel durch die Übereinanderstellung von Zahlen zu verhindern. Wie wir bereits erfahren haben, besitzen die Variablen numerische Adressen, die sich zwischen 0 und 10751 und zwischen -1 und -3072 befinden. Die Variablen 0 bis 10751 nehmen die gleichen Bereiche ein wie die Kapitel 21 bis 1 (s. Abb. 5). Die Variablen -1 bis -3072 werden je nach Adressbereich in vier Gruppen geteilt. Die erste (-1 bis -1536) zerstört bei der Benutzung das in 2.1. erwähnte Übersetzungsprogramm; die Gruppe von -1537 bis -2048 ist bei preserve Befehl auf dem Niveau von Unterkapiteln (2. Ebene) nicht zu verwenden; die Gruppe von -2049 bis -2560 darf nicht benutzt werden, wenn der Befehl preserve sich auf Kapitel-Niveau (1. Ebene) befindet; die Gruppe von -2561 bis -3072 ist auszuschließen, falls Erweiterungen für Matrizen benutzt werden bzw. es im Programm den Befehl rmp gibt.
Diese letzten Beschränkungen hängen mit der Existenz von drei Speicherbereichen zusammen. Zwei von ihnen werden MASTER und UNTERKAPITEL genannt und belegen jeweils die Sektoren von 48 bis 63 und 64 bis 79. Sie speichern den Inhalt des aktuellen Speichers von Zahlen, die durch den Befehl preserve geordnet werden. Das dritte, das Sonderspeicherbereich genannt wird, speichert den Inhalt des aktuellen Befehlsspeichers während der Anwendung von Erweiterungen für Matrizen oder während einer neuen durch rmp ausgelösten Übersetzung ab.

Die Abbildung zeigt in schematischer Form jeden Hauptteil des großen Speichers. Ihre Anwendung empfiehlt sich bei der Organisation von Makroprogrammen, die mehrere Programme und Hilfsvariablen bzw. die Befehle down und preserve verwenden. Tabelle I gibt die Zahl der ersten Hilfsvariable jedes Sektors sowie die nach Bereich gruppierten Sektoren an. Diese Tabelle ist nützlich, weil sie sich auf die Bereiche bezieht, die wir isolieren, sowie auf die Sektoren, die wir durch post-mortem oder andere Erweiterungen drucken möchten.

TABELLE I

Die ungeraden Spalten zeigen die Nummern der Sektoren an. Die geraden Spalten weisen die Nummer der ersten Hilfsvariablen des zu ihrer Linken angezeigten Sektors auf. Acht Reihen bilden zusammen einen Bereich, dessen Nummer an seinem Kopf anzuzeigen ist. Diese Nummern stimmen mit denjenigen der entsprechenden Isolationsschlüssel überein.

TROMMEL 0

1

32	−3072	40	−2816	48	−2560	56	−2304
33	−3040	41	−2784	49	−2528	57	−2272
34	−3008	42	−2752	50	−2496	58	−2240
35	−2976	43	−2720	51	−2464	59	−2208
36	−2944	44	−2688	52	−2432	60	−2176
37	−2912	45	−2656	53	−2400	61	−2144
38	−2880	46	−2624	54	−2368	62	−2112
39	−2848	47	−2592	55	−2336	63	−2080

2

64	−2048	72	−1792	80	−1536	88	−1280
65	−2016	73	−1760	81	−1504	89	−1248
66	−1984	74	−1728	82	−1472	90	−1216
67	−1952	75	−1696	83	−1440	91	−1184
68	−1920	76	−1664	84	−1408	92	−1152
69	−1888	77	−1632	85	−1376	93	−1120
70	−1856	78	−1600	86	−1344	94	−1088
71	−1824	79	−1568	87	−1312	95	−1056

3

96	−1024	104	−768	112	−512	120	−256
97	−992	105	−736	113	−480	121	−224
98	−960	106	−704	114	−448	122	−192
99	−928	107	−672	115	−416	123	−160
100	−896	108	−640	116	−384	124	−128
101	−864	109	−608	117	−352	125	−96
102	−832	110	−576	118	−320	126	−64
103	−800	111	−544	119	−288	127	−32

4

128	0	136	256	144	512	152	768
129	32	137	288	145	544	153	800
130	64	138	320	146	576	154	832
131	96	139	352	147	608	155	864
132	128	140	384	148	640	156	896
133	160	141	416	149	672	157	928
134	192	142	448	150	704	158	960
135	224	143	480	151	736	159	992

5

160	1024	168	1280	176	1536	184	1792
161	1056	169	1312	177	1568	185	1824
162	1088	170	1344	178	1600	186	1856
163	1120	171	1376	179	1632	187	1888
164	1152	172	1408	180	1664	188	1920
165	1184	173	1440	181	1696	189	1952
166	1216	174	1472	182	1728	190	1984
167	1248	175	1504	183	1760	191	2016

