FACHSCHUL–
FERNSTUDIUM

Jürgen Hämmerlein

GRUNDLAGEN DER INFORMATIONSVERARBEITUNG FÜR INGENIEURE

1

Grundlagen

Herausgeber:
Institut für Fachschulwesen der
Deutschen Demokratischen Republik
Karl-Marx-Stadt

03 0160 01 0

Dieser Lehrbrief wurde
verfaßt von:

Fachlehrer Jürgen H ä m m e r l e i n
Zentralstelle für Bildung im
VE Kombinat Datenverarbeitung Berlin

lektoriert von:

FSD Dipl.-Ing.-Ök. Dieter G r ü l l i c h
Ingenieurschule Wildau

FSD Dipl.-Ök. Rudi S c h m i d t
Ingenieurschule für Bergbau und Energetik
„Ernst Thälmann" Senftenberg

bearbeitet von:

FSD Dipl.-Phys. Werner S e i f e r t
Institut für Fachschulwesen Karl-Marx-Stadt

Als Manuskript gedruckt • Alle Rechte vorbehalten
Veröffentlicht:
INSTITUT FÜR FACHSCHULWESEN DER DEUTSCHEN DEMOKRATISCHEN REPUBLIK
Karl-Marx-Stadt

Druck und buchbinderische Verarbeitung:

ZENTRALSTELLE FÜR LEHR- UND ORGANISATIONSMITTEL DES MINISTERIUMS FÜR HOCH- UND FACHSCHULWESEN, ZWICKAU
Ag 613/4021/84/4000 1. Ausgabe 7. Auflage
Vorzugsschutzgebühr: 1,50 M

Inhaltsverzeichnis

1. Die elektronische Datenverarbeitung – ein Mittel zur Intensivierung und Rationalisierung

Die Hauptaufgabe der entwickelten sozialistischen Gesellschaft besteht in der weiteren Erhöhung des materiellen und kulturellen Lebensniveaus des Volkes. Die Grundlage dafür bilden ein hohes Entwicklungstempo der sozialistischen Produktion, die Erhöhung der Effektivität, der wissenschaftlich-technische Fortschritt und das Wachstum der Arbeitsproduktivität. Dazu bedarf es der schöpferischen Tätigkeit aller Werktätigen und ihrer bewußten Mitwirkung zur Lösung der grundlegenden Fragen:

- Steigerung des Nationaleinkommens,
- Beschleunigung des wissenschaftlich-technischen Fortschritts,
- unbedingte Erfüllung der Außenwirtschaftsaufgaben,
- Beherrschung herangereifter Fragen der Leitung und Planung,
- allseitige Verwirklichung des Sparsamkeitsprinzips.

In dem Maße, wie die erweiterte Reproduktion voranschreitet, nimmt das Gewicht der vergegenständlichten Arbeit zu. Die bessere Ausnutzung des in den Arbeitsmitteln und Arbeitsgegenständen verkörperten gesellschaftlichen Reichtums wird für die Erhöhung der Effektivität des gesamten Reproduktionsprozesses immer bedeutsamer.

Mit der gleichen oder einer geringeren Anzahl von Arbeitskräften durch Modernisierung und bessere Ausnutzung der vorhandenen Grundfonds mehr zu produzieren – das heißt Intensivierung. Sie macht es notwendig, jede Aufgabe und jedes Arbeitsergebnis mit dem strengen Maßstab des volkswirtschaftlichen Nutzens zu messen. Dazu ist es erforderlich, in weitaus stärkerem Maße die Faktoren für das Wachstum der Volkswirtschaft zu erschließen. Der wissenschaftlich-technische Fortschritt zählt zu den entscheidenden Grundlagen der Intensivierung und ist gleichzeitig ihre wichtigste Voraussetzung. Das Niveau der vorhandenen Produktionsanlagen wird durch neue wissenschaftlich-technische Erkenntnisse und ihre Anwendung weiterentwickelt. Damit werden die Produktionsanlagen intensiver im Reproduktionsprozeß wirksam.
Der wissenschaftlich-technische Fortschritt leistet nicht nur selbst den Hauptbeitrag zur Steigerung der Effektivität der Produktion und damit zum Leistungsanstieg unserer Volkswirtschaft, er durchdringt auch in zunehmendem Maße alle Faktoren des Wirtschaftswachstums, wie die rationelle Gestaltung der gesellschaftlichen Organisation der Produktion, die sozialistische ökonomische Integration, die steigende Bildung und Qualifikation der Werktätigen.

Die Produktivkraft der menschlichen Arbeit wird in entscheidendem Maße von den materiell-gegenständlichen Elementen der Produktion, vor allem den

Arbeitsmitteln, bestimmt. Arbeitsmittel und Arbeitsgegenstände bilden zusammen mit den durch sie bedingten technologischen Verfahren, Strukturen und Organisationsformen der Produktion sowie mit den materiell-technischen Grundlagen der nichtproduzierenden Bereiche die materiell-technische Basis der Gesellschaft. Die Anwendung der elektronischen Datenverarbeitung, als Teil der materiell-technischen Basis, bei der Beschleunigung des wissenschaftlich-technischen Fortschritts muß dabei unter zwei Gesichtspunkten betrachtet werden:

- In den vergangenen Jahren und auch in der zukünftigen Planperiode wurden und werden umfangreiche Mittel für den Einsatz der elektronischen Datenverarbeitungsanlagen bereitgestellt. Der Einsatz der vorhandenen EDVA brachte für die Volkswirtschaft schon zahlreiche Erfolge. Viele EDVA sind jedoch noch nicht durch Aufgaben ausgelastet, die ihrer vollen Leistungsfähigkeit entsprechen.
- Die Wissenschaft wird immer stärker zur Produktivkraft.
  Ein wesentliches Element der wissenschaftlichen Arbeit ist aber die Verarbeitung von Informationen. Gerade hier kann der Einsatz von EDVA zu einer bedeutenden Befreiung der Menschen von Routinearbeiten und damit zur Steigerung der Effektivität der wissenschaftlichen Arbeit führen.

Die entsprechende Qualifizierung der am Einsatz der elektronischen Datenverarbeitungsanlagen beteiligten Werktätigen ist bei beiden Problemen eine entscheidende Bedingung für die Verbesserung der Situation. Die gesellschaftliche Arbeitsteilung führte beim Einsatz der EDV zu zwei Gruppen von Werktätigen, den unmittelbar in der Datenverarbeitung Beschäftigten und den Nutzern der EDV im Reproduktionsprozeß, die diesen Einsatz mitbestimmen. Nur die gemeinsame Beschäftigung mit den Problemen ist der Schlüssel für den optimalen Einsatz des durch den erreichten Stand der EDV repräsentierten Teils unseres gesellschaftlichen Reichtums. D. h., die unmittelbar in der Datenverarbeitung Tätigen müssen sich mit dem Reproduktionsprozeß und die im Reproduktionsprozeß Tätigen müssen sich mit den Möglichkeiten der EDV intensiver beschäftigen. Die Lösung dieser Aufgabe ist notwendig, weil die Beherrschung informationeller Prozesse nicht nur auf den „traditionellen" Gebieten (z. B. Planung, Abrechnung und Statistik) zu erheblichen Effektivitätssteigerungen führt, sondern auch mit dem Einsatz von qualitativ vollkommen neuen Elementen der elektronischen Informationsverarbeitung – den Mikroprozessoren – in der materiellen Produktion eine wesentliche Beschleunigung des wissenschaftlich-technischen Fortschritts erreicht werden muß.

Für den Ingenieur, der nicht unmittelbar für die Datenverarbeitung verantwortlich ist, bedeutet das:

– Er muß über ein entsprechendes Grundwissen zur elektronischen Datenverarbeitung verfügen,
– beurteilen zu können, wie ihn die EDV bei seinen unmittelbaren Arbeitsaufgaben unterstützen kann,
– zu erkennen, wie der Einsatz elektronischer Informationsverarbeitung bei den Ergebnissen seiner Tätigkeit zur Effektivitätssteigerung führen kann.

Die Hauptentwicklungsrichtungen des Einsatzes der elektronischen Informationsverarbeitung in der DDR wurden in den Dokumenten unserer Partei und Regierung deutlich herausgearbeitet. Siehe [1] und [2].
Dabei ergeben sich auch für die weitere Entwicklung vier Schwerpunkte des Einsatzes:
– Intensivierung der sozialistischen Produktion ,
– Erhöhung der Wirksamkeit von Leitung und Planung,
– Erhöhung der Effektivität der schöpferischen Arbeit,
– Erhöhung der Effektivität des Einsatzes der erarbeiteten Produkte.

Im Bericht des Politbüros an die 11. Tagung des Zentralkomitees der SED stellte Genosse Erich Honecker u. a. nochmals fest: *„. . .Beschlossen wurde die langfristige Konzeption zur beschleunigten Entwicklung und Anwendung der Mikroelektronik in der Volkswirtschaft der DDR; Maßnahmen zur Entwicklung, Produktion und zum Einsatz von Industrierobotern und von numerischen und nichtnumerischen Steuerungen von Maschinen; die Konzeption zur Entwicklung und weiteren Anwendung der elektronischen Datenverarbeitung sowie Vorschläge zur Rationalisierung der Verwaltungsarbeit in den Ministerien, Kombinaten und Betrieben der Industrie, des Bauwesens und in anderen volkswirtschaftlichen Bereichen. Nun sind daraus die konkreten Aufgaben für die Kombinate abzuleiten. . . Es stellt wachsende Ansprüche an die Qualität der materiell-technischen Basis, die Position der DDR als moderner sozialistischer Industriestaat zu behaupten. Dem tragen wir Rechnung, indem wir jene Produktionen entwickeln, die den wissenschaftlich-technischen Erfordernissen von Gegenwart und Zukunft entsprechen. Ein charakteristisches Beispiel dafür sind die Mikroelektronik und die sich daran anschließenden Zweige ihrer gerätetechnischen Anwendung . .“*

2. Aufbau und Arbeitsweise von EDVA

In einer EDVA sind die vielfältigsten Erkenntnisse vereinigt, die die Menschheit in der Praxis ihrer „Informationsverarbeitung“ gemacht hat. Eine EDVA ist heute ein ***Ensemble von wissenschaftlich-technischen Spitzenleistungen*** und jede neue Erkenntnis aus Wissenschaft und Technik wird heute auf ihre Verwertbarkeit für die EDV überprüft.

Trotz der verschiedensten neuen technischen Entwicklungen hat sich das Grundprinzip einer EDVA nach den ersten entscheidenden Entwicklungen kaum verändert. Obwohl die EDVA in einer verhältnismäßig kurzen Zeit einen gewaltigen Fortschritt in ihrem Einsatz gemacht hat, geht ihre Arbeitsweise immer noch auf drei wesentliche Grundprinzipien zurück:

- Die vom Menschen benutzten Informationen werden vorwiegend durch Zeichenfolgen ausgedrückt und diese Zeichen können auf verschiedenste Art durch physikalische Zustände dargestellt werden.
- Die elektronische Verknüpfung von physikalischen Zuständen läßt sich so gestalten, daß in ihrem Ergebnis neue physikalische Zustände entstehen, denen wieder neue sinnvolle Zeichenfolgen zugeordnet werden können.
- Die Anlagenelemente, die diese Verknüpfungen durchführen, können ebenfalls durch physikalische Zustände gesteuert werden. Dabei kann diese Steuerung sehr variabel sein und vor allen Dingen im voraus geplant werden.

Daraus ergibt sich für jede EDVA die grundsätzliche Aufteilung in **Ein-** und **Ausgabegeräte,** die **Verarbeitungseinheit** und **Speichereinheiten** für die Informationen einschließlich der Steuerinformationen.

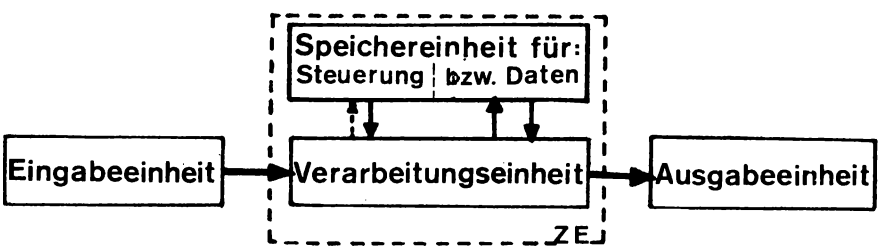


Bild 2.1: Prinzipielle Bestandteile einer EDVA

Die **Eingabegeräte** haben dabei die Aufgabe, die Zeichenfolgen (Eingabeinformationen) in der anlagenspezifischen Form als physikalische Zustände in der Speichereinheit bereitzustellen.

In der **Verarbeitungseinheit** werden die bereitgestellten physikalischen Zustände zu neuen physikalischen Zuständen verknüpft. Die Verknüpfung wird dabei vom Steuerprogramm (Programminformationen) bestimmt. D. h., die physikalischen Zustände im Programm steuern die Verknüpfung und bestimmen damit die Art der neuen physikalischen Zustände.

Die **Ausgabegeräte** haben dann die Aufgabe, die neuentstandenen, physikalischen Zustände wieder als Zeichenfolgen (Ausgabeinformation) zur Auswertung bereitzustellen.

Betrachtet man den Aufbau einer EDVA unter dem Gesichtspunkt des Informationsflusses, dann ergibt sich daraus der grundsätzliche Ablauf:

Die **Eingabeinformationen** werden in der Verarbeitungseinheit unter Steuerung durch die **Programminformationen** zu neuen **Ausgabeinformationen** verarbeitet.

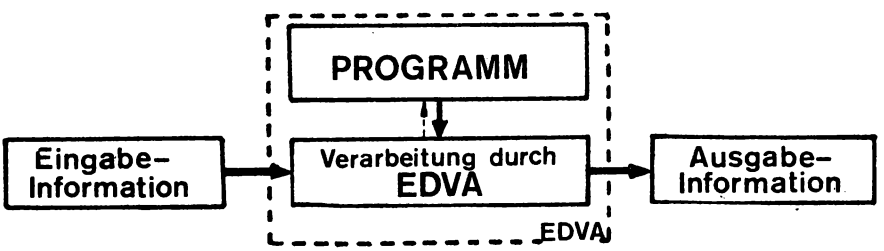


Bild 2.2: Informationsfluß in der EDVA

Beim konkreten Beispiel könnte diese allgemeine Beschreibung dann folgendermaßen aussehen:

Die Zahlen zur Beschreibung eines ökonomischen Prozesses (Eingabedaten) werden auf einem Datenträger (z. B. Lochkarte) als physikalische Zustände (Lochungen) dargestellt. Dabei entspricht jeder Ziffer (Zeichen) eine bestimmte Lochung im Datenträger. Bei der Eingabe durch den Lochkartenleser (Eingabegerät) werden diese Lochungen nochmals in andere physikalische Zustände (Magnetisierungen von Ferritspeicherkernen) umgewandelt.

In der Verarbeitungseinheit werden diese – die Zahlen darstellenden physikalischen Zustände (interne Zahlendarstellung) – dann in Abhängigkeit von den Programminformationen (z. B. Multiplikationsbefehl) verknüpft. Dabei entstehen neue physikalische Zustände, die genau der internen Zahlendarstellung des Produktes entsprechen. Durch den Drucker (Ausgabegerät) werden die neuen physikalischen Zustände wieder in Ziffern (Zeichen) umgewandelt und als neue Zahlen zur Beschreibung des ökonomischen Prozesses (Ausgabedaten) zur weiteren Auswertung in einer Druckliste ausgegeben.

Die Betrachtung dieser Zusammenhänge macht vier wesentliche Tatsachen deutlich:

- Die Verarbeitung der Informationen als physikalische Zustände kann durch die moderne Elektronik mit einer nahezu unvorstellbaren Geschwindigkeit erfolgen.
  Die Verarbeitungseinheit der EDVA ES 1055 des ESER II kann beispielsweise fast eine halbe Million Operationen in jeder Sekunde durchführen. Die sowjetische EDVA ES 1065 wird sogar ca. fünf Millionen Operationen in einer Sekunde ausführen.
- Die physikalischen Zustände können die verschiedensten Zeichen darstellen. Damit kann eine EDVA nicht nur Zahlen (numerische Zeichen) verarbeiten, sondern auch Zeichenfolgen aus Buchstaben (Textverarbeitung) oder Punktkoordinaten (grafische Verarbeitung). Das Auswechseln der Steuerprogramme führt dabei zu einer großen Universalität der elektronischen Datenverarbeitungsanlagen.
- Die erstaunlichen „geistigen" Leistungen der modernen EDVA werden nicht von den Geräten erbracht, sondern durch die Menschen, die diese Anlagen konstruieren, herstellen, programmieren und anwenden.
- Für die Anwendung der EDV ist nicht so sehr die Kenntnis der physikalisch-technischen Realisierung einer EDVA, sondern das Verständnis für die Möglichkeiten einer EDVA und ihre augenblicklichen Grenzen entscheidend.

Die Speichereinheiten für das Steuerprogramm und die Ein- bzw. Ausgabeinformationen können in einem **Hauptspeicher** zusammengefaßt werden, weil sich die technischen Realisierungen nicht unterscheiden. Damit ergibt sich der prinzipielle Aufbau des Gerätesystems, der **Hardware**, einer EDVA aus den Eingabegeräten, der Zentraleinheit mit dem Hauptspeicher (HS) und der zentralen Verarbeitungseinheit (ZVE), sowie den Ausgabegeräten.

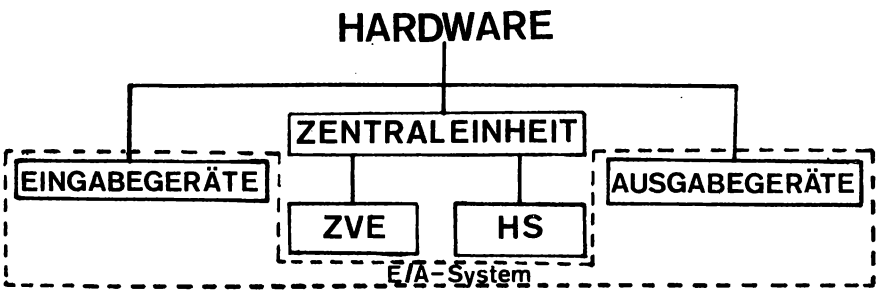


Bild 2.3: Prinzipielle Bestandteile der Hardware

Die Ein- und Ausgabegeräte der modernen EDVA werden zum physischen Ein/Ausgabesystem zusammengefaßt.
Zu einer modernen EDVA gehören aber neben dem Gerätesystem noch ausgefeilte Systeme von Programmen, die **Software,** mit deren Hilfe diese leistungsfähigen Anlagen erst effektiv eingesetzt werden können. Die Software kann dabei in zwei Bestandteile aufgegliedert werden: (s. Bild 2.4)

Die maschinenorientierten Systemunterlagen (MOS) unterstützen die unmittelbare Arbeit mit der Anlage. Ihr Kernstück ist das Betriebssystem.
Die problemorientierten Systemunterlagen (POS) sind vorgefertigte Lösungen von typischen Problemstellungen, die mit der EDV bearbeitet werden können. Sie können mit geringem Aufwand dem konkreten Problem angepaßt werden.

2.1. Informationsdarstellung

Der Informationsdarstellung in allen modernen EDVA liegt die Unterscheidung zwischen nur zwei physikalischen Zuständen zu Grunde. Dem entspricht in der Mathematik das Dualsystem.
Daraus ergibt sich, daß der kleinstmögliche Teil zur Informationsdarstellung ein Element ist, das genau zwei verschiedene Zustände annehmen kann. Dieses Element der Informationsdarstellung wird als **Binärstelle** oder **Bit** bezeichnet.

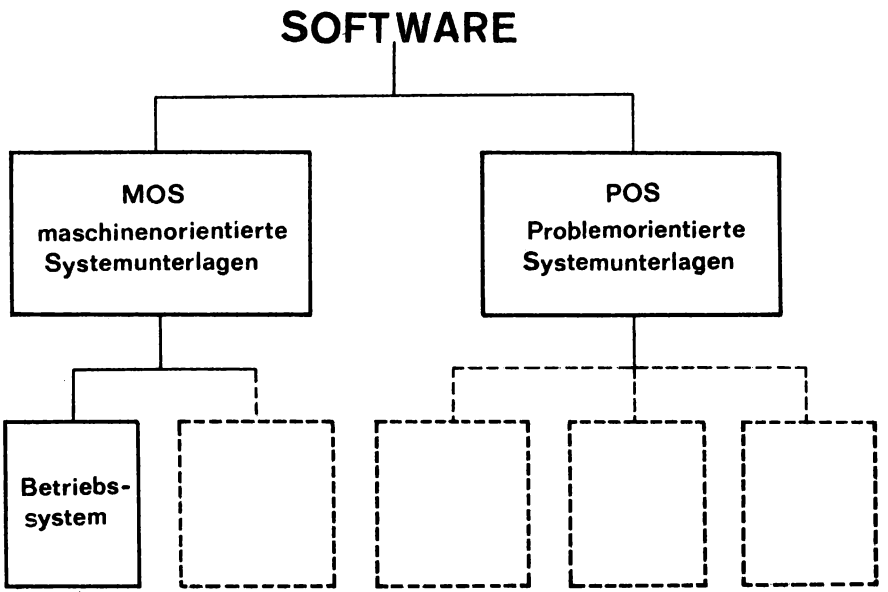


Bild 2.4: Bestandteile der Software

Ein* Bit *ist ein Element zur Informationsdarstellung, das genau zwei verschiedene Zustände annehmen kann.
Die beiden Zustände eines Bit werden vorwiegend durch die Zeichen 0 und 1 gekennzeichnet.

Zur Darstellung von Informationen im menschlichen Kommunikationsprozeß werden Ziffern. Buchstaben und andere Zeichen benutzt. Dabei müssen immer mehr als zwei Zeichen unterschieden werden, so daß jedes Zeichen durch mehrere Bit dargestellt werden muß.

2.1.1. Zeichendarstellung

Ein **Zeichen** *ist ein Element eines Systems sinnlich-wahrnehmbarer Gegenstände, die Träger der Informationen in der gesellschaftlichen Praxis sind. Mit ihrer Hilfe können Informationen zwischen allen Menschen ausgetauscht werden, denen ihre Bedeutung (ihr Sinn) bewußt ist.*
In der EDV werden vorwiegend Buchstaben, Ziffern und Sonderzeichen benutzt.

Zur Darstellung eines Zeichens in einer EDVA hat sich allgemein eine Einheit von 8 Bit durchgesetzt, die als ein **Byte** bezeichnet wird.

Ein **Byte** *ist eine Einheit von 8 Bit, die zur Zeichendarstellung in modernen EDVA benutzt wird.*

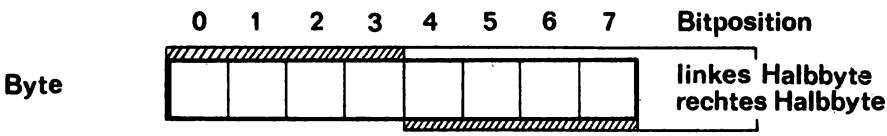


Bild 2.5: Byteaufbau

Aus der Kombinatorik geht hervor, daß mit diesen 8 Bit-, die ja jeweils die Zustände 0 bzw. 1 annehmen können, genau $2^8 = 256$ verschiedene Kombinationen dargestellt werden können. D. h., mit einem Byte können 256 verschiedene Zeichen dargestellt werden. Diese 256 Kombinationen ergeben sich, wenn man systematisch die Zustände der 8 Bit variiert.

00000000 ; 00000001 ; 00000010 ; 00000011 ; 00000100 ; 00000101 usw.

Für den Gebrauch in der EDV ist diese Darstellung sehr unübersichtlich, deshalb werden immer jeweils 4 Bit (ein Halbbyte) zusammengefaßt und durch ein besonderes Zeichen dargestellt. Dafür müssen wieder $2^4 = 16$ verschiedene Zeichen benutzt werden. Man spricht deshalb von **hexadezimaler Darstellung** des Byte.

Die hexadezimale (auch sedezimale) Darstellung des Byteinhaltes ist eine spezielle Darstellung des Zustandes von jeweils 4 Bit durch ein Zeichen. Es werden dabei die Zeichen 0,1, . . . ,9,A,B,C,D,E und F benutzt.

BITMUSTER	Hexadezimalzeichen	BITMUSTER	Hexadezimalzeichen
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Bild 2.6: Bitmuster und Hexadezimalzeichen

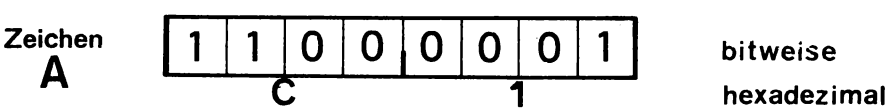


Bild 2.7: Darstellung eines Byteinhaltes

Die Zuordnung der Ziffern, Buchstaben und Sonderzeichen zu den 256 verschiedenen Bitmustern eines Byte muß natürlich für eine EDVA umkehrbareindeutig festgelegt werden. Diese Zuordnung erfolgt durch den **Maschinencode,** der durch eine ***Codetabelle*** dargestellt wird (Bild 2.8.).

> ***Ein Maschinencode ist eine Vorschrift, die jedem Zeichen umkehrbar eindeutig eine bestimmte Bitkombination zuordnet. Durch die umkehrbare Eindeutigkeit der Zuordnung wird der Informationsgehalt — die Information — bei der Umwandlung in die interne Darstellung (Eingabe) bzw. bei der Umwandlung aus der internen Darstellung (Ausgabe) nicht verfälscht.***

EBCD-Code (ESER)	hexa-dezimal → / binär ←	rechtes Halbbyte (Ziffernteil) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
linkes Halbbyte (Zonenteil)	0 0000																
	1 0001																
	2 0010																
	3 0011																
	4 0100	blank											.	¤	(	+	
	5 0101												$	*	)		
	6 0110		/										,	%	_		
	7 0111												*	@	'	=	
	8 1000																
	9 1001																
	A 1010																
	B 1011																
	C 1100		A	B	C	D	E	F	G	H	I						
	D 1101		J	K	L	M	N	O	P	Q	R						
	E 1110			S	T	U	V	W	X	Y	Z						
	F 1111	0	1	2	3	4	5	6	7	8	9						

Bild 2.8: Maschinencode des ESER

Die bisher betrachteten Zusammenhänge machen deutlich, daß zwischen der Informationsdarstellung in der gesellschaftlichen Praxis und der Informationsdarstellung in der EDVA unterschieden werden muß.

Das **Zeichen** *ist das Element der Informationsdarstellung für die gesellschaftliche Praxis.*
Das **Bit** *ist das Element der Informationsdarstellung für eine elektronische Datenverarbeitungsanlage.*
Bei der Darstellung eines Zeichens in einer EDVA wird heute allgemein eine Einheit aus 8 Bit, ein Byte, benutzt. Die Zuordnung zwischen Ziffern, Buchstaben und Sonderzeichen der gesellschaftlichen Praxis zur anlageninternen Darstellung wird durch den Maschinencode festgelegt.

2.1.2. Zahlendarstellung

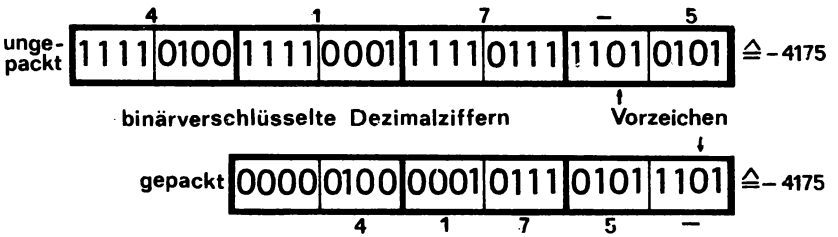
Die Entwicklung der modernen EDVA basiert auf den elektronischen Rechenanlagen, die anfangs erst nur Zahlen verarbeiten konnten. Auch heute stellen Zahlen noch einen wesentlichen Teil der in einer EDVA „verarbeiteten" Daten dar. Deshalb gibt es in einer jeden EDVA bestimmte Darstellungsarten, die besonders zur Zahlendarstellung geeignet sind. Sie unterscheiden sich teilweise sehr erheblich von der allgemeineren Zeichendarstellung, da ja nur die Ziffern 0 bis 9 sowie das Vorzeichen und eventuell das Dezimalkomma dargestellt werden müssen. Für die Darstellung der 10 verschiedenen Ziffern, mit denen Dezimalzahlen geschrieben werden, benötigt man 4 Bit. Viele EDVA benutzen zur Darstellung von Zahlen jeweils 4 Bit für die Ziffern einer Dezimalzahl. Diese Darstellungsform wird deshalb als **dezimaltetradische Darstellung** bezeichnet.

Bei der dezimaltetradischen Darstellung wird jede Ziffer einer Zahl durch 4 Bit oder ein Halbbyte dargestellt.

Der Übergang von der allgemeinen Zeichendarstellung des ESER zur dezimaltetradischen Zahlendarstellung kann sehr einfach durch das Weglassen der jeweils ersten 4 Bits eines Bytes (Zonenhalbbyte) erfolgen. Dabei wird die Darstellung der Zahl auf beinahe die Hälfte der Bit zusammengedrängt. Es wird deshalb von der ungepackten und gepackten Darstellung gesprochen.

Eine zweite, für die Verarbeitung in der EDVA noch besser geeignete Darstellung der Zahlen ist die Umwandlung der Dezimahlzahl in eine Zahlendarstellung des Dualsystems. Sie wird als **rein duale Darstellung** bezeichnet. Dabei wird die Zahl nur mit den Dualziffern 0 und 1 dargestellt. Die Stellenwerte sind von rechts nach links wachsende Potenzen von 2 und der Wert der Dualzahl ergibt sich aus der Summe aller Stellenwerte die mit einer 1 belegt sind.

Das Vorzeichen ist in diesem Beispiel bei der ungepackten Darstellung mit der Einerstelle kombiniert, dies ergäbe beim Ausdruck das Zeichen N entsprechend der Codetabelle.



Jede gepackte Zahl wird links bis zur Bytegrenze mit Nullen aufgefüllt.

Bild 2.9: Ungepackte und gepackte Darstellung des ESER

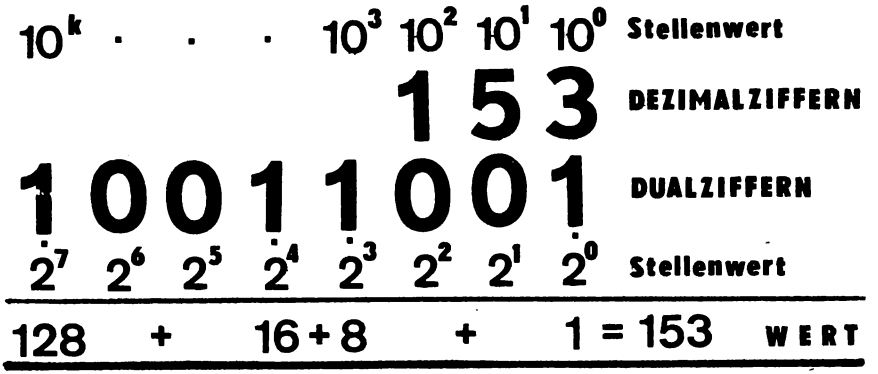


Bild 2.10: Dezimalzahl und Dualzahl

Den beiden Dualziffern 0 und 1 können jetzt unmittelbar die Zustände 0 bzw. 1 der einzelnen Bit zugeordnet werden.

Bei der rein dualen Darstellung wird die Dezimalzahl im Dualsystem dargestellt und jeder Dualziffer genau ein Bit zugeordnet.

Dem Vorzeichen der Zahl, das ja ebenfalls nur + bzw. – sein kann, wird der Zustand 0 bzw. 1 eines weiteren Bit zugeordnet.

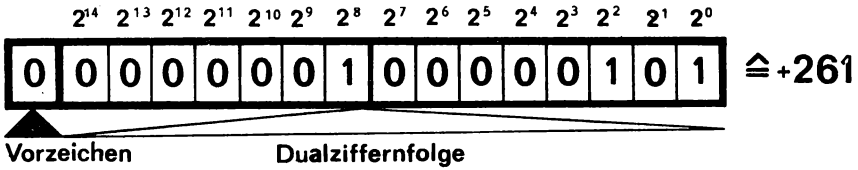


Bild 2.11: Zahlendarstellung des KRS 4201

Bei allen vorhergehenden Betrachtungen wurde bisher das Dezimalkomma einer Zahl nicht berücksichtigt. Für die Darstellung des Kommas einer Zahl gibt es in einer EDVA ebenfalls zwei grundsätzliche Möglichkeiten.
Die für die Konstruktion einer EDVA einfachste Möglichkeit besteht darin, daß die Anlage alle Zahlendarstellungen so behandelt, als stehe das Komma immer an der gleichen Stelle (Maschinenkomma). Bei dieser Darstellung muß der Programmierer die wirkliche Stellung des Komma bei der Programmierung berücksichtigen. Diese Darstellungsform wird als **Festkommadarstellung** bezeichnet. Sie ist für kleinere EDVA typisch.

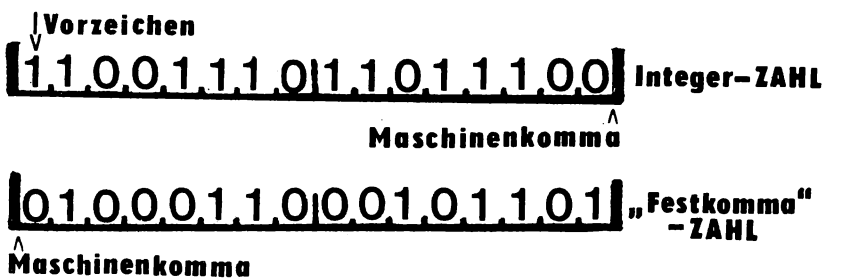


Bild 2.12: Festkommadarstellungen beim KRS 4201

Größere EDVA besitzen noch zusätzlich zur Festkommadarstellung eine technisch wesentlich aufwendigere Möglichkeit, mit Zahlen in einer **Gleitkommadarstellung** zu rechnen.

Bei der Gleitkommadarstellung wird jede Zahl so umgewandelt, daß sie ein Produkt zweier Faktoren ist. Der eine Faktor ist dadurch gekennzeichnet, daß das Komma immer an der gleichen Stelle steht. Dieser Faktor wird als Mantisse bezeichnet. Der zweite Faktor ist eine Potenz zu einer vorgegebenen Basis. Der Exponent dieser Darstellung charakterisiert dann die wirkliche Stellung des Kommas.

$$2728{,}7563 = 0{,}0027287563 \cdot 10^{6}$$
$$319{,}47 \quad = 0{,}0000031947 \cdot 10^{8}$$

Bild 2.13: Dezimalzahl in halblogarithmischer Darstellung

Die maschineninterne Darstellung der Zahl besteht dann aus den Ziffern der Mantisse und den Ziffern des Exponenten. Man nennt diese Darstellung auch eine halblogarithmische Darstellung, weil beispielsweise bei der Multiplikation von zwei Zahlen in dieser Darstellung die Mantissen multipliziert und die Exponenten addiert werden.

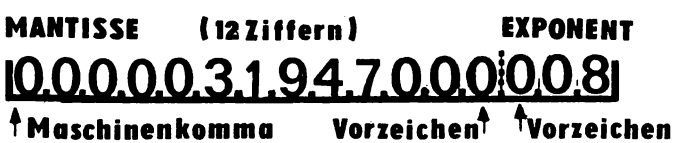


Bild 2.14: Mögliche interne Darstellung der Dezimalzahl

Der Anwender einer EDVA braucht dieses Problem bei größeren Anlagen im allgemeinen nicht zu berücksichtigen, wenn er nicht selbst maschinenorientiert programmiert.

2.2. Informationsstrukturen

Die Informationen der gesellschaftlichen Praxis, die mit Hilfe einer EDVA verarbeitet werden, bestehen normalerweise nicht nur aus Zahlen und einzelnen Zeichen. Sie besitzen eine Struktur.

Jeweils mehrere Ziffern und die Sonderzeichen + bzw. – sowie das Dezimalkomma stellen beispielsweise eine Zahl dar. Eine Material- oder Warenbezeichnung besteht im allgemeinen aus mehreren Buchstaben und/oder Ziffern.

Eine Einheit aus logisch zusammengehörenden Zeichen bezeichnet man als ein* Wort *oder* Datenelement *der logischen Datenstruktur.

Die Informationen der gesellschaftlichen Praxis **(Daten)** beschreiben Gegenstände, Personen, Ereignisse und Zusammenhänge. Dafür sind normalerweise mehrere Worte notwendig.
Eine Person wird durch den Namen, Vornamen, das Geburtsdatum, die Wohnanschrift, den Beruf usw. gekennzeichnet. Ein Bauelement kann durch die Materialart, die Abmessungen, die Form und andere Merkmale beschrieben werden.

***Eine Einheit aus logisch zusammengehörenden Datenelementen zur Beschreibung eines Gegenstandes, einer Person, eines Ereignisses oder eines Zusammenhanges bezeichnet man als einen logischen* Satz *oder* Datensatz.**

Die Lösung von Problemen mit Hilfe der EDV bezieht sich im allgemeinen nicht nur auf einen Gegenstand, eine Person, . . . usw. . Es werden meistens mehrere Datensätze der gleichen Verarbeitung unterworfen.

***Die Gesamtheit aller logisch zusammengehörenden Datensätze, die der gleichen Verarbeitung bei einer Problemlösung unterworfen werden, bezeichnet man als einen* Datenbestand.**

Den Elementen der logischen Informations- bzw. Datenstruktur entsprechen bei der Informationsdarstellung in der EDVA bestimmte Elemente einer physischen Daten- bzw. Informationsstruktur.
Ein Datenelement wird in einem **Feld** oder bei anderen Anlagen in einem **Maschinenwort** dargestellt.
Mehrere Felder zur Aufnahme eines Datensatzes werden als **Bereich** bezeichnet.
Dem Datenbestand der logischen Struktur entspricht eine **Datei** in der Programmierung.
In der Praxis der elektronischen Datenverarbeitung wird der Unterschied zwischen der logischen und physischen Struktur nicht immer berücksichtigt. Das kann zu erheblichen Mißverständnissen führen. Für den Anwender der EDV ist eigentlich nur die logische Datenstruktur von Interesse. Nur bei der Programmierung muß er eventuell bestimmte Begriffe der physischen Struktur benutzen.

2.3. Befehlsaufbau und -darstellung

Die bisher getroffenen Aussagen treffen sowohl auf die Eingabedaten als auch auf die Ausgabedaten zu. Für die Arbeitsweise einer EDVA ist noch eine andere Art der Informationsdarstellung von Bedeutung. Die Steuerinformationen für die Verarbeitungseinheit haben unter dem Gesichtspunkt ihrer Wirkung ebenfalls eine innere Struktur. Jeweils bestimmte Bit der Befehlsdarstellung haben für die Steuerung eine festgelegte Bedeutung. Dabei wird zwischen Operationsteil und Adreßteil eines Maschinenbefehls unterschieden.

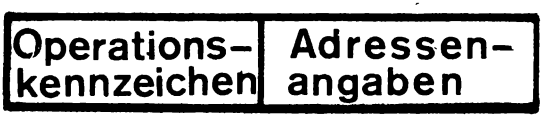


Bild 2.15: Prinzipieller Aufbau eines Maschinenbefehls

Die Bit des **Operationsteils** bestimmen die Art und Weise der Informationsverknüpfung, wenn dieser Befehl zur Steuerung benutzt wird. Die Bit des **Adreßteils** bestimmen den Ort der Informationsdarstellung im Hauptspeicher, mit der die Verknüpfung durchgeführt werden soll, wenn dieser Befehl zur Steuerung benutzt wird. D. h.,sie kennzeichnen nur den Ort für die Informationsdarstellung nicht den Inhalt der abgespeicherten Bit – die eigentliche Information.
Die EDVA werden entsprechend der Anzahl möglicher Adreßangaben in einem Maschinenbefehl in ***Einadreß-, Zweiadreß-*** oder ***Mehradreßanlagen*** unterschieden. Kleinere EDVA sind oft Einadreßanlagen.

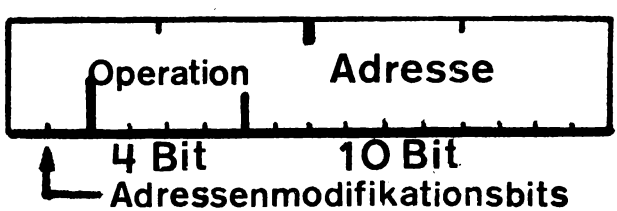


Bild 2.16: Befehlsaufbau KRS 4201

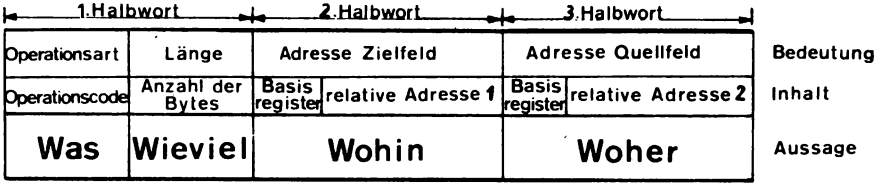


Bild 2.17: Befehlsaufbau ESER (SS-Format)

2.4. Befehls- bzw. Programmabarbeitung

Obwohl sich in der EDV immer mehr abzeichnet, daß mit Hilfe spezieller Programme die Art und Weise der Programmierung der menschlichen Ausdruckweise angenähert wird, ist es auch für den Anwender notwendig sich über die grundsätzliche Arbeitsweise einer EDVA Klarheit zu verschaffen. Diese Kenntnisse sind die Voraussetzung für eine EDV-gerechte Beschreibung eines Problems. Außerdem werden sie für einen Überblick über die Einsatzmöglichkeiten der EDV und den zu erwartenden Nutzeffekt benötigt.
Verfolgt man den Weg der Informationen in einer EDVA, dann ergibt sich folgender, allgemeiner Ablauf:

Die Daten werden in einer Form bereitgestellt, die es ermöglicht mit Hilfe der Eingabegeräte (Tastaturen, Lochbandleser, Lochkartenleser, Magnetbandgeräte, Magnetplattengeräte usw.) diese Daten in interner Darstellung im Hauptspeicher abzuspeichern. Dazu dienen die **Eingabebefehle.**

> ***Die Bereitstellung der Daten in interner Darstellung erfolgt auf Speicherplätzen, deren Ort (Adresse) bei der Festlegung der Eingabesteuerung (Eingabebefehle) bestimmt wird.***

Nachdem die Daten im Hauptspeicher bereitgestellt sind, können mittels Steuerung – durch **Verarbeitungsbefehle** – die Daten bearbeitet werden. Für die Verarbeitung der Daten gibt es in jeder EDVA eine bestimmte Auswahl von Befehlen.
Dabei sind aber in allen EDVA typische Befehlsgruppen vorhanden.

Durch die **Transportbefehle** werden die Daten so gesteuert, daß die interne Darstellung der Informationen von ihrem Platz im Hauptspeicher (Quelladresse) auf einen neuen Platz im Hauptspeicher (Zieladresse) übertragen wird.

Daten können durch Transportbefehle umgeordnet werden.

Durch die **arithmetischen Befehle** wird die Datenverarbeitung so gesteuert, daß aus der internen Darstellung von zwei Informationen (Operanden) eine interne Darstellung entsteht, die der Summe, der Differenz, dem Produkt usw. entspricht. Es entstehen dabei echte neue Datendarstellungen, die vorher nicht in der Anlage vorhanden waren.

Aus den Eingabedaten können durch arithmetische Befehle neue Daten berechnet werden.

Eine entscheidend andere Qualität einer **ZVE** gegenüber einem einfachen Taschenrechner – der ja auch neue Daten berechnen kann – ergibt sich aus der Möglichkeit die Steuerung der Verarbeitung in Abhängigkeit von den konkreten Daten vorher variabel zu planen (programmieren). Dazu dienen die **bedingten Sprungbefehle** oder **Verzweigungsbefehle.** Im Gegensatz zu den Transport- und Arithmetikbefehlen, die nur auf die Datendarstellung im Hauptspeicher einwirken, führen die bedingten Sprungbefehle zu einer Einwirkung auf den Ablauf der Steuerung. Jedes Progamm besteht aus einer Folge von Steuerinformationen (Befehle), die nacheinander im Hauptspeicher stehen. Im Normalfall wird nach der Abarbeitung eines Befehls automatisch der physisch folgende Befehl ausgeführt. D. h. er bewirkt die Steuerung des folgenden Verarbeitungsschrittes.
Bei den bedingten Sprungbefehlen gibt es dagegen zwei Möglichkeiten:

1. Die zum Steuerbefehl gehörende Bedingung wird überprüft und es ergibt sich, daß sie nicht erfüllt ist. Dann wird wie im Normalfall der physisch folgende Befehl aufgerufen.
   Die zu überprüfende Bedingung im Befehl könnte z. B. das Vorzeichen des Ergebnisses einer Addition sein. Dazu wird dann das Vorzeichenbit der Informationsdarstellung der Zahl herangezogen. Bezieht sich die Bedingung auf ein negatives Vorzeichen (Vorzeichenbit 1) und ist das Vorzeichenbit im Zustand 0, dann ist die Bedingung nicht erfüllt – der folgende Befehl wird wirksam.

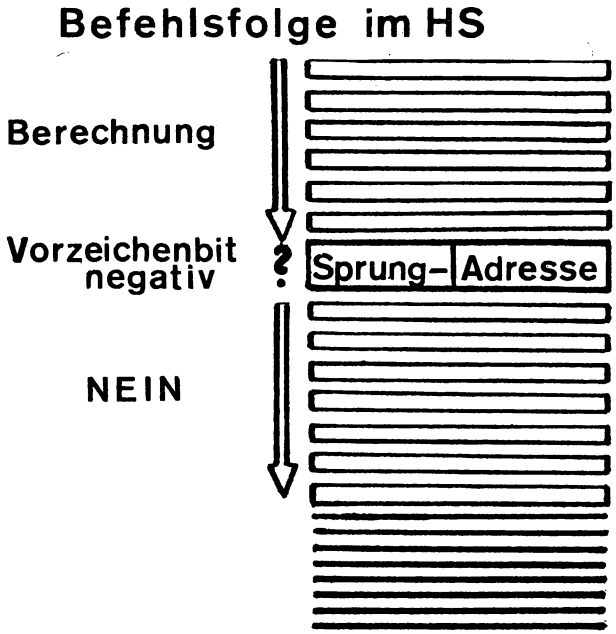


Bild 2.18: Befehlsfolge bei nichterfüllter Bedingung

2. Die überprüfte Bedingung ist erfüllt. Dann wird als nächster Befehl die Information wirksam, deren Ort im Hauptspeicher als Sprungadresse im Befehl angegeben ist.
   Das Ergebnis der Addition ist ein negativer Wert. D. h., das Vorzeichenbit der Zahlendarstellung nimmt den Zustand 1 an und stimmt damit mit der Bedingung im Befehl überein. Jetzt wird nicht der physisch folgende Befehl wirksam, sondern eine andere Befehlsfolge, die durch die Adresse im bedingten Sprungbefehl vom Programmierer festgelegt wurde.

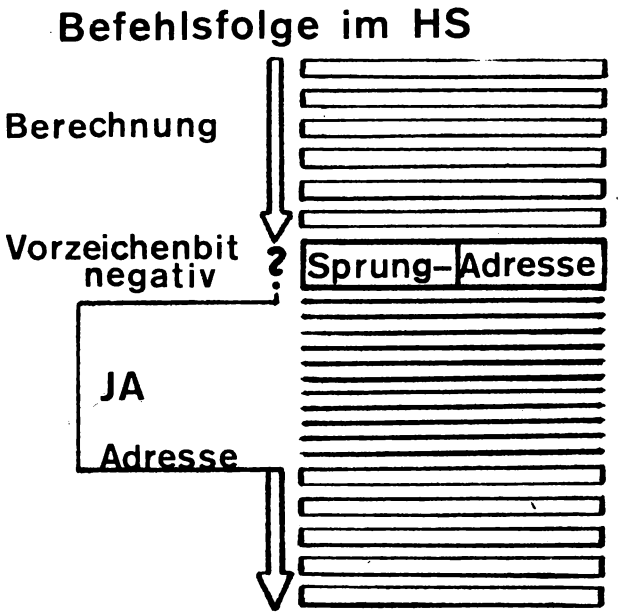


Bild 2.19: Befehlsfolge bei erfüllter Bedingung

– **Durch die bedingten Sprungbefehle kann die Verarbeitung in Abhängigkeit von bestimmten Eigenschaften der konkreten Daten (Bedingung) unterschiedlichen Befehlsfolgen unterworfen werden.**

Das Stellen solcher Bedingungsbit kann nicht nur durch arithmetische Operationen erfolgen. In den EDVA gibt es auch **Vergleichsbefehle**, die in Abhängigkeit von der Informationsdarstellung Bedingungsbit auf 0 oder 1 setzen. Diese Bit können dann wieder durch entsprechende bedingte Sprungbefehle ausgewertet werden.

Mit Hilfe von Vergleichsbefehlen, bedingten Sprungbefehlen und Transportbefehlen können Daten ausgewählt und sortiert werden.

Nachdem die Eingabedaten auf solche Art und Weise im Hauptspeicher umgeordnet, ausgewählt und sortiert sind bzw. neue Daten berechnet wurden, können

sie als Ausgabedaten über die Ausgabegeräte für die weitere Verarbeitung oder die sofortige Auswertung bereitgestellt werden. Dazu dienen die **Ausgabebefehle.**

Bei der Ausgabe können auch zwei Fälle unterschieden werden.

1. Wenn die Daten durch den Menschen weiter ausgewertet werden sollen, dann wird durch die Ausgabegeräte (Drucker, Schreibmaschine, Bildschirmgeräte, Mikrofilmgeräte, Zeichengeräte, Sprachausgabe usw.) die interne Informationsdarstellung wieder in die entsprechenden Zeichen umgewandelt.
2. Sollen die Ausgabedaten durch ein Programm der EDVA weiterverarbeitet werden, dann werden sie normalerweise in einer maschinenlesbaren, externen Informationsdarstellung mit Hilfe der Ausgabegeräte (Magnetbandgeräte, Magnetplattenspeicher, Lochbandstanzer usw.) auf einem entsprechenden Datenträger gespeichert.

Die Geräte und Datenträger einer EDVA, die sowohl die Eingabe als auch die Ausgabe von Informationen in maschinenlesbarer Form erlauben, bezeichnet man als **externe Speicher.** Sie können innerhalb eines Programms sowohl bei der Eingabe als auch bei der Ausgabe benutzt werden.

Aus dem Ablauf der Informationsverarbeitung ergibt sich auch die oft gebrauchte gerätetechnische Einteilung einer EDVA in Zentraleinheit und periphere Geräte. Zu den peripheren Geräten gehören die Ein- und Ausgabegeräte einschließlich der externen Speicher. Die Verbindung zwischen der Zentraleinheit und den peripheren Geräten wird bei modernen EDVA durch „Kanäle" hergestellt, die bestimmte Funktionen unabhängig von der Zentraleinheit ausführen können. Dadurch kann die Arbeit der Anlage bei den Ein- und Ausgaben, die die Zentraleinheit nicht benötigen, effektiver gestaltet werden.

Das Programm unterscheidet sich in der maschinenlesbaren Form und in der internen Darstellung nicht von den Daten. Beide Darstellungen sind in der EDVA letztlich nur bestimmte Bitmuster. Dadurch kann ein Programm genau wie die Ein- und Ausgabedaten durch das Ein/Ausgabesystem behandelt werden. Die EDVA kann sogar benutzt werden, um eine auf den Menschen zugeschnittene Beschreibung eines Programmes mit Hilfe von speziellen Übersetzungsprogrammen in die für die Anlage notwendige Form zu überführen. Solche Übersetzungsprogramme sind ein wesentlicher Bestandteil der Betriebssysteme von modernen EDVA.

3. Das einheitliche System elektronischer Rechentechnik

Der heutige Entwicklungsstand der EDV in der DDR wird durch die Hardware und Software des einheitlichen Systems elektronischer Rechentechnik (ESER) bestimmt. Mit dem ESER ist es den sozialistischen Staaten gelungen, ein abgestimmtes System von Hardware und Software zu planen, zu entwickeln und in kürzester Zeit bereitzustellen, das dem internationalen Standard der EDV entspricht. Im Gegensatz zu ähnlichen Entwicklungen großer Konzerne der kapitalistischen Länder, bei denen die notwendige Abstimmung der einzelnen Komponenten durch ökonomischen und politischen Druck im Konkurrenzkampf imperialistischer Länder erreicht wurde, haben die am ESER beteiligten sozialistischen Staaten bewiesen, welchen Vorteil die sozialistische ökonomische Integration auf dem Gebiete der EDV für alle beteiligten Länder bringt.

Anfang der 70er Jahre hatten die sozialistischen Länder durch den Einsatz der EDV umfangreiche Erkenntnisse auf nationaler Ebene gesammelt (in der DDR – R 300). Deshalb konnte im Rahmen der sozialistischen ökonomischen Integration, d. h. auf der Grundlage von freundschaftlicher Zusammenarbeit, gegenseitiger Hilfe und unter Berücksichtigung nationaler Interessen, durch die Sowjetunion, die DDR, die VR Polen, die CSSR, die VR Ungarn und die VR Bulgarien die Entwicklung des ESER in Angriff genommen werden. Das ESER war das erste Vorhaben der sozialistischen ökonomischen Integration im Jahre 1969.

An der Realisierung des ESER arbeiten etwa 20 000 Wissenschaftler, Ingenieure und Techniker und etwa 300 000 Facharbeiter in den beteiligten Ländern mit. Schon im Mai 1973 konnten in Moskau auf der ersten ESER-Ausstellung die Reihe der Zentraleinheiten des ESER I und über 100 periphere Geräte vorgestellt werden und damit die Leistungsfähigkeit der sozialistischen ökonomischen Integration eindrucksvoll demonstriert werden.

Im Juni 1979 wurden ebenfalls in Moskau schon die Hardware und die Software des ESER II und die ersten Ergebnisse des Systems der Kleinrechner (SKR) vorgestellt. Erstmals waren die Republik Kuba und die SR Rumänien mit eigenen, kompletten Anlagen vertreten.

3.1. Die neue Qualität des ESER

Gegenüber den vor den ESER-Anlagen eingesetzten EDVA wurde durch neue Bauelemente, neue Gerätetechnik und vor allem durch neue Systemunterlagen eine vollkommen neue Qualität der EDV-Anwendung möglich. Diese Entwicklung ist aber nicht stehengeblieben. Heute werden durch die Reihe II des ESER

bereits Maßstäbe gesetzt, die einen noch wesentlich effektiveren Einsatz der EDV ermöglichen.
Die neue Qualität ist sowohl auf eine neue Hardware als auch auf die neue Software zurückzuführen. Dabei muß beachtet werden, daß das eine nicht ohne das andere möglich ist und eine EDVA auch in ihrer Einheit aus Hardware und Software noch kein Urteil über EDV-Anwendung zuläßt. Erst wenn man die Möglichkeiten der Anlagen und ihren Einsatz betrachtet, kann eine objektive Beurteilung erfolgen.

3.1.1. Neue Elemente der Hardware

Während die ersten Anlagen des ESER noch durch den Einsatz von integrierten Schaltungen mit geringer Bauelementeanzahl gekennzeichnet waren, werden bei den Anlagen des ESER II immer mehr integrierte Schaltungen mit wesentlich größerer Bauelementeanzahl eingesetzt.
Bei der Realisierung der Hauptspeicher wurde von den teueren Ferritkernspeichern zu Halbleiterspeichern übergegangen. Durch solche und ähnliche Maßnahmen wurde nicht nur die Zuverlässigkeit und der Energieverbrauch der Anlagen positiv beeinflußt, sondern auch eine erhebliche Platz- und Materialeinsparung erreicht.

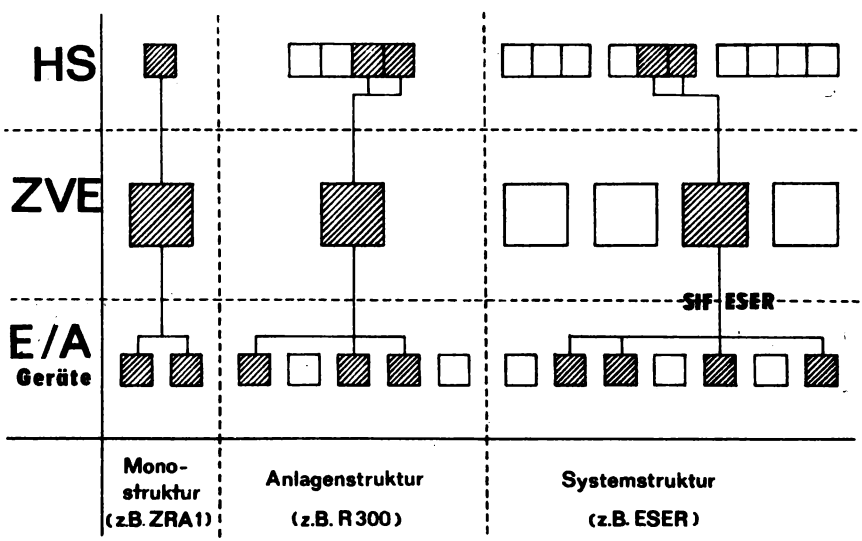


Bild 3.1: Strukturentwicklung der EDVA

Gegenüber den vorher eingesetzten EDVA (z. B. R 300) stehen im ESER außer verschiedenen peripheren Geräten auch noch verschiedene leistungsmäßig abgestufte Zentraleinheiten zur Verfügung, die über einen Standardanschluß (SIF ESER) mit den verschiedenen Geräten gekoppelt werden können. Dadurch ergibt sich für die Anlagen des ESER die Möglichkeit aus der Vielzahl der Zentraleinheiten und der peripheren Geräte eine optimale, dem Einsatz angepaßte Konfiguration auszuwählen.

Ein für die Hauptmodelle des ESER einheitlicher Befehlssatz und die einheitliche Informationsdarstellung sind weitere Voraussetzungen für diese Variabilität.
Für den Einsatz der Anlagen wesentliche neue periphere Geräte sind die Wechselplattenspeicher. Mit ihrer Hilfe ist es möglich genau wie mit Magnetbändern sehr große Datenmengen zu speichern. Im Gegensatz zum Magnetband können aber alle gespeicherten Daten unabhängig von der Stelle ihrer Speicherung in sehr kurzer Zeit zur Verarbeitung bereitgestellt werden. Sie werden deshalb auch als **Speicher** mit **wahlfreiem Zugriff** bezeichnet. Diese Speicher sind die Voraussetzung für den Einsatz der entsprechenden Betriebssysteme, für Datenbanken und für die im ESER II realisierte Technologie der **virtuellen Speicherung.**

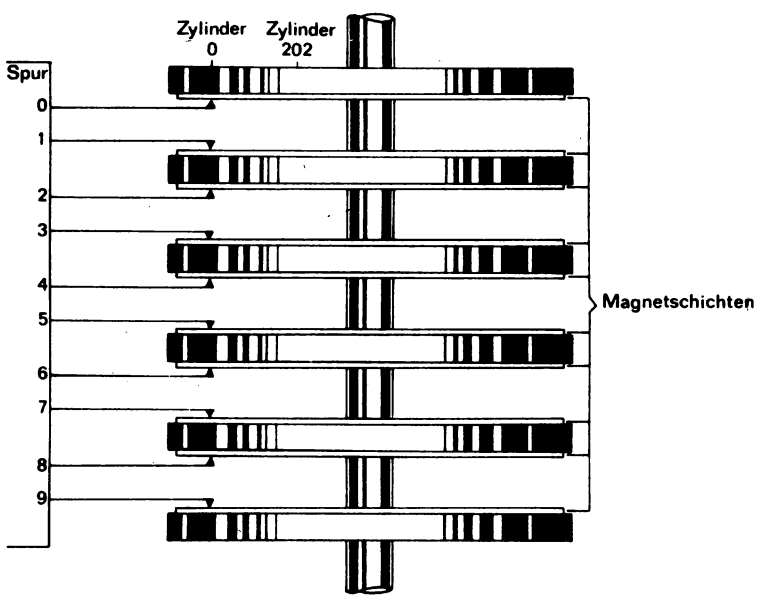


Bild 3.2: Magnetplattenspeicher (Prinzip)

Neue Ein- und Ausgabegeräte führen zu einem wesentlich effektiveren Einsatz der ESER-Anlagen. Zu diesen Geräten gehören u. a. **Bildschirmgeräte** und das in der DDR entwickelte **Mikrofilmausgabegerät.** Dadurch kann gegenüber den üblichen druckenden Ausgabegeräten eine wesentlich größere Ausgabegeschwindigkeit und eine erhebliche Papiereinsparung erreicht werden.

Für den EDV-Anwender außerhalb der großen Rechenzentren wird die neue Qualität vor allem durch den erweiterten Einsatz der ***Datenfernverarbeitung*** sichtbar werden.
Für spezielle Anwendungen können einige Anlagen des ESER II z. B. mit Zusatzeinrichtungen versehen werden, die auch hardwaremäßig eine bessere Programmierung ermöglichen. Für die EDVA ES 1055 aus der DDR wird es beispielsweise einen Matrixmodul (MAMO) geben, der Befehle für arithmetische Operationen mit Matrizen ermöglicht. Diese Operationen mußten bisher programmtechnisch realisiert werden.

3.1.2. Neue Elemente der Software

Für die ESER-Anlagen wurden eine Reihe von Betriebssystemen entwickelt, die auf die jeweilige Leistungsfähigkeit der einzelnen Konfigurationen abgestimmt sind.
Für die kleineren Modelle des ESER wird vorwiegend das **Betriebssystem DOS/ES** eingesetzt. Es ermöglicht beispielsweise in drei Bereichen des Hauptspeichers gleichzeitig verschiedene Programme zu verarbeiten (**Multiprogrammbetrieb**).

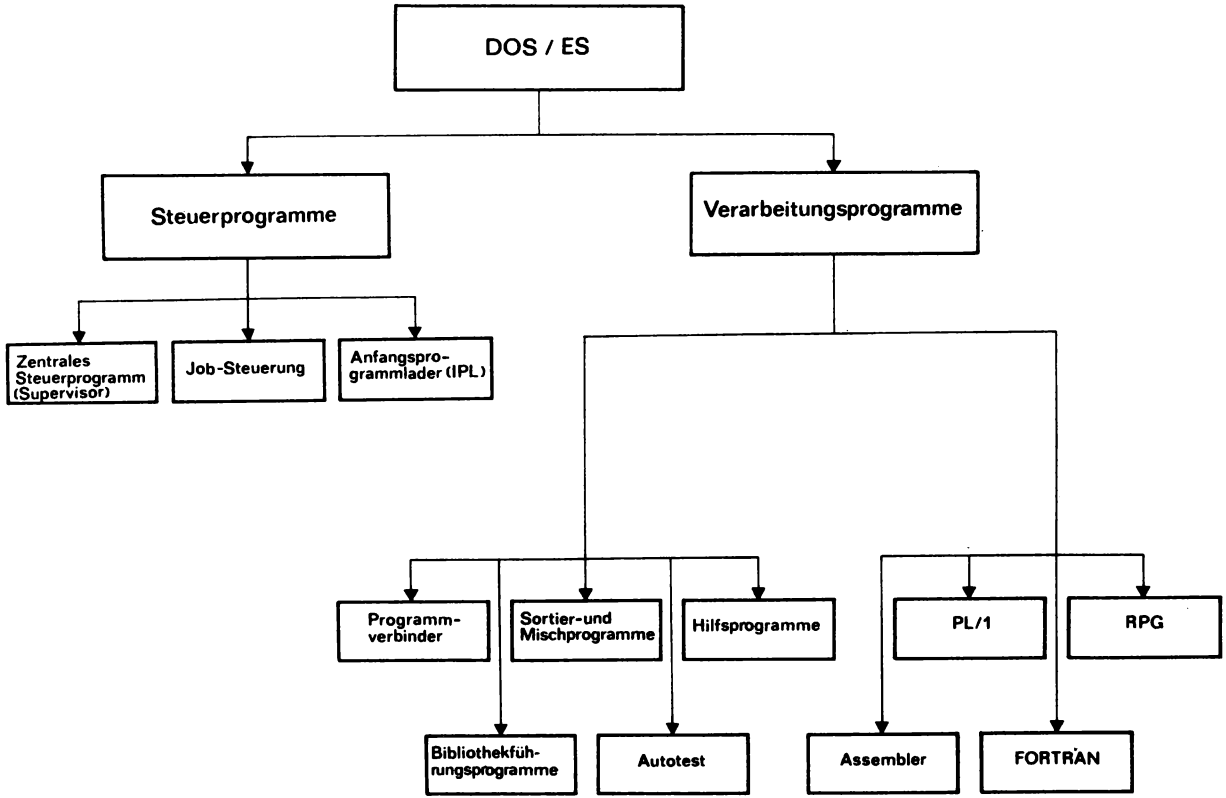


Bild 3.3: Das Betriebssystem DOS/ES

Für die größeren Modelle wird das **Betriebssystem OS/ES** benutzt, das in der Variante – Multiprogrammbetrieb mit einer festen Anzahl von Aufgaben (MFT) – beispielsweise bis zu 15 Programme im Hauptspeicher verarbeiten kann.

Für die Anlagen des ESER II ist diese Unterstützung durch das Betriebssystem OS/ES noch weiter ausgebaut. Die Variante – System mit einem virtuellen Adreßraum (SVS) – ermöglicht es z. B. beim ES 1055 mit einer scheinbaren Hauptspeicherkapazität vom 16 M Byte zu arbeiten, obwohl die Anlage nur mit einer maximalen Speicherkapazität von 2 M Byte ausgerüstet werden kann. Dazu wird durch das Betriebssystem die Arbeit so gesteuert, daß nur jeweils die notwendigen Teile der scheinbaren Hauptspeicherbelegung sich im realen Hauptspeicher befinden. Die nicht unmittelbar benötigten Teile werden auf die Speicher mit wahlfreiem Zugriff ausgelagert und automatisch in den realen Hauptspeicher transportiert, wenn sie für die Abarbeitung benötigt werden. Alle dafür notwendigen Steuerabläufe sind sowohl geräteseitig als auch im Betriebssystem berücksichtigt, so daß der Anwender der Anlage arbeiten kann, als hätte sie einen Hauptspeicher mit ca. 16 000 000 Byte Speicherkapazität.

Während bei den Anlagen des ESER I noch die Programme für das Betriebssystem DOS/ES verändert werden müssen, wenn sie im Betriebssystem OS/ES bearbeitet werden sollen, ist es bei der ES 1055 möglich, durch eine spezielle Einrichtung der Anlage diese Programme ohne Änderungen im Betriebssystem OS/ES abzuarbeiten.
Bei diesen Möglichkeiten erscheint es unwahrscheinlich, daß auch diese Anlagen nach den im Abschnitt 2 beschriebenen Prinzipien arbeiten sollen. Die im Abschnitt 2 gemachten Aussagen sind jedoch vollständig richtig. Diese Möglichkeiten sind nur ein Ausdruck dafür, daß einer EDVA wesentliche Teile der Arbeit selbst überlassen werden können. Genauer muß man allerdings sagen, daß wesentliche Teile der Steuerung einer EDVA schon bei der Konstruktion und bei der Entwicklung der entsprechenden Programme durch die Menschen realisiert wurden.

Wie groß der Unterschied zwischen EDVA und dem Menschen ist, kommt auch darin zum Ausdruck, daß zwar auf dem Gebiet der Anlagensteuerung dieser gewaltige Fortschritt möglich war. Aber wenn es um die Programmierung der Anlage für die Lösung von Problemen der gesellschaftlichen Informationsverarbeitung geht, dann zeigen sich erhebliche Schwierigkeiten. In den Forschungslabors sind schon einige EDVA in der Lage gesprochene Worte zu „verstehen". Heute ist jedoch noch kein real einsetzbares Übersetzungsprogramm in der Lage, aus solchen Sätzen wie „Sortiere 3, 4 und 5 der Größe nach" oder „Du sollst 3, 4 und 5 nach der Größe sortieren" ein Programm zu erstellen.

Für die Programmierung einer EDVA muß man deshalb immer noch eine **Programmiersprache** erlernen, die nicht mehr unbedingt auf eine spezielle EDVA (Assemblersprache) abgestimmt sein muß. Sie muß sich aber durch ihre Regeln zur „Satzbildung" den in dieser Hinsicht sehr bescheidenen Möglichkeiten einer EDVA anpassen.

Aus diesem Grund wird der größte Rationalisierungseffekt für den Anwender der EDV heute durch die **problemorientierten Systemunterlagen** erreicht. Dabei handelt es sich um von Spezialisten vorprogrammierte Problemlösungen bzw. Teillösungen für bestimmte Sachgebiete des gesellschaftlichen Reproduktionsprozesses, die mit verhältnismäßig geringem Aufwand an die konkreten Bedingungen angepaßt werden können. Diese Anpassung erfolgt dann mit Mitteln, die es dem Anwender weitgehend erlauben sich in einer seinem Fachgebiet entsprechenden Form auszudrücken. Diese Mittel können sowohl spezielle Sprachen als auch vorgegebene Formulare sein.

3.2. Übersichten zum ESER

Die zum ESER gehörenden Komponenten sind so zahlreich, das an dieser Stelle nur eine kleine Auswahl dargeboten werden kann. Außerdem bleibt gerade auf diesem Gebiet die Entwicklung so dynamisch, daß niemand die Garantie dafür übernehmen kann bei einzelnen Angaben wirklich noch auf dem neuesten Stand zu sein. Wesentlich für das Verständnis auf dem Gebiet der EDV für den Anwender ist jedoch nicht die Kenntnis einzelner Details – dafür hat sich die gesellschaftliche Arbeitsteilung schon sehr frühzeitig herausgebildet – sondern der Überblick und die Einsicht, daß trotz der manchmal sehr kompliziert erscheinenden Zusammenhänge die EDV in unserer Gesellschaft ein nicht mehr wegzudenkender Faktor zur Effektivitätssteigerung ist.

Die Tafeln 1 bis 3 im Anhang geben einen kleinen Einblick in die Vielfalt der für das ESER bereitgestellten Hardware und die wesentlichen Leistungsparameter.

Eine ähnliche Situation besteht bei der Software. Besonders die Entwicklung der problemorientierten Systemunterlagen ist so dynamisch, daß nur eine beispielhafte Aufzählung gegeben werden soll:

BASTEI – Bankspeicherung technischer Informationen
Dieses **s**achgebiets**o**rientierte **P**rogrammier**s**ystem (SOPS) wurde zur Arbeit mit Dateien, – insbesondere für Dateien zur Planung und Steuerung der Produktion in Betrieben mit Fertigung von Einzelteilen, Baugruppen und montierbaren Erzeugnissen – geschaffen.

MAWI – Betriebliche Materialwirtschaft
Dieses SOPS unterstützt die Anwendung der EDV bei den Problemen der mittelfristigen Materialplanung, der Bedarfsvorhersage sowie der Kontrolle von Bestellung und Lieferung

GRUMI – Grundmittel
GRUMI enthält Lösungen für die Grundmittel- und Instandhaltungsrechnung

PLUS – Planung und Steuerung der Produktion
Zu den vorgefertigten Problemlösungen gehören Auftragsbildung, -terminisierung und -auslösung, Bilanzierung und Fortschrittskontrolle.

KORAST – Kostenrechnung

Die SOPS beinhalten Programme, die bei der Bearbeitung der von ihnen abgedeckten Sachgebiete mit Hilfe der EDV eingesetzt werden können. Zur Anpassung der einzelnen Teillösungen an die konkreten Bedingungen besitzen sie eine **Problemvariabilität,** so daß der Anwender aus den angebotenen Teillösungen eine Auswahl treffen kann, um seine speziellen Probleme zu lösen. Er kann auch die Teillösungen durch eigene Programme ergänzen, um so eine optimale Lösung seines speziellen Problems zu erhalten.

Der oft noch unterschiedliche Aufbau der Daten in den einzelnen Betrieben und die unterschiedliche eingesetzte Gerätetechnik macht neben der Problemvariabilität auch noch eine **Dateivariabilität** notwendig. Diese Dateivariabilität ermöglicht es dem Anwender, die vorgegebenen Programme an die logische Struktur seiner Daten anzupassen. Das heißt, daß die Länge, die Stellung der einzelnen Worte im Satz und auch die Datenträger bei der Ein- und Ausgabe an seine speziellen Bedingungen angepaßt werden können.

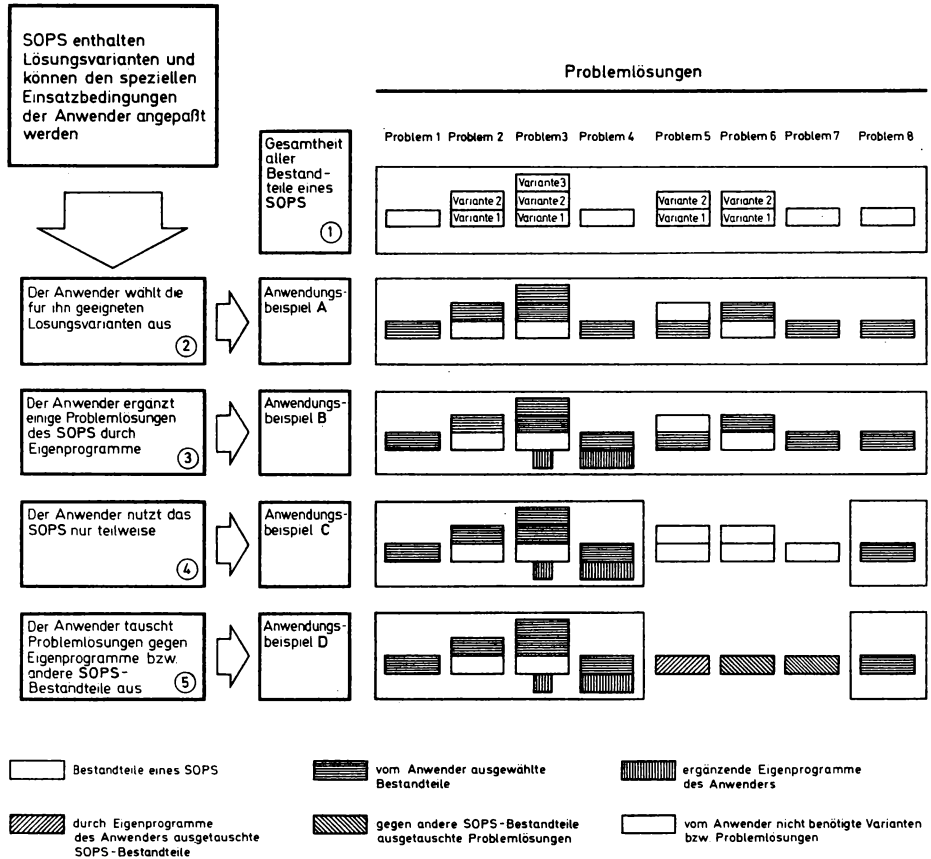


Bild 3.4: Problemvariabilität

Bild 3.5: Dateivariabilität

4. Die Mikrorechentechnik

In den letzten 10 bis 15 Jahren hat sich im Rahmen der wissenschaftlich-technischen Revolution auf den verschiedensten Gebieten eine quantitative Erhöhung von Erkenntnissen und Technologien ergeben, die sich auf die Beherrschung informationeller Prozesse bezieht. Besonders daran beteiligt sind solche Gebiete wie Kybernetik und Festkörperphysik sowie die Anwendung der elektronischen Datenverarbeitung und die Meß-, Steuerungs- und Regelungstechnik. Aber auch die Kosmosforschung, die Unterhaltungselektronik und zahlreiche andere Gebiete haben dazu beigetragen, daß die Beherrschung der informationellen Prozesse in eine neue Qualität umschlagen konnte.

Die stürmische Entwicklung der Herstellungstechnologien für mikroelektronische Schaltkreise, die es heute ermöglicht, einige Tausend Bauelemente auf einem winzigen Siliziumplättchen unterzubringen, schuf die technologische Voraussetzung für diese Revolution.

Die Erfahrungen beim Einsatz der elektronischen Datenverarbeitung, der Prozeßrechentechnik und der MSR-Technik waren die wissenschaftlich-informationstheoretischen Voraussetzungen.

Als dritte Komponente kommt die wesentliche Ausweitung des Anwendungsbereiches informationeller Prozesse hinzu. Sie ermöglicht erst im Zusammenhang mit den beiden anderen Komponenten eine ökonomisch vertretbare Anzahl bei der Herstellung der neuen Bauelemente.

Bis zum Jahre 1971 wurden zwar schon zahlreiche mikroelektronische Bauelemente mit standardisierter Aufgabenstellung zur Informationsverarbeitung bereitgestellt. Sie waren aber dadurch gekennzeichnet, daß sie auf eine entsprechende Eingabeinformation eine schon bei der Herstellung festgelegte Ausgabeinformation abgeben.

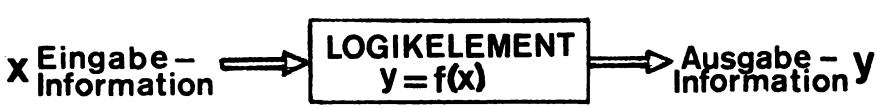


Bild 4.1: Informationsverarbeitung mit Logikelement

Die Entwicklung von Geräten auf der genannten Basis zur Steuerung informationeller Prozesse erfordert einen großen Aufwand. Die numerische Steuerung einer Fräsmaschine und eine EDVA wurden zwar weitgehend aus den gleichen Bauelementen aufgebaut, aber für beide Anwendungsfälle mußten sehr unterschiedliche Bauelementekombinationen zusammengestellt werden.
Die neuen Elemente der Informationsverarbeitung, **die Mikroprozessoren,** besitzen gegenüber den alten Bauelementen die Eigenschaft, auf eine Eingabeinformation mit einer großen Anzahl Ausgabeinformationen reagieren zu können. Ihre Reaktion wird erst beim speziellen Einsatz durch ein entsprechendes Programm festgelegt.

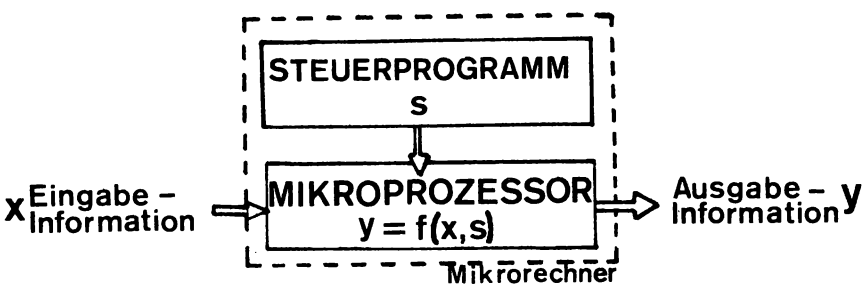


Bild 4.2: Informationsverarbeitung mit Mikroprozessor

Das bedeutet, daß unterschiedliche Einsatzfälle nicht mehr zu unterschiedlichen Kombinationen von Bauelementen sondern zu unterschiedlichen Programmen führen. Welche Vorteile sich dabei ergeben, kann vielleicht aus folgendem Beispiel ersichtlich werden:
In einem Steuerprogramm soll eine Zeitverzögerung realisiert werden. (z. B. Fotobelichtungsuhr, Waschmaschine, Steuerung chemischer Prozesse usw.)

Wenn die technische Realisierung durch ein elektronisches Zeitglied aus einem Widerstand und einem Kondensator realisiert wird, dann kann zwar in einem bestimmten Bereich durch einen veränderlichen Widerstand auch die Zeit verändert werden, aber eine größere Zeitveränderung würde andere Bauelemente notwendig machen.
Erfolgt die Zeitverzögerung im Rahmen des Steuerprogramms eines Mikroprozessors durch wiederholte Subtraktion einer 1 von einer vorgegebenen Zahl und wird

die Verzögerung beim Erreichen des Wertes 0 beendet, dann kann die Verzögerung durch ein einfaches Verändern der Ausgangszahl in beinahe unbegrenztem Maße variiert werden. Außerdem ist es möglich, mit dem gleichen Mikroprozessor auch noch vorher in Abhängigkeit von den konkreten Bedingungen des zu steuernden Prozesses die optimale Zeitverzögerung zu berechnen.
Der gleiche Mikroprozessor könnte aber auch mit einem anderen Programm die Steuerung der Bremsstrecken auf einem Rangierbahnhof der Deutschen Reichsbahn in Abhängigkeit vom Gewicht der Eisenbahnwagen übernehmen.

Heute sind auch im Weltmaßstab noch gar nicht alle Einsatzmöglichkeiten der Mikroprozessoren einschätzbar. In dem Maße, wie es uns in unserer Republik gelingen wird, Mikroprozessoren immer kostengünstiger herzustellen, wird sich ihr Einsatz über die Anlagen der chemischen Industrie und des Maschinenbaus oder die Datenverarbeitung hinaus immer mehr erweitern. Die Anwendung von Mikroprozessoren ist konkreter Ausdruck der Nutzung von Ergebnissen aus Wissenschaft und Technik. Im Zeitraum nach 1980 wird der Einsatz von Mikroprozessoren bis hin zu Erzeugnissen der Konsumgüterindustrie nicht nur eine Revolution in der Arbeit von Entwicklungsingenieuren mit sich bringen, sondern auch in vielen anderen Bereichen der geistig-schöpferischen Arbeit wird die Fähigkeit, dieses Hilfsmittel zu nutzen, über das Entwicklungstempo unserer Volkswirtschaft mitentscheiden.

Der Einsatz von Mikroprozessoren führt zur
- Verringerung der Gerätekosten
- Verkürzung der Entwicklungszeit von Gerätesystemen
- Verringerung des Raum-, Energie- und Materialbedarfes und zur
- Erhöhung der Zuverlässigkeit technischer Systeme.

Mit ihrer Hilfe können technische und auch geistige Prozesse optimal gestaltet werden. Sie können damit einen wesentlichen Beitrag zur Intensivierung unserer Volkswirtschaft und zur weiteren Verbesserung des gesellschaftlichen Lebens leisten.

4.1. Der Mikroprozessor

Unter einem **Mikroprozessor** (MP) versteht man in der Regel einen oder wenige Halbleiterschaltkreise, die die prinzipielle Funktion der ZVE einer EDVA realisieren. Man spricht deshalb auch von einer Mikroverarbeitungseinheit (MVE).
Durch Verbindung mit ebenfalls in Großintegrationstechnik (LSI-large scale integration) hergestellten Halbleiterspeichern und entsprechenden Anschlußschaltungen (Tore) für die Ein- und Ausgabeinformationen erhält man einen **Mikrorechner.**

Entsprechend den verschiedenen Einsatzmöglichkeiten der Mikrorechner wurden auch verschiedene Halbleiterspeicher realisiert.

ROM (read only memory) – **Nurlesespeicher**

Ein ROM wird nach den Angaben des Anwenders bei der Herstellung durch verschiedene Masken so gefertigt, daß die im Speicher vorhandenen physikalischen Zustände den Steuerinformationen des Programms entsprechen. Sie können nach der Herstellung des ROM zwar ausgewertet (gelesen) aber nicht verändert werden.
ROM können sehr kostengünstig hergestellt werden.

RAM (random acces memory) – **Schreib- und Lesespeicher mit wahlfreiem Zugriff**

In ein RAM können nach der Herstellung vom Anwender bzw. vom Mikroprozessor Informationen eingespeichert (geschrieben) und wieder gelesen werden. Er erfüllt also die Aufgabe des Hauptspeichers einer EDVA. RAM sind, bezogen auf die gleiche Bitanzahl, wesentlich teuerer als ROM und erfordern noch zusätzlichen Aufwand. Bei Stromausfall gehen die Informationen verloren.

Es wurden noch weitere Zwischenformen wie PROM, EPROM oder EAROM entwickelt. Diese Speicher sind nach der Herstellung vom Anwender programmierbare ROM, die aber auch wieder (z. B. durch ultraviolettes Licht) gelöscht und neu programmiert werden können. Ihre Herstellungskosten liegen zwischen denen der ROM und RAM. Bei der Anwendung eines Mikrorechners kann dann in Abhängigkeit vom Einsatzfall die ökonomisch günstigste Speicherform ausgewählt werden.

Für Steuerprogramme, die beim Einsatz nicht geändert werden, können die kostengünstigen ROM eingesetzt werden. Bei den modernen EDVA werden sie heute schon in der Steuerung eingesetzt. Für Steuerprogramme, die erst nach längerem Einsatz geändert werden, – z. B. bei verändertem technologischen Ablauf – sind programmierbare ROM einsetzbar. Zur Datenverarbeitung werden sie beispielsweise erfolgreich in Datenerfassungsgeräten verwendet. Für Daten oder variable Steuerinformationen werden dann zusätzlich nur noch die notwendigen, teueren RAM angeschlossen.
Damit ergibt sich für einen Mikrorechner ein prinzipieller Aufbau aus Mikroprozessor, ROM, RAM, E/A-Tore und BUS-System.

Das **BUS-System** besteht dabei aus den **Sammelleitungen für die Signale** der **Steuer-, Adreß-** und **Dateninformationen.**

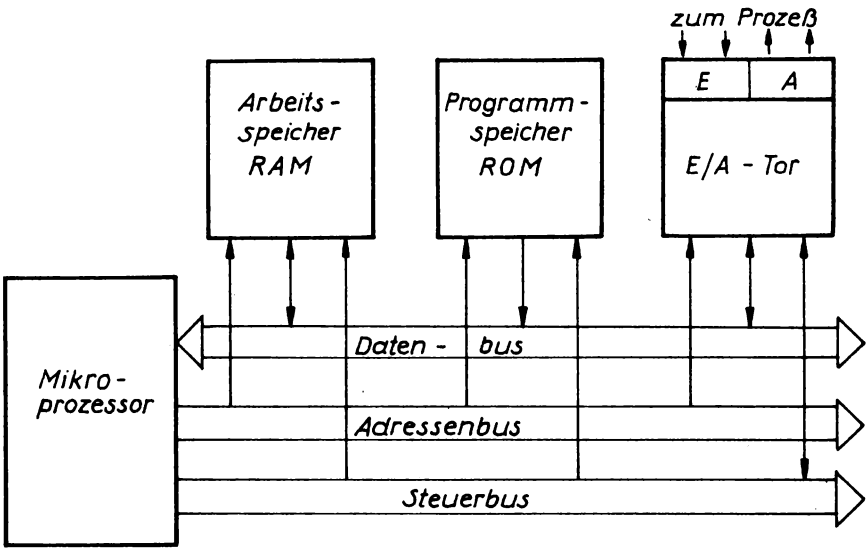


Bild 4.3: Blockschaltbild eines Mikrorechners

Die peripheren E/A-Geräte werden sich je nach Einsatzfall des Mikrorechners unterscheiden. Für die Steuerung von technischen Anlagen werden die Eingabegeräte vorwiegend Meßwertgeber und die Ausgabegeräte vorwiegend Stellglieder sein.

Beim Einsatz als Prozeßrechner werden die E/A-Geräte durch entsprechende Dateneinund Datenausgabegeräte ergänzt.
Ein Mikrorechner mit der „normalen" EDVA-Peripherie ergibt dann eine leistungsfähige Kleindatenverarbeitungsanlage oder ein „intelligentes" **Terminal.**

4.2. Einsatz und Wirtschaftlichkeit

Der Mikroprozessor und die Mikrorechner bieten die Möglichkeit, die elektronische Informationsverarbeitung überall dort einzusetzen, wo der Aufwand für eine EDVA oder einen herkömmlichen Prozeßrechner nicht vertretbar oder aus anderen Gründen nicht realisierbar wäre. Unmittelbar in den Anlagen kann beispielsweise durch die algorithmisierbare Verarbeitung von Informationen eine wesentliche Intensivierung (d. h. Zeiteinsparung, Materialverbrauchssenkung, Qualitätssteigerung, Verbesserung der Arbeitsbedingungen etc.) erreicht werden.
Der wesentliche Faktor für die Einsatzentscheidung wird dabei sicher in der ersten Zeit noch der Kostenfaktor für die notwendigen Baueinheiten sein. Doch schon heute ist abzusehen, daß mit wachsenden Erfahrungen in der Herstellungstechnologie die Herstellungskosten erheblich gesenkt werden können. Außerdem muß eingeschätzt werden, daß ein ständig wachsender Anteil von Geräten nur dann exportfähig bleiben wird, wenn ihre Anwendung und ihr effektiver Einsatz mit Hilfe von Mikroprozessoren optimiert wird.

Gegenüber einer EDVA oder einem herkömmlichen Prozeßrechner ergeben sich neben den geringeren Herstellungskosten für Mikroprozessoren und Mikrorechner noch weitere Vorteile:

- Wesentlich geringerer Platzbedarf
- Geringerer Energieverbrauch
- Keine speziellen Anforderungen an das Raumklima
- Optimierung des Aufwandes durch Aufgabe der Universalität und Anpassung an die konkrete Aufgabenstellung

Aus diesen Vorteilen ergibt sich eine wesentliche Aussage für den Einsatz von Mikrorechnern:

Der Mikrorechner wird unmittelbar am Ort der informationellen Prozesse und zugeschnitten auf die konkreten Aufgaben im informationellen Prozeß eingesetzt!

Die breite Skala der Einsatzmöglichkeiten wird dabei an den extrem unterschiedlichen Fällen einer Aufzugssteuerung und der Steuerung einer Blechwalzstraße deutlich.

Bei der Aufzugssteuerung müssen verhältnismäßig wenig Informationen (Anforderungen der Stockwerke, Bewegungsrichtung und Standort der Aufzüge) verarbeitet werden. Nach einem relativ einfachen Algorithmus sind die Fahrten von beispielsweise vier Personenaufzügen in einem Bürohochhaus so zu optimieren, daß in kürzester Zeit und energieoptimal die notwendigen Personentransporte

durchgeführt werden. Dabei paßt sich die Fahrweise automatisch dem Personenstrom zu den jeweiligen Tageszeiten an.

Für die optimale Steuerung einer Walzstraße sind wesentlich mehr Informationen (Stellung, Temperatur, Oberflächenbeschaffenheit und Geschwindigkeit der einzelnen Walzen sowie Materialeigenschaften des Walzgutes) nach einem komplizierten Algorithmus zu verarbeiten.

4.3. Auswirkungen in der Datenverarbeitung

Durch die Möglichkeit des Einsatzes von Mikrorechnern unmittelbar am Ort der Entstehung der Daten wird sich im Laufe der Zeit der Prozeß der Datenbereitstellung wesentlich verändern. Die Datenverarbeitung wird noch besser an den Menschen angepaßt werden. D. h., die Geräte für die Ein- bzw. Ausgabe der Daten werden immer mehr auf die menschliche Arbeitsweise abgestimmt und für ihren Einsatz werden immer weniger Spezialkenntnisse notwendig werden. In einem absehbaren Zeitraum wird die Benutzung von Anlagen der EDV ähnlich selbstverständlich werden wie heute schon der Einsatz von elektronischen Taschen- und Tischrechnern.

Die Entwicklung des Einsatzes von Mikroprozessoren in der Datenverarbeitung hat in der DDR erst begonnen und schreitet schnell vorwärts. Eines ist sicher: In den nächsten Jahren werden wir uns an den Einsatz der Mikrorechentechnik immer mehr gewöhnen, ohne daß auch nur ein Mensch in der DDR dadurch arbeitslos wird. Im Gegenteil, der Einsatz von Mikroprozessoren wird mithelfen, unsere Politik der sozialen Sicherheit und des Wohlstandes fortzuführen. Auch das ist ein Zeichen dafür, daß wissenschaftlich-technische Revolution und elektronische Datenverarbeitung durchaus nicht ideologiefrei sind, wie es einige westliche Ideologen auch heute noch behaupten.

Literaturverzeichnis

Abschnitt 1.

[1] Bericht des Zentralkomitees der Sozialistischen Einheitspartei Deutschlands an den IX. Parteitag der SED. Berichterstatter: E. Honecker. Berlin: Dietz Verlag 1976, S. 69 – S. 89

[2] Direktive des IX. Parteitages der SED zum Fünfjahrplan für die Entwicklung der Volkswirtschaft der DDR in den Jahren 1976 – 1980. Berlin: Dietz Verlag 1976, S. 24, 27, 55.

[3] Bericht und Diskussion zur 11. Tagung des Zentralkomitees der SED. Berlin: Dietz Verlag 1979.

[4] Rechentechnik Datenverarbeitung. Berlin: Verlag Die Wirtschaft 14 (1977) Heft 11.

[5] Rechentechnik Datenverarbeitung. Berlin: Verlag Die Wirtschaft 15 (1978) Beiheft 1.

Abschnitt 2.

[6] ***K.-H. Steuer:*** EDV-Handbuch für den Leiter. Berlin: Verlag Die Wirtschaft 1974.

[7] Autorenkollektiv: Stoffsammlung Datenverarbeitungstechnik. Berlin: Zentralstelle für Bildung der VVB MR 1979.

Abschnitt 3.

[8] ***J. C. Quiniou:*** Marxismus und Informatik. Berlin: Akademie-Verlag 1974.

[9] ***J. Hämmerlein:*** Stoffsammlung ESER. Berlin: Zentralstelle für Bildung der VVB MR 1975.

[10] Rechentechnik Datenverarbeitung. Berlin: Verlag Die Wirtschaft 16 (1979) Beiheft 2.

Abschnitt 4.

[11] ***Jugel, A:*** Mikroprozessorsysteme. Berlin: Verlag Technik 1978.

[12] Autorenkollektiv: Mikrorechner. Dresden: Zentralstelle für Aus- und Weiterbildung des Industriebereiches Elektrotechnik/Elektronik 1979.

Anlage

Tafel 1

	ESER I				
ESER-Chiffre-Nr.	ES2610	ES2620	ES2630	ES2640	ES2650
Verantwortliches Land	UVR	VRB/UdSSR	UdSSR/VRP	DDR	UdSSR
Hauptspeicherkapazität in K Byte	8 – 64	64 – 256	128 – 512	256 – 1024	256 – 1024
Operationsgeschwindigkeit ca. Op/s	5 000	9 000	60 000	380 000	500 000
E/A-Kanäle	1 spez. Kanal	2*/1	3*/1	6*/1	6*/1
Übertragungsgeschwindigkeit ca. in K Byte/s	40 – 200	20 – 300	40 – 800	50 – 1300	110 – 1250

* Selektorkanäle zum Anschluß schneller E/A-Geräte mit maximal 256 E/A-Geräten
Multiplexkanal für den Anschluß langsamer E/A-Geräte mit maximal 256 E/A-Geräten

	ESER II (vorläufige Parameter)				
ESER-Chiffre-Nr.	ES2615	ES2625	ES2645	ES2655	ES2665
Verantwortliches Land	UVR	CSSR	UdSSR/VRP	DDR	UdSSR
Hauptspeicherkapazität in K Byte	64 – 160	128 – 256	256 – 3072	512 – 2048	4096 – 16384
Operationsgeschwindigkeit ca. Op/s	24 000	50 000	750 000	470 000	5 000 000
E/A-Kanäle	1	1	5*/2	4*/2	4*/2
Übertragungsrate ca. in K Byte/s	19 – 29	24 – 29	50 – 1600	40 – 1500	110 – 1500

* Blockmultiplexkanäle zum Anschluß schneller E/A-Geräte mit maximal 256 E/A-Geräten
Bytemultiplexkanäle zum Anschluß langsamer E/A-Geräte mit maximal 256 E/A-Geräten

Neben den vorgestellten Zentraleinheiten gibt es noch eine ganze Anzahl von Zwischenstufen, so daß die Modellreihe des ESER einen wesentlich größeren Umfang hat. Einige Zentraleinheiten z. B. ES2620 sind inzwischen durch Weiterentwicklungen z. B. ES2622 ersetzt worden.

Anlage

Tafel 2

Magnetplattenspeicher	ES 5052	ES 5055	ES 5066	ES 5067
Herstellerland	VR B	DDR	UdSSR	VR B
Speicherkapazität/Plattenstapel in M Byte	7,25	7,25	100	200
mittlere Zugriffszeit in ms	ca. 60	ca. 75	30	30
Übertragungsrate in K Byte/s	156	156	806	806
Stapelanzahl	1	1	1	2

Magnetbandspeicher	ES 5016	ES 5017	ES 5002	ES 5004*
Herstellerland	DDR	UdSSR/DDR	UdSSR/DDR	CSSR
Bandgeschwindigkeit in m/s	1,5	2	3	2
Aufzeichnungsdichte in Bit/mm	32	8/32	32/63	32/63
Datenrate in K Byte/s	48	64	96/189	64/126

* Die Aufnahme des ES 5004 in den Bestand des ESER ist noch nicht abgeschlossen

Lochkartengeräte	ES 6012	ES 6016	ES 7010	ES 7014
Herstellerland	UdSSR	CSSR	UdSSR	CSSR
Lesegeschwindigkeit Karten/min	600	1000		
Stanzgeschwindigkeit Karten/min			100	160

Lochbandstation	ES 7902 (Herstellerländer DDR/CSSR)
Lochbandleser	Lesegeschwindigkeit bis 2000 Zeichen/s
Lochbandstanzer	Stanzgeschwindigkeit 110 Zeichen/s

Drucker	ES 7031	ES 7034	ES 7035	ES 7037	ES 7039
Herstellerland	DDR	CSSR	DDR	UdSSR	UdSSR
Druckgeschwindigkeit Zeilen/min	900	600 – 900	600	700 – 1000	1100
Zeichen/Zeile	156	132	132	132/150	80/132/156

Anlage

Tafel 3

Bildschirmgerät ES 7906
UdSSR

Zeichenanzahl	40/80/Zeile
Zeilenanzahl	16/12
Zeichensatz	94

Tischzeichengerät ES 7054
CSSR

max. Zeichengeschwindigkeit in mm/s	50
Elementarschrittlänge in mm	0,05
Papierformat in mm	1750 x 1370
darstellbare Zeichen	96

Mikrofilmausgabegerät ES 7602
DDR

Ausgabeleistung	ca. 5 Mikrofiches/min
oder	ca. 100 000 Zeichen/s
Zeichenvorrat	64 Zeichen (Grundausstattung)
Zeichendarstellung	7 x 10 Raster

Datenendplatz ES 8505 DDR mit Multiplexor ES 8404 (KRS 4201)

mögliche Baugruppen
- Register-Ausweis-Leser
- Meßwertabfrageeinheit
- Alphanumerische Tastatur
- Druckwerte
- Lochbandleser und -stanzer

FACHSCHUL–
FERNSTUDIUM

FSD Dipl.-Ing. Peter Buffleb

GRUNDLAGEN DER INFORMATIONSVERARBEITUNG FÜR INGENIEURE

2/1

Programmablaufplanung

Herausgeber:

Institut für Fachschulwesen der
Deutschen Demokratischen Republik
Karl-Marx-Stadt

03 0160 02 1

Dieser Lehrbrief wurde
verfaßt von:

FSD RR Dipl.-Ing. Peter B u f f l e b
Ingenieurschule für Verkehrstechnik
„Erwin Kramer" Dresden

lektoriert von:

FSD Dipl.-Gewl. Engelbert B i e n
Ingenieurschule für Bauwesen und
Ingenieurpädagogik
Magdeburg

bearbeitet von:

FSD Dipl.-Phys. Werner S e i f e r t
Wissenschaftlicher Mitarbeiter am
Institut für Fachschulwesen der DDR
Karl-Marx-Stadt

Redaktionsschluß: 30. 12. 1983

Druck und buchbinderische Verarbeitung:
Zentralstelle für Lehr- und Organisationsmittel des Ministeriums
für Hoch- und Fachschulwesen, Zwickau

1. Auflage 1984
3. unveränderter Nachdruck 1987
Ag 613/48/87/5 300
Vorzugsschutzgebühr: 3,00 M

Inhaltsverzeichnis

0. Vorwort

Der vorliegende Lehrbrief vermittelt die elementaren Grundlagen der grafischen Darstellung von Algorithmen, der Programmablaufplanung.
Nach einer Vorstellung der Darstellungselemente nach TGL 22451 werden die elementaren Strukturen „Sequenz", „Alternative" und „Zyklus" erläutert. Bei den letzteren werden 6 Grundtypen vorgestellt.

Mit diesem Grundlagenwissen sollte jeder Student in der Lage sein, einfache Programmieraufgaben zu lösen.
Über die angegebenen Übungen hinaus sollten im Unterricht fachspezifische Aufgaben zur Vertiefung behandelt werden.

1. Darstellung von Informationsflüssen

1.0. Allgemeines

Der DV-Prozeß ist ein komplizierter und stark arbeitsteiliger Prozeß der Erfassung, Aufbereitung, Verarbeitung, Speicherung, Übertragung und Auswertung sowie Verwertung von Daten, der sich außerhalb und innerhalb der EDVA vollzieht.
Die Analyse und Darstellung von Informationsflüssen sind deshalb eine wichtige Aufgabe der DV-Projektierung, weil die komplizierten Beziehungen in einfacher und übersichtlicher Form widergespiegelt werden müssen, um den Prozeß zu beherrschen. Weiterhin wird für die DV-Projektierung eine einfache und eindeutige Verständigung zwischen den am DV-Projektierungsprozeß beteiligten Werktätigen (Organisatoren, Programmierer, Bediener von EDVA, Mitarbeiter der verschiedenen Fachbereiche u. a.) gefordert.

Der Informationsflußdarstellung kommen damit folgende Funktionen zu:

1. Organisationsfunktion
   Die Informationsflüsse sind wissenschaftlich zu durchdringen, optimal zu gestalten und in den Projektunterlagen zu dokumentieren.
2. Kommunikationsfunktion
   Die Informationsflußdarstellungen dienen als Verständigungsmittel zwischen den an der DV-Projektierung und Realisierung der DV-Projekte beteiligten Werktätigen. Diese Funktion gewinnt durch die immer engere internationale Zusammenarbeit der sozialistischen Länder im Rahmen des ESER an Bedeutung.

Grundsätzlich gibt es zwei Darstellungsmöglichkeiten für Informationsflüsse:

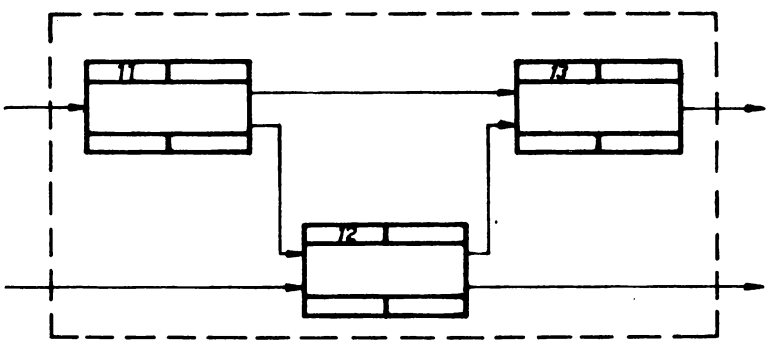
1. Textmethode
   Diese Methode ist sehr aufwendig und unübersichtlich.
   Außerdem weisen textliche Beschreibungen einen hohen Grad von Redundanz auf, und zum anderen entstehen Verständigungsprobleme bei einer internationalen Zusammenarbeit.
2. Grafische Darstellungen
   Da die Textmethode die erwähnten Nachteile aufweist, liegt deshalb der Schwerpunkt in der grafischen Darstellung. Die grafischen Darstellungen werden aber oft noch durch verbale Beschreibungen ergänzt.

   Für die grafische Darstellung von Informationsflüssen werden überwiegend
   – Blockschaltbilder (BSB)
   – Problemablaufpläne (PEP)
   – Datenflußpläne (DFP) und
   – Programmablaufpläne (PAP)
   verwendet.

Ihre zweckmäßige Einordnung in den Prozeß der Datenverarbeitungsprojektierung sind in /1/ ausführlich dargelegt. An dieser Stelle soll nur eine kurze Übersicht gegeben werden.

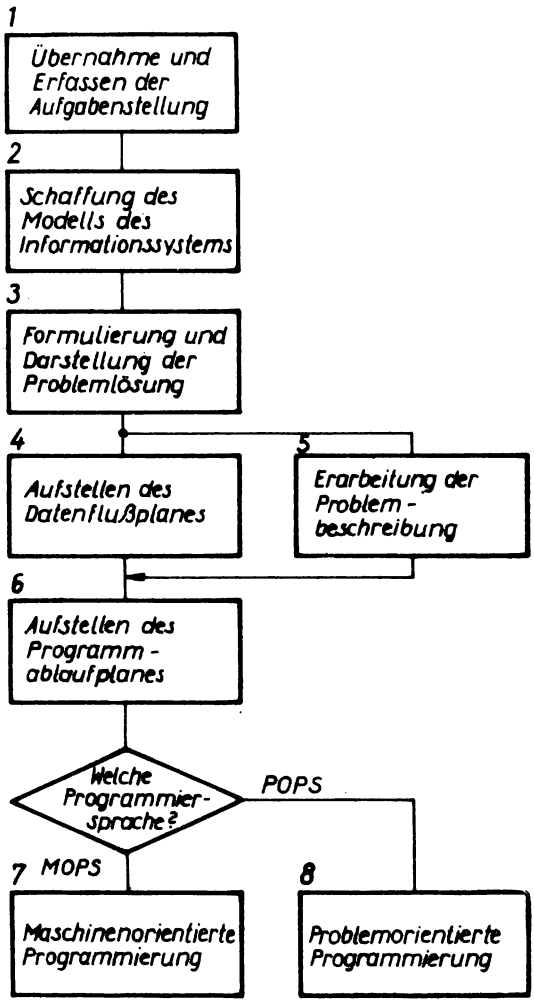
1.1. Blockschaltbilder

Im Rahmen der Vorbereitungsarbeiten, bei der Einarbeitung in die Problemstellung, bei der Untersuchung aller einflußnehmenden vorgegebenen Bedingungen, geht es um die Schaffung eines Modells des gesamten Informationssystems. Entsprechend den Leitungs- und Leistungsverknüpfungen einzelner Struktureinheiten geht es dabei um die Zerlegung des Gesamtsystems in einzelne Informationskomplexe.
Bei einer übersichtlichen grafischen Darstellung dieser Zusammenhänge bedient man sich der Blockschaltbilder. Ein solcher Block wird durch ein Rechteck dargestellt in dessen Unterteilung mindestens Nummer und Bezeichnung der Struktureinheit sowie eingehende und ausgehende Informationsträger eingetragen werden. Die Informationsverbindungen zwischen den Blöcken werden durch Pfeile gekennzeichnet (/1/, Kap. 13).



1.2. Problemablaufpläne

Im Ergebnis der unter 1.1. beschriebenen Modellierung des Informationssystems entsteht eine Problemlösung. Sie enthält alle Elementarinformationen und ihre Verknüpfungen in einer verbalen Beschreibung des zu lösenden Problems.
Um auch hier bei komplexen und komplizierten Strukturen eine bessere Übersicht zu erhalten, benutzt man Problemablaufpläne zur grafischen Darstellung. Sie charakterisieren den Prozeßablauf einschließlich der verwendeten mathematischen Modelle und Alternativen. Als Beispiel eines solchen Planes sei der in diesem Kapitel 1 beschriebene vereinfachte Ablauf einer Datenverarbeitungsprojektierung dargestellt.

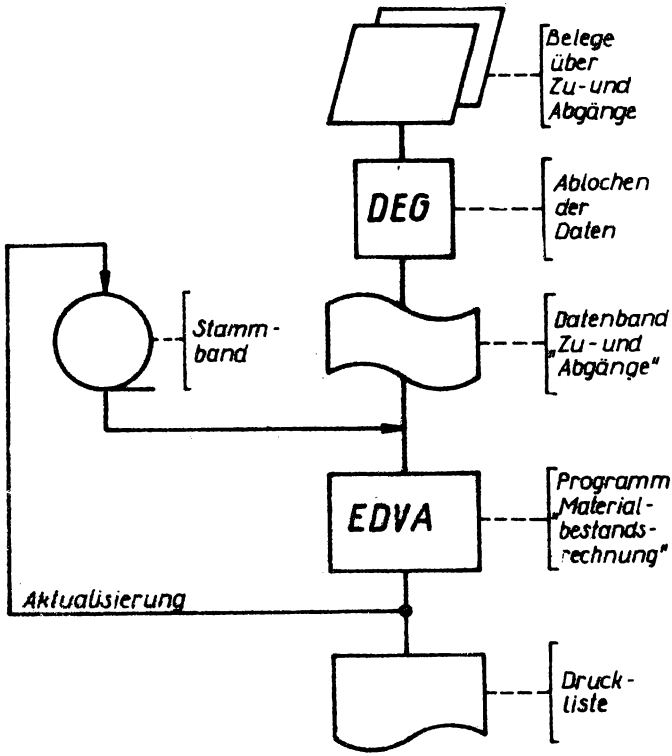


1.3. Datenflußpläne

Mit der Problemlösung ist die Grundlage gegeben, die datenverarbeitungsorganisatorische Lösung zu erarbeiten, Datenflußplan genannt. Mit ihm wird die Etappe der eigentlichen Projektierung der Aufgabe eingeleitet. Ein Datenflußplan kann demnach als Technologie zur Realisierung der im Modell des Informationssystems erarbeiteten Problemlösung mit Hilfe vorhandener oder vorgesehener Gerätetechnik angesehen werden (/1/, Kap. 6). Ein Datenflußplan beschreibt alle Organisationsabläufe für die Teilprozesse der Datenbereitstellung, Dateneingabe und Datenausgabe sowie der. Fehlererkennung und Fehlerbehandlung. In dieser Darstellung sind alle dazu notwendigen manuellen und gerätetechnischen Arbeitsvorgänge enthalten. Außerdem werden alle

materiellen Datenträger ausgewiesen.

Als Beispiel sei eine vereinfachte Materialbestandsrechnung dargestellt. Die verwendeten Symbole entsprechen der TGL 22451 /3/.



1.4. Programmablaufpläne

1.4.1. Definition

Für die Informationsverarbeitungsprozesse, die durch programmgesteuerte, digitale Rechenautomaten realisiert werden, erfolgt auf der Grundlage der bisher erarbeiteten Unterlagen die eigentliche Programmierung.

Dabei ist es möglich, in Abhängigkeit von der Kompliziertheit und Komplexität des Programms, von der Häufigkeit seiner Abarbeitung, vom Qualifizierungsgrad des Programmierers und von vorhandener hardware und software des Automaten, ein maschinenorientiertes oder ein problemorientiertes Programm zu erarbeiten. Beide stehen nach erfolgter Übersetzung durch die Anlage selbst auf maschinenlesbarem Datenträger zur Verarbeitung bereit.

Die Programmablaufplanung nimmt im Rahmen der Erarbeitung eines EDV-Programmes eine bedeutende Rolle ein. Bei der Vorbereitung der Programmierung kommt es darauf an, das zur Lösung des Problems gefundene mathematische Modell algorithmisch aufzubereiten, d. h. in lösbare Teilschritte zu zerlegen und grafisch darzustellen.

Die Programmablaufplanung als schöpferische Phase der Programmerarbeitung nimmt deshalb auch in der Ausbildung eine zentrale Stellung ein. In diesem Lehrbrief sollen deshalb die Grundfertigkeiten des Aufstellens von Programmablaufplänen vermittelt werden.

Während in einem Datenflußplan der Ablauf der Verarbeitung von Informationen in einer EDVA nur durch ein Kästchen angedeutet wird, stellt ein Programmablaufplan gerade diesen Ablauf als Grundlage für die Programmierung ausführlich dar.
„Der Programmablaufplan ist die Darstellung des Ablaufes in einem informationsverarbeitenden System in Abhängigkeit von den jeweils vorhandenen Daten" /3/.

1.4.2. Arten von PAP

Je nach Verwendungszweck und Kompliziertheit der Aufgabe unterscheidet man zwei Arten von PAP.

1. Verfahrensorientierte PAP
   Sie sind die unmittelbare Grundlage für die Programmierung in einer maschinenorientierten Programmiersprache und werden deshalb auch maschinenorientierte PAP genannt. Sie berücksichtigen die Besonderheiten der konkreten Verfahrensweise auf einem bestimmten Typ der EDVA.
   Jeder Schritt eines verfahrensorientierten PAP entspricht einem einzelnen Befehl oder einer Befehlsgruppe des auszuarbeitenden Programms. Nach dieser ins Detail gehenden oder groben Darstellungsweise spricht man auch von „feinen" oder „groben" PAP. Dabei richtet sich der Feinheitsgrad nach dem Inhalt des Problems, der Qualifikation des Programmierers, der verwendeten Programmiersprache und den Eigenheiten der verwendeten EDVA.
2. Problemorientierte PAP
   Sie dienen der Darstellung eines Algorithmus in seiner abstraktlogischen Struktur, ohne daß auf maschinentechnische Einzelheiten eingegangen wird. Sie eignen sich deshalb zur Vorbereitung der Programmierung in einer problemorientierten Programmiersprache.

In diesem Lehrbrief kann nicht eindeutig die eine oder die andere Art bevorzugt werden, da sowohl eine maschinenorientierte als auch eine problemorientierte Programmierung anschließen kann.

Da sich dieser Lehrbrief jedoch an Anfänger der Programmierung wendet, wird versucht, den Feinheitsgrad dem allgemeinen Verständnis anzupassen.

1.4.3. Qualitätsforderungen an PAP

An einen PAP werden allgemein folgende Qualitätsforderungen gestellt:

1. Das mathematische Modell des zu lösenden Problems ist vollständig und fehlerfrei darzustellen. Dabei sind Umrechnungen, die sich aus technischen oder physikalischen Maßeinheiten ergeben, zu berücksichtigen.
2. Alle Eingangs- und Ausgangsgrößen sowie der Anfangs- und Endzustand des Programms müssen eindeutig definiert sein.
3. Alle für das Verständnis des PAP erforderlichen Hinweise und Erläuterungen müssen vorhanden und die Möglichkeit von Änderungen gegeben sein.
4. Der PAP muß mit geringem Zeichenaufwand übersichtlich zu gestalten sein.

1.4.4. Darstellung von PAP

Ausgehend von den genannten Qualitätsforderungen entstanden in der DDR zwei Darstellungsarten, die in der TGL 22 451 ihren Niederschlag fanden /3/.

1. Kästchenmethode
   Hierbei werden kästchenartige Symbole für die Darstellung der verschiedenen Operationsarten verwendet. Die so gezeichneten PAP sind anschaulich und übersichtlich, weil die Art der Operation und ihre Aufeinanderfolge schon am Symbol erkannt werden. Eine Spezifizierung der Operationsarten erfolgt durch eine zusätzliche Beschriftung. Nachteile dieser Methode sind ein verhältnismäßig hoher Zeichenaufwand, der jedoch durch Verwendung einer Schablone weitgehend eingeschränkt werden kann, sowie ein hoher Platzbedarf.
2. Programmlinienmethode
   Dominierendes Element dieser Methode ist eine durchgehende Programmlinie, auf der die einzelnen Operationen nur durch kurze waagerechte Striche gekennzeichnet werden. Die Art der Operation ist nur durch die nebenstehende Beschriftung zu erkennen. Mit dieser Methode wird zwar Zeichenarbeit eingespart, jedoch sind Übersichtlichkeit und Anschaulichkeit geringer.

Während nach der TGL vom Oktober 1967 beide Methoden gleichberechtigt nebeneinander bestanden, gibt die im RGW-Bereich abgestimmte Neufassung vom Juni 1975 der Kästchenmethode den Vorrang.

Aus diesem Grunde und um die Vorteile der besseren Anschaulichkeit für die Lehre auszunutzen, wird empfohlen, vorerst überwiegend die Kästchenmethode zu verwenden. Nach einer gewissen Einarbeitungszeit, insbesondere bei der Erarbeitung größerer PAP, die sich über mehrere Seiten erstrecken würden, sollte man für Entwurfszwecke durchaus die Programmlinienmethode verwenden.

In diesem Lehrbrief werden deshalb beide Methoden nebeneinander angeboten.

Im folgenden wird, sofern dies zeichnerisch möglich ist und beide Darstellungsformen angegeben werden, links die Kästchenmethode und rechts daneben die Programmlinienmethod dargestellt.

1.4.5. PAP-Strukturen

Alle praktisch vorkommenden Programmablaufpläne lassen sich auf drei Grundstrukturen zurückführen:

- Lineare PAP (Sequenz)
- Verzweigte PAP (Alternative)
- Zyklische PAP

Nachfolgend sollen diese ausführlich behandelt werden.

2. Lineare PAP

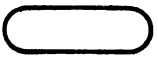
2.1. Die wichtigsten Grundsymbole

2.1.1. Grenzstellen

Unter einer Grenzstelle versteht man den Anfang, das Ende oder eine Unterbrechung des Programmablaufs. Die Art der Grenzstelle wird in das Symbol mit Großbuchstaben eingetragen.

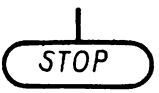
Die Symbole gelten für beide Darstellungsarten.

– Symbol Grenzstelle:

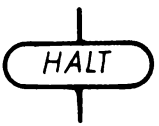


– Symbol PAP-Anfang:

– Symbol PAP-Ende:



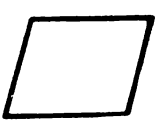
– Symbol PAP-Unterbrechung:



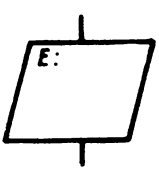
2.1.2. Transputanweisungen

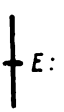
Unter dem Begriff Transputanweisungen faßt man alle Operationen zusammen, die den Datentransport zwischen der Zentraleinheit und den peripheren Geräten einer EDVA organisieren.

Allgemeines Symbol:

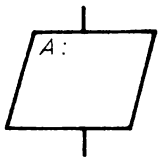


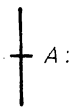
Je nach Transportrichtung, entweder in die Zentraleinheit (Hauptspeicher) hinein oder aus der Zentraleinheit (Hauptspeicher) heraus, unterscheidet man Eingabe- und Ausgabeoperationen. Dies wird durch eine Buchstabenangabe E oder A gekennzeichnet.





– Symbol Datenausgabe:

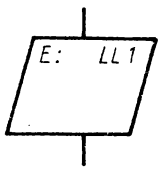




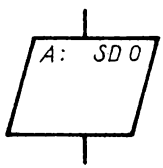
Ist es notwendig, das bei der Datenübertragung verwendete periphere Gerät mit anzugeben, dann erfolgt dies durch Angabe der Abkürzung.

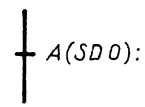
Beispiele:

– Eingabe über
Lochbandleser 1:



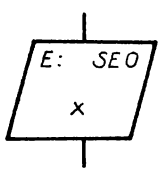
– Ausgabe über
Seriendrucker 0:

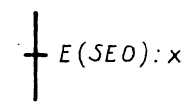




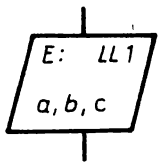
Weitere Angaben zu den zu übertragenden Informationen werden wie folgt eingetragen:

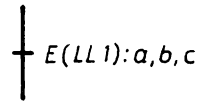
– Eingabe der Variablen x über
die Bedienschreibmaschine 0:



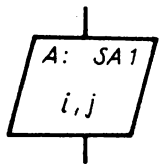


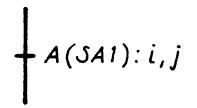
– Eingabe der Variablen a, b und c über Lochbandleser 1:



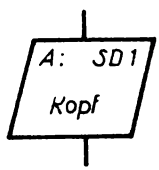


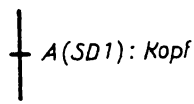
– Druck der Variablen i und j auf der Bedienschreibmaschine 1:





– Druck des Textes eines Tabellenkopfes auf dem Seriendrucker 1:

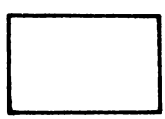




2.1.3. Ergibt-Anweisungen

Ergibt-Anweisungen beschreiben den Informationstransport innerhalb der Zentraleinheit einschließlich der erforderlichen Verknüpfungen in der zentralen Verarbeitungseinheit.

Allgemeines Symbol:



Die eigentliche Ergibt-Anweisung wird in das Rechteck-Symbol bzw. rechts neben die Programmlinie eingetragen. Dabei sollte einheitlich die mathematische Schreibweise des Modells oder die Schreibweise der verwendeten Programmiersprache benutzt werden. Ersteres ist zu bevorzugen, da die später verwendete Programmiersprache zu diesem Zeitpunkt noch nicht bekannt ist.

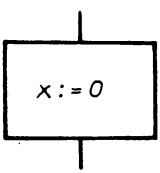
Eine Ergibt-Anweisung besteht aus drei Teilen:

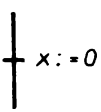
Linke Seite	Ergibt-Zeichen	rechte Seite

Die linke Seite besteht immer aus einer **Variablen**. Jeder Variablen wird bei der Ausführung des Programms ein Speicherplatz zugeteilt, auf dem der Wert der Variablen abgespeichert wird. Deshalb kann diese Variable zugleich als symbolische Speicherplatzadresse betrachtet werden.
Das Ergibt-Zeichen besteht aus Doppelpunkt und Gleichheitszeichen := und wird „ergibt sich aus" gesprochen.
Die rechte Seite besteht aus einem **arithmetischen Ausdruck.** Im einfachsten Fall ist das eine Konstante, es kann aber auch eine Variable sein. Bei den „echten" arithmetischen Ausdrücken finden dann immer Verknüpfungen statt.

Beispiele für Ergibt-Anweisungen

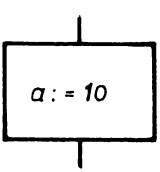
(1)

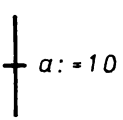




x ergibt sich aus null, d. h., der Variablen x wird der Wert 0 zugewiesen, oder die Speicherzelle mit der symbolischen Adresse x wird mit dem Wert 0 belegt.

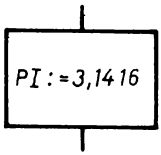
(2)

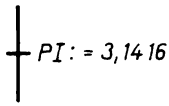




a ergibt sich aus 10, d. h., der Variablen a wird der Wert 10 zugewiesen, oder die Speicherzelle mit der symbolischen Adresse a wird mit dem Wert 10 belegt.

(3)

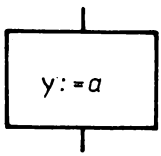


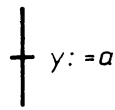


PI ergibt sich aus 3,1416, d. h., der Variablen PI wird der Wert 3,1416 zugewiesen, oder die Speicherzelle mit der symbolischen Adresse PI wird mit dem Wert 3,1416 belegt.

Solche Wertzuweisungen werden häufig zu Beginn eines PAP notwendig. Man nennt sie deshalb auch Anfangswertzuweisungen. Es ist besonders zu beachten, daß die Konstanten für die Anfangswertzuweisungen **nicht** über periphere Geräte eingelesen werden.

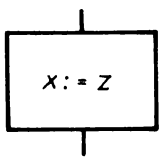
(4)

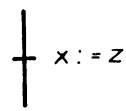




y ergibt sich aus a, d. h., der Variablen y wird der Wert a zugewiesen, oder die Speicherzelle mit der symbolischen Adresse y wird mit dem Wert der Speicherzelle mit der symbolischen Adresse a belegt.

(5)

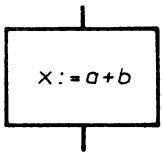


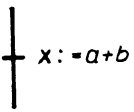


x ergibt sich aus z, d. h., der Variablen x wird der Wert z zugewiesen, oder die Speicherzelle mit der symbolischen Adresse x wird mit dem Wert der Speicherzelle mit der symbolischen Adresse z belegt.

In diesen Fällen spricht man von einer Umspeicherung. Der Zahlenwert der rechts stehenden Speicherzelle wird auf die linksstehende Speicherzelle übertragen oder umgespeichert. Der Inhalt der links stehenden Speicherzelle wird dadurch verändert, der Wert der rechts stehenden Speicherzelle bleibt erhalten. Generell ist dabei zu beachten, daß die rechts stehenden Variablen vorher berechnet, zugewiesen oder eingelesen worden sind.

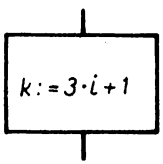
(6)

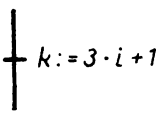




x ergibt sich aus a plus b, d. h., der Variablen x wird der Wert der Summe aus a und b zugewiesen. Die Werte der Speicherzellen mit den symbolischen Adressen a und b werden addiert. Die Speicherzelle mit der symbolischen Adresse x wird mit dem Wert der Summe belegt.

(7)





k ergibt sich aus 3 mal i plus 1, d. h., der Variablen k wird der Wert des rechts stehenden arithmetischen Ausdrucks zugewiesen. Der Wert der Speicherzelle mit der symbolischen Adresse i wird mit 3 multipliziert und zum Produkt wird eins addiert. Die Speicherzelle mit der symbolischen Adresse k wird mit dem Ergebnis der Berechnung belegt.

Diese beiden Beispiele zeigen den Normalfall einer Ergibt-Anweisung. Aus den bekannten Größen des rechts stehenden arithmetischen Ausdrucks wird ein Wert berechnet, der der links stehenden Variablen zugeordnet wird.

Weitere ergänzende Beispiele werden bei der Behandlung zyklischer PAP erläutert

(Kap. 3.6.1. (2), S. 41).

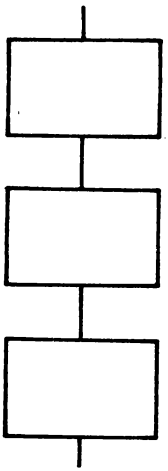
2.1.4. Grafisches Verbindungselement der Operationen

(1) Flußlinie

Die Flußlinie bei der Kästchenmethode ist eine senkrechte Gerade, die die Symbole der einzelnen Operationen zum eigentlichen Programmablauf verbindet.
Bei Einhaltung der Vorzugsrichtung von oben nach unten mündet die Linie **ohne** Pfeilspitze in das nächste Symbol ein.

Weitere Anwendungen werden in späteren Kapiteln behandelt

(3.6.1. (1), S. 40).



(2) Programmlinie

Die Programmlinie der Programmlinienmethode ist eine senkrechte oder waagerechte Gerade, die jedoch **nur** im senkrecht verlaufenden Teil des Ablaufs die Operationen miteinander verbindet.

2.2. Einfache lineare PAP (Sequenz)

Die einfachste Struktur haben lineare PAP. Ein solcher PAP ist dadurch gekennzeichnet, daß alle einzelnen Operationen nacheinander zur Verarbeitung kommen. Im einfachsten Falle kommen nur die bisher behandelten Grundsymbole vor.

Beispiele

(1) Auf einem Lochband stehen die Werte a und b zur Verarbeitung bereit ($a \neq b$). Mit ihnen ist der Wert

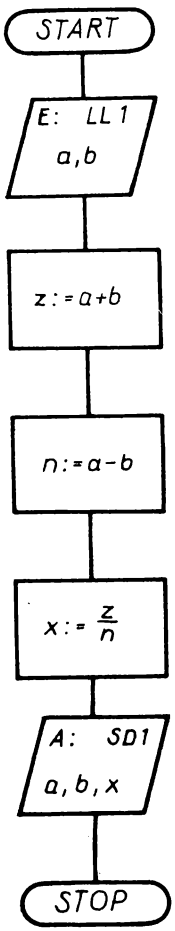
$$x = \frac{a+b}{a-b}$$

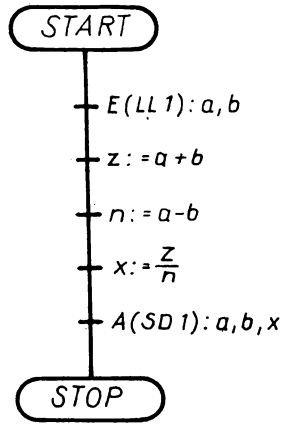
zu berechnen. Die gegebenen Werte und das Ergebnis sind zu drucken.

Die Lösung dieser Aufgabe erfordert folgende Schritte:

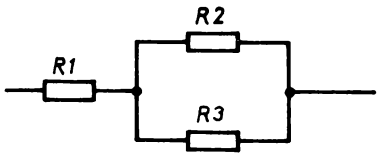
1. Eingabe der Werte a und b
2. Berechnung des Zählers
3. Berechnung des Nenners
4. Bildung des Quotienten
5. Ausgabe der Werte a, b und x

Damit erhält man folgende PAP-Darstellung:





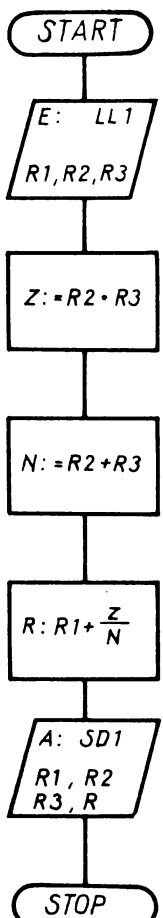
(2) Gegeben ist die skizzierte Widerstandskombination.

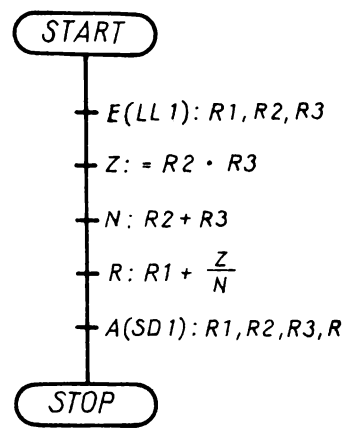


Die gegebenen Werte der drei Widerstände in Ohm stehen auf einem Lochband zur Verarbeitung bereit.
Stellen Sie einen PAP zur Berechnung des Gesamtwiderstandes der Schaltung auf!
Die gegebenen Werte und der berechnete Wert sind zu drucken.

Mathematisches Modell:

$R = R1 + R2 \parallel R3$ $\qquad R = R1 + \frac{R2 \cdot R3}{R2 + R3}$





2.3. Weitere Grundsymbole

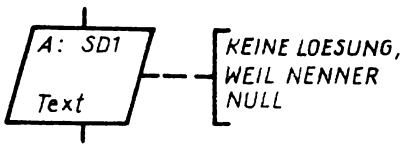
2.3.1. Bemerkungen

(1) Kästchenmethode

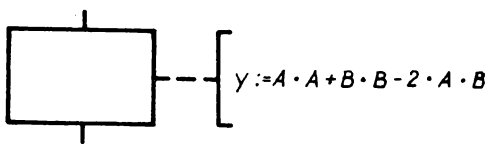
Reicht der Platz im Innern der PAP-Symbole für die Beschriftung nicht aus oder will der Programmierer ergänzenden oder erläuternden Text anbringen, kann dies durch eine öffnende eckige Klammer gekennzeichnet werden, die durch eine gestrichelte Linie mit dem Symbol verbunden ist.

Beispiele:

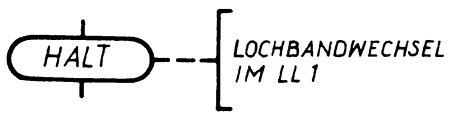
(1)



(2)



(3)

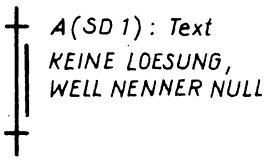


(2) **Programmlinienmethode**

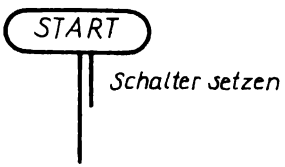
Erläuternde Texte werden an eine zur Programmlinie verlaufende Parallele geschrieben.

Beispiele:

(1)



(2)

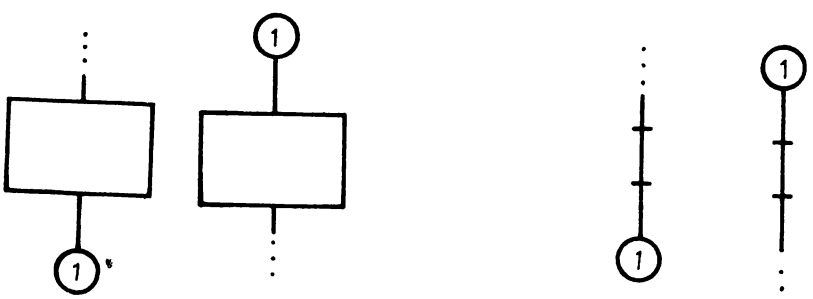


2.3.2. Konnektoren

Um die Übersichtlichkeit der Darstellung von PAP zu gewährleisten, sollen sich Programm- oder Flußlinien nach Möglichkeit nicht kreuzen. Auch Unterbrechungen der Programmlinie müssen klar gekennzeichnet sein.
Um diese Forderungen erfüllen zu können, gibt es das Symbol des Konnektors. Die TGL unterscheidet zwei Arten.

(1) Zeichenkonnektor
Er kennzeichnet eine Programmlinienunterbrechung mit Fortsetzung auf der **gleichen** Seite. Das Symbol ist ein Kreis, in den eine Zahl eingetragen wird.

Beispiel für die Anwendung von Zeichenkonnektoren



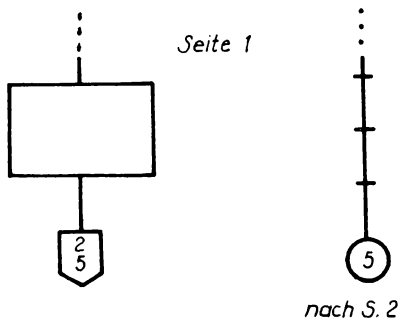
(2) Seitenkonnektor

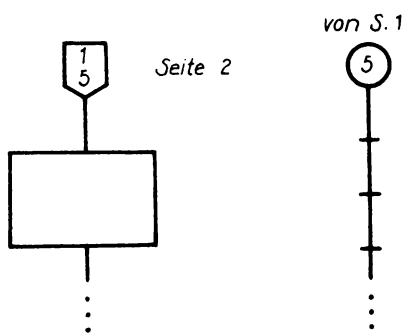
Er kennzeichnet eine Programmlinienunterbrechung mit Fortsetzung auf einer **anderen** Seite.

Bei der **Kästchenmethode** wird ein Fünfeck verwendet in das oben die Seiten-Nr. und darunter das Konnektorkennzeichen eingetragen wird.

Bei der **Progammlinienmethode** wird die Anschlußangabe an den Kreis herangeschrieben.

Beispiel für die Anwendung von Seitenkonnektoren

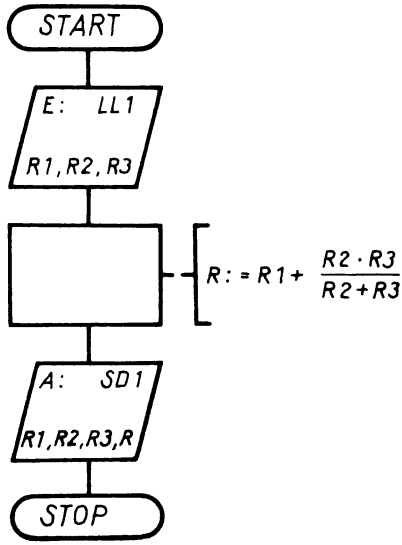




2.4. Weitere lineare PAP

Die Anwendung der neuen Grundsymbole soll an zwei Beispielen gezeigt werden.

(1) Beispiel (2) aus Kapitel 2.2. soll **ohne** Zerlegung der Lösungsformel in Teilschritte bearbeitet werden (nur Kästchenmethode):



(2) Für gegebenen Durchmesser eines Kreises (Eingabe über Bedienschreibmaschine) sollen berechnet werden:

- Kreisumfang
- Flächeninhalt des Kreises
- Widerstandsmoment
- Trägheitsmoment

Die Ergebnisse sind zu drucken.

Mathematisches Modell:

$$U = d \cdot \pi$$

$$A = \frac{d^2 \cdot \pi}{4}$$

$$W = \frac{d^3 \pi}{32}$$

$$I = \frac{d^4 \pi}{64}$$

Durch Verwendung einmal berechneter Zwischenergebnisse kann man schreiben:

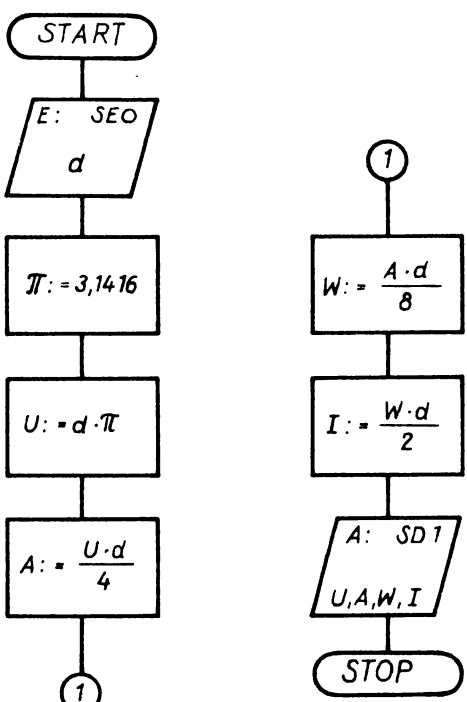
$$U = d \cdot \pi$$

$$A = \frac{U \cdot d}{4}$$

$$W = \frac{A \cdot d}{8}$$

$$I = \frac{W \cdot d}{2}$$

Damit ergibt sich folgender PAP (nur Kästchenmethode):



2.5. Übungen zu linearen PAP

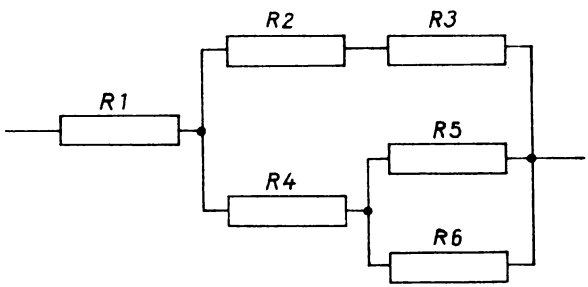
Es sind jeweils mathematisches Modell und PAP auszuführen. Außerdem sind Festlegungen zur Eingabe der gegebenen und zur Ausgabe der zu berechnenden Größen zu treffen.

(1) Von einem geraden Pyramidenstumpf mit quadratischer Grund- und Deckfläche werden gemessen:

Kantenlänge der Grundfläche	a cm
Kantenlänge der Deckfläche	b cm
Höhe	h cm

Programmieren Sie die Berechnung des Volumens und der Oberfläche des Pyramidenstumpfes.

(2) Gegeben ist folgende Schaltung ohmscher Widerstände:



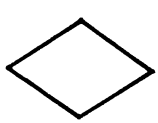
Programmieren Sie die Berechnung des Gesamtwiderstandes der Schaltung.

(3) Um die Tiefe eines Brunnens zu messen, läßt man einen Stein hineinfallen. Man hört das Aufschlagen nach x Sekunden.
Programmieren Sie die Berechnung der Tiefe des Brunnens.

3. Verzweigte und zyklische PAP

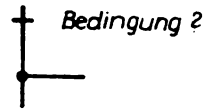
3.1. Einfache Verzweigungen

Viele praktische Aufgaben lassen zwei oder mehrere Möglichkeiten bei der Lösung zu. Um solche Aufgaben zu lösen, muß es möglich sein, ein Programm zu verzweigen. Das entsprechende PAP-Element nennt man Verzweigung.



In welchem Zweig dabei weitergearbeitet wird, ist von einer Bedingung abhängig.

Bei der Kästchenmethode wird die Bedingung als Frage in das Symbol eingetragen, bei der Programmlinienmethode steht sie neben dem Operationsstrich.



Im allgemeinen wird die Bedingung als mathematischer Vergleich formuliert.

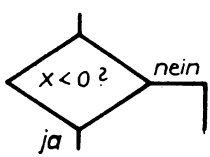
Die dabei verwendeten Vergleichsoperatoren sind:

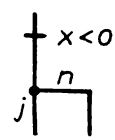
$<$ kleiner als	$\geqq$ größer als oder gleich
$\leqq$ kleiner als oder gleich	$>$ größer als
$=$ gleich	$\neq$ ungleich

Rechts und links von diesen Operatoren können arithmetische Ausdrücke stehen.
In Abhängigkeit von der Beantwortung der formulierten Entscheidung mit ja oder nein hat das Verzweigungssymbol zwei Ausgänge, die mit der Antwort beschriftet werden. Ihre grafische Anordnung kann unterschiedlich sein.

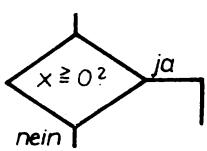
Beispiel für Verzweigungen:

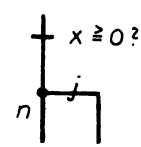
(1)



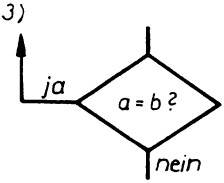


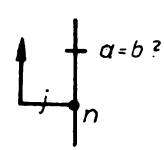
(2)

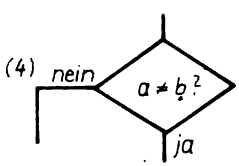


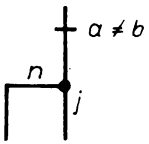


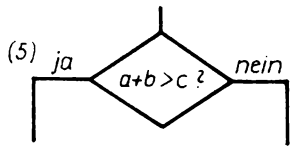
(3)

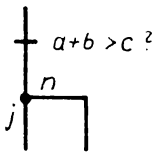


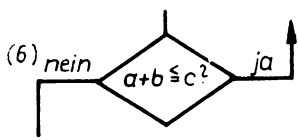


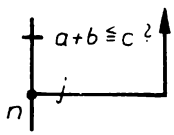








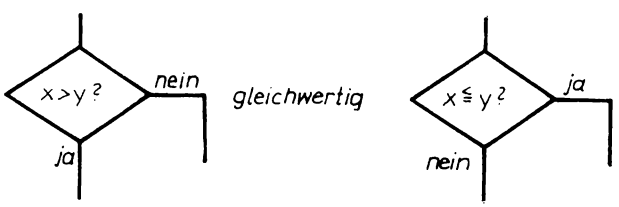




Aus zeichen- oder programmiertechnischen Gründen ist es oftmals notwendig, ja- und nein-Zweig zu vertauschen. Das ist sehr einfach möglich, wenn man zugleich den Vergleichsoperator durch seine Negation ersetzt. Nachstehende Beispiele sollen das zeigen.

Beispiel für gleichwertige Verzweigungen:

(1) (Kästchenmethode)

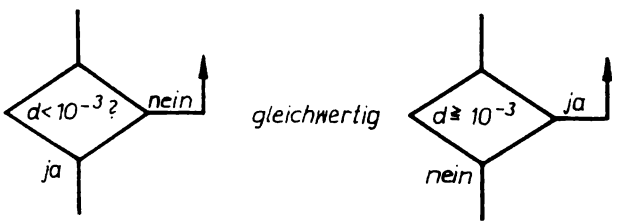


(Programmlinienmethode)

(2) (Kästchenmethode)

(Programmlinienmethode)

(3) (Kästchenmethode)



(Programmlinienmethode)

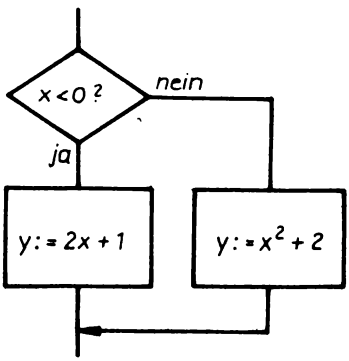
Jede Programmverzweigung wird also durch eine Bedingung ausgelöst. Bei ihrer Erfüllung wird eine bestimmte Anweisung oder Anweisungsfolge (der ja-Zweig), bei ihrer Nichterfüllung eine andere Anweisung oder Anweisungsfolge (der nein-Zweig) abgearbeitet.

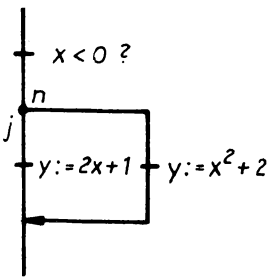
Dazu einige Beispiele:

(1) Eine Funktion ist definiert

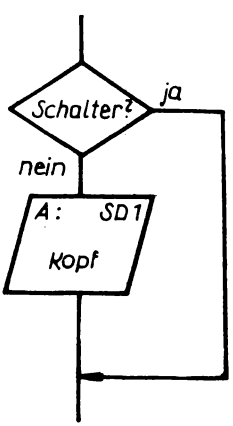
$$y = \begin{cases} 2x + 1 & \text{für } x < 0 \\ x^2 + 2 & \text{für } x \geqq 0 \end{cases}$$

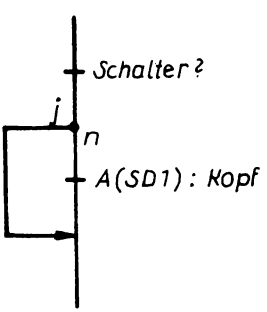
Die Funktionswerte sollen für beliebige x berechnet werden.
Folgende Arbeitsschritte sind notwendig:
– prüfe, ob $x < 0$ ist
– wenn ja, berechne $y = 2x + 1$
– wenn nein, berechne $y = x^2 + 2$
– danach wird fortgesetzt



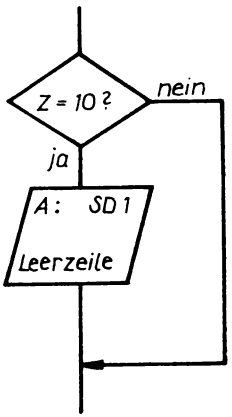


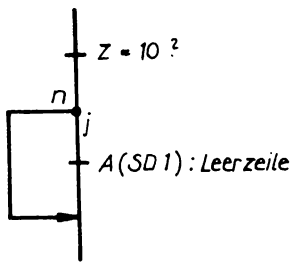
(2) Jede Druckliste beginnt mit einem Listenkopf. Bei Verwendung von weißem Papier ist dieser zu drucken. Benutzt man einen Vordruck ist dieser Schritt zu unterdrücken, wozu man das Setzen eines Schalters (Selektor) am Bedienpult der EDVA benutzen kann.
Folgende Arbeitsschritte sind notwendig:
– prüfe, ob Schalter gesetzt ist
– wenn ja, erfolgt kein Druck eines Kopfes
– wenn nein, erfolgt der Druck des Kopfes
– danach wird fortgesetzt





(3) Beim Druck größerer Listen wird zur besseren Übersicht nach einer bestimmten Anzahl von Zeilen eine Leerzeile vorgesehen, z. B. nach 10 Zeilen.
Folgende Arbeitsschritte sind notwendig:
– prüfe, ob Zeilenzahl erreicht ist (z = 10)
– wenn ja, Leerzeile realisieren
– wenn nein, keine Leerzeile nötig
– danach wird fortgesetzt





Hierbei ist zu beachten, daß beim Zusammenführen von Flußlinien die einmündende Linie durch eine Pfeilspitze zu kennzeichnen ist.

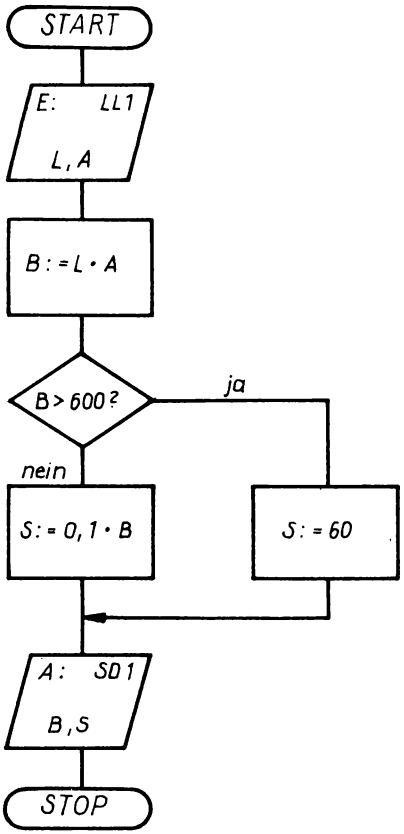
3.2. Einfache Verzweigung PAP (Alternative)

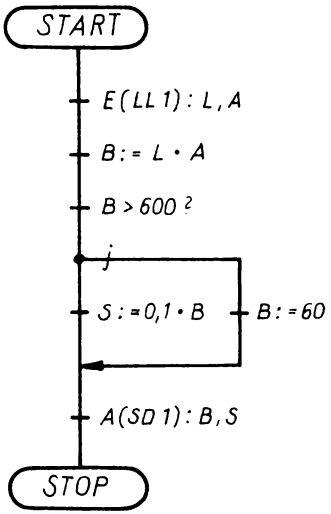
Sind in einem PAP von einer bestimmten Stelle an (Verzweigung) verschiedene Wege möglich, spricht man von einem verzweigten Programm. In welchem Zweig die Abarbeitung fortgesetzt wird, ist von der Erfüllung oder Nichterfüllung der als Frage formulierten Bedingung abhängig. Charakteristisch für diese PAP ist, daß in beiden Zweigen die Ablaufrichtung gleich ist.

Beispiele für verzweigte PAP

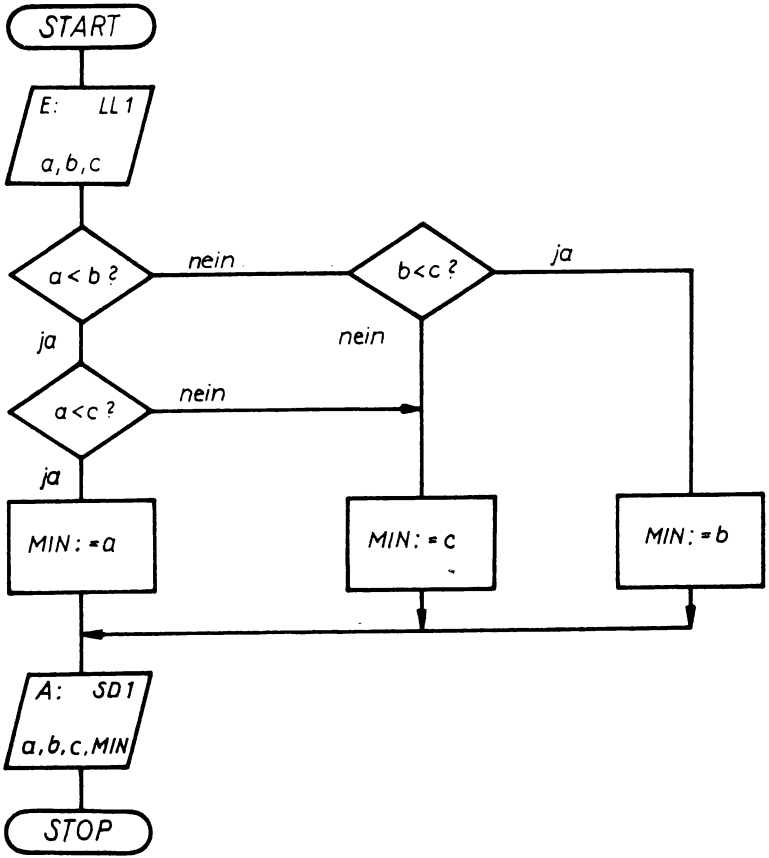
(1) Für eine vereinfachte Bruttolohnrechnung ist der Sozialversicherungsbeitrag zu ermitteln. Auf einem Lochband stehen Stundenlohn L und Anzahl der monatlichen Arbeitsstunden A zur Eingabe bereit.
Der Bruttolohn errechnet sich nach der Beziehung
$B = A \cdot L$

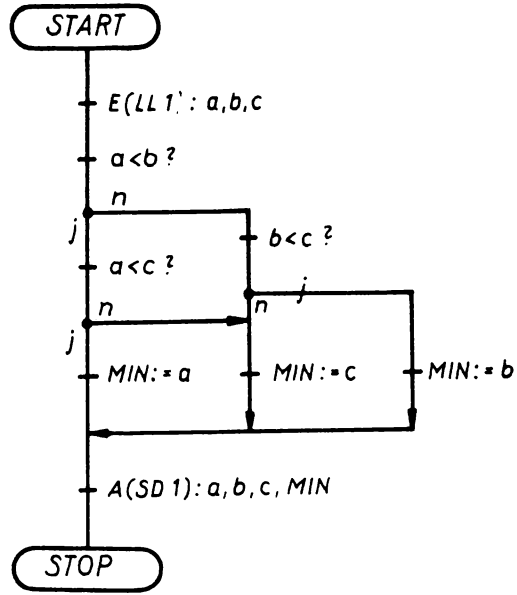
Ist der Bruttolohn größer als 600,– M, sind konstant 60,– M als SV-Beitrag einzusetzen, anderenfalls beträgt der SV-Beitrag 10% des Bruttolohns.
Bruttolohn und SV-Beitrag sind zu drucken.



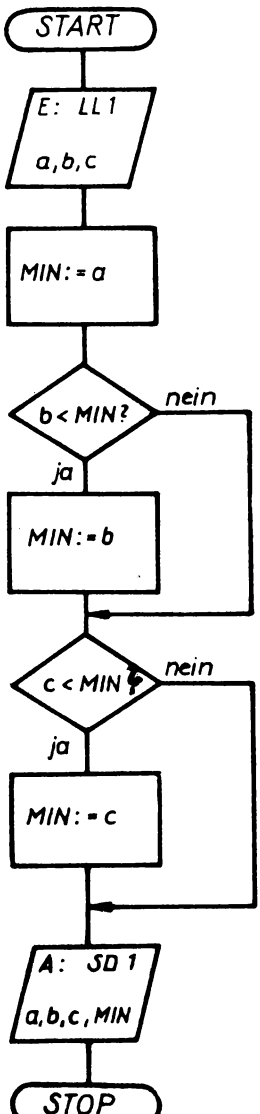


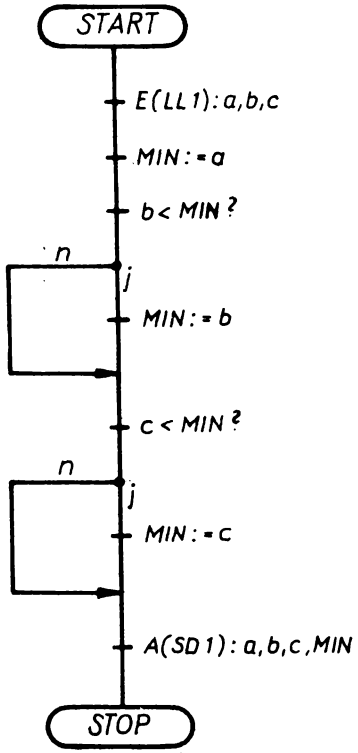
(2) Von drei gegebenen Zahlen a, b, c ist die kleinste (Minimum) zu bestimmen. Die drei Zahlen sind über Lochband einzugeben. Die drei Zahlen und das gefundene Minimum sind zu drucken.


START
E: LL1
a,b,c
a < b ?
nein
b<c ?
ja
ja
nein
a<c ?
nein
ja
MIN: = a
MIN: = c
MIN: = b
A: SD1
a,b,c,MIN
STOP


START
E(LL1): a,b,c
a<b ?
n
j
b<c ?
a<c ?
j
n
n
j
MIN: = a
MIN: = c
MIN: = b
A(SD1): a,b,c, MIN
STOP

(3) Dieses Beispiel soll nochmals auf andere Art und Weise, mit einer Verzweigung weniger gelöst werden. Dabei wird eine Zahl einfach als Minimum vorausgesetzt, und die beiden anderen werden damit verglichen.





Die Metnode dieses dritten Beispiels läßt sich auch zur Ermittlung des Maximums oder des Minimums von mehr als drei Zahlen verwenden.

3.3. Trockentest

Bevor man nach der PAP-Vorlage mit der Programmierung beginnt, sollte man den PAP auf rechnerische und logische Richtigkeit sowie auf Vollständigkeit überprüfen.

Eine dafür geeignete Methode nennt man „Trockentest". In ein tabellenartiges Schema trägt man alle im PAP vorkommenden Symbolde für Speicherplätze ein und überprüft den PAP mit einfachen, überschaubaren Zahlenwerten so oft, daß alle Zweige mindestens einmal durchlaufen werden. Mitunter ist es sinnvoll, auch die Verzweigungen und die Ein- und Ausgaben mit einzubeziehen.

Am Beispiel (2) des Kapitels 3.2. soll ein solcher Trockentest vorgeführt werden:

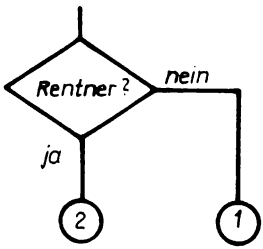
	Eingabe			Verzweigungen				Ausgabe			
	a	b	c	a < b	a < c	b < c	MIN	a	b	c	MIN
1.	10	15	18	ja	ja	–	10	10	15	18	10
2.	10	18	15	ja	ja	–	10	10	18	15	10
3.	15	10	18	nein	–	ja	10	15	10	18	10
4.	15	18	10	ja	nein	–	10	15	18	10	10
5.	18	10	15	nein	–	ja	10	18	10	15	10
6.	18	15	10	nein	–	nein	10	18	15	10	10

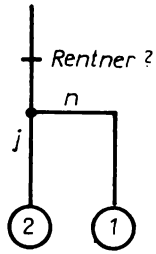
Für das Beispiel (3) des Kapitels 3.2. führen Sie selbst den Trockentest aus. Gewöhnen Sie sich daran, von nun an die Produkte Ihres schöpferischen Denkprozesses durch einen Trockentest zu prüfen.

3.4. Kombinierte Verzweigungen

3.4.1. Verwendung logischer Funktionen

Durch die Verwendung der logischen Funktionen „Konjunktion" und „Disjunktion" ist es möglich, mehrere Verzweigungen zu einer zu reduzieren.
Dazu ein Beispiel:
Bei der Verarbeitung von Personenstammdaten soll gefragt werden, ob eine Person bereits Altersrentner ist.

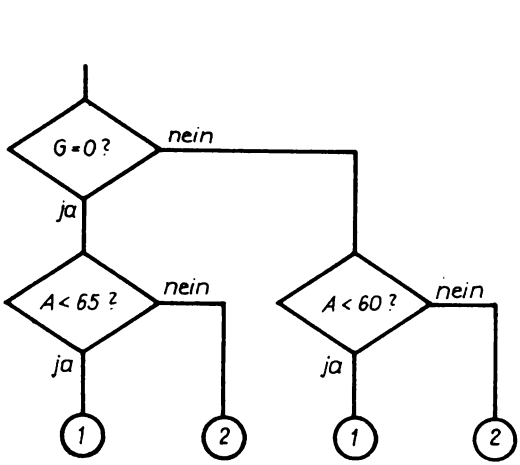