6

192	2048	200	2304	208	2560	216	2816
193	2080	201	2336	209	2592	217	2848
194	2112	202	2368	210	2624	218	2880
195	2144	203	2400	211	2656	219	2912
196	2176	304	2432	212	2688	220	2944
197	2208	205	2464	213	2720	221	2976
198	2240	206	2496	214	2752	222	3008
199	2272	207	2528	215	2784	223	3040

7

224	3072	232	3328	240	3584	248	3840
225	3104	233	3360	241	3616	249	3872
226	3136	234	3392	242	3648	250	3904
227	3168	235	3424	243	3680	251	3936
228	3200	236	3456	244	3712	252	3968
229	3232	237	3488	245	3744	253	4000
230	3264	238	3520	246	3776	254	4032
231	3296	239	3552	247	3808	255	4064

TROMMEL 1

0

256	4096	264	4352	273	4608	280	4864
257	4128	265	4384	273	4640	281	4896
258	4160	266	4416	274	4672	282	4928
259	4192	267	4448	275	4704	283	4960
260	4224	268	4480	276	4736	284	4992
261	4256	269	4512	277	4768	285	5024
262	4288	270	4544	278	4800	286	5056
263	4320	271	4576	279	4832	287	5088

1

288	5120	296	5376	304	5632	312	5888
289	5152	297	5408	305	5664	313	5920
290	5184	298	5440	306	5696	314	5952
291	216	299	5472	307	5728	315	5984
292	5248	300	5504	308	5760	316	6016
293	5280	301	5536	309	5792	317	6048
294	5312	302	5568	310	5824	318	6080
295	5344	303	5600	311	5856	319	6112

2

320	6144	328	6400	336	6656	344	6912
321	6176	329	6432	337	6688	345	6944
322	6208	330	6464	338	6720	346	6976
323	6240	331	6496	339	6752	347	7008
324	6272	332	6528	340	6784	348	7040
325	6304	333	6560	341	6816	349	7072
326	6336	334	6592	342	6848	350	7104.
327	6368	335	6624	343	6880	351	7136

3

252	7168	360	7424	368	7680	376	7936
353	7200	361	7456	369	7712	377	7968
354	7232	362	7488	370	7744	378	8000
355	7264	363	7520	371	7776	379	8032
356	7296	364	7552	372	7808	380	8064
357	7328	365	7584	373	7840	381	8096
358	7360	366	7616	374	7872	382	8128
359	7392	367	7648	375	7904	383	8160

4

384	8192	392	8448	400	8704	408	8960
385	8224	393	8480	401	8736	409	8992
386	8256	394	8512	402	8768	410	9024
387	8288	395	8544	403	8800	411	9056
388	8320	396	8576	404	8832	412	9088
389	8352	397	8608	405	8864	413	9120
390	8384	398	8640	406	8896	414	9152
391	8416	399	8672	407	8928	415	9184

5

416	9216	424	9472	432	9728	440	9984
417	9248	425	9504	433	9760	441	10016
418	9280	426	9536	434	9792	442	10048
419	9312	427	9568	435	9824	443	10080
420	9344	428	9600	436	9856	444	10112
421	9376	429	9633	437	9888	445	10144
422	9408	430	9664	438	9920	446	10176
423	9440	431	9696	439	9952	447	10208

6

448	10240	456	10496
449	10272	457	10528
450	10304	458	10560
451	10336	459	10592
452	10368	460	10624
453	10400	461	10656
454	10432	462	10688
455	10464	463	10720

ANHANG 2

KONVENTIONELLE BEFEHLE, DIE VON AUTOCODE AKZEPTIERT WERDEN.

Zur Erhöhung ihrer Flexibilität kann die Autocode-Sprache in konventionelle Sprache[1] geschriebene Befehle mit einigen Varianten akzeptieren. Die Varianten lassen sich wie folgt unterscheiden, je nachdem, ob die verwendeten Adressen (Variablen, Indizes oder Marken) in konventioneller Sprache oder in Autocode geschrieben sind.

1. Adresse in Autocode.

Wenn der Operand ein von Autocode verwendeter Index, eine Variable oder eine Marke ist, wären die Befehle in konventioneller Sprache wie folgt:

FB(α)	α = eine beliebige Hauptvariable, Sondervariable oder Konstante
FB(i)	i = ein Index oder eine Ganzzahl
FB(m)	m = eine Marke

Im ersten Fall geht es um Operationen mit großen Zahlen (40 bits), die auf konventionellem Code mit den Zahlen 40 bis 45 und 50 bis 55 bezeichnet werden. Bei Gebrauch von Variablen mit variablem Index (wie x_i, $y_{(j+1)}$) muss das B-Zeichen gleich Null sein.
Im zweiten Fall sind die anwendbare Codes durch die Zahlen 00 bis 07 und 20 bis 27 angezeigt.
Im dritten Fall kann die Funktion eine beliebige Sprungfunktion (jump-Funktion) sein.

Beispiele:

a) Folgende Ausdrücke sind möglich:

411(x)
422(z_s)
510(x_i)
200(k)

b) Um die Werte des Polynoms

$$y = a_0 x^{10} + a_1 x^9 + \ldots + a_{10}$$

---

1 Diese Anmerkungen dienen denjeniegen, die schon die konventionelle Sprache kennen. Wir möchten daran erinnern, dass der Befehl lautet:

F-B D

F, das durch zwei Dezimalziffern ausgedrückt wird, stellt die Zielfunktion dar. B ist durch eine einzige Dezimalziffer ausgedrückt und stellt das Register B dar, das am Befehl teilnimmt. D (das auf unterschiedliche Weise dargestellt werden kann) ist die Adresse des Operanden.

zu berechnen, geht man folgendermaßen vor:

400(a_0)
i = 1 (1) 10
500(x)
420(a_i)

repeat
410(y)

2. Adresse in konventioneller Sprache.

Der Teil des Befehls, der im vorigem Abschnitt in Klammern ausgedrückt wurde, kann nur Adresse zeigen. Deshalb ist es nicht möglich, die besagte Konvention für diejenigen Befehle anzuwenden, in denen diese Ziffer eine Konstante zeigt. Die folgende Konvention lässt sich bei jeglichem Befehl anwenden:

FB, α α = eine Ganzzahl oder eine absolute Adresse

Die Art der Adresse Seiten-Register (2.16 +) und die Symbole ≠ und = können benutzt werden. Floating bzw. relative Adresse sind nicht zulässig.

Beispiele:

a) Sind zulässig

321, 28
400, 2
512, 2.16
590, 1.0

Sind nicht zulässig

400, v1
430, 2x1

b) Um das Polynom

$y = a_0 x^{10} + a_1 x^9 + \ldots + a_{10}$

zu berechnen, geht man folgendermaßen vor:

400(a0)
300, – 9
1) 500(x)
427(a10)
380(i)
410(u)

3. Anwendung von Erweiterungen in konventioneller Sprache für die Ausgabe.

Eine der wichtigsten Anwendungen ist das Erreichen einer Flexibilität in der Ausgabe bei der Anwendung vom Befehl 620, n. Dieser Befehl locht im Ergebnis-Lochstreifen das durch n bezeichnete Zeichen (s. Tabelle II), in dem n der Wert jeder Lochung in Binärzahlen zeigt. Es wird laut angehängter Tabelle erläutert. Es ist wichtig, sich die Bedeutung der Zeichen 0 (FS) und 27 (LS) zu merken, denn sie müssen vor jeder Zahlengruppe (oder vor Sonderzeichen) oder vor jeder Buchstabengruppe gelocht werden.

Beispiel: Wenn man während des Verfahrens die Ergebnisse auf diese Weise

CASO N *[Fall N]*
a1 = entsprechender Wert a2 = entsprechender Wert

schreiben möchte, schreibt man

```
newline
620,27   Buchst. folgen
620,3      c
620,1      a
620,19     s
620, 5     e
620,0    Ziffern folgen
620,14   Leerzeichen
620,14   Leerzeichen

print (u) 2,0
newline
620,27   Buchst. folgen
620,1      a
620,0    Zahlen folgen
620,1      1
620,10     =
print (a1)2,4
620,27   Buchst. folgen
620,1      a
620,0    Zahlen folgen
620,2      2
620,10     =
print (a2) 2,4
```

LOCH-STREIFEN	WERT	DRUCKER	
		ZIFFERN	BUCHST.
.	0	ABB.	SHIFT.
. ●	1	1	A
.●	2	2	B
.●●	3	*	C
●.	4	4	D
●. ●	5	(	E
●.●	6	)	F
●.●●	7	7	G
● .	8	8	H
● . ●	9	≠	I
● .●	10	=	J
● .●●	11	-	K
●●.	12	v	L
●●. ●	13	LF	M
●●.●	14	SP	N
●●.●●	15	,	O
● .	16	0	P
● . ●	17	>	Q
● .●	18	>	R
● .●●	19	3	S
● ●.	20	→	T
● ●. ●	21	5	U
● ●.●	22	6	V
● ●.●●	23	/	W
●● .	24	x	X
●● . ●	25	9	Y
●● .●	26	+	Z
●● .●●	27	LET.	SHIFT.
●●●.	28	.	.
●●●. ●	29	n	?
●●●.●	30	CR	£
●●●.●●	31	ER	ER

TABELLE II

Inhaltsverzeichnis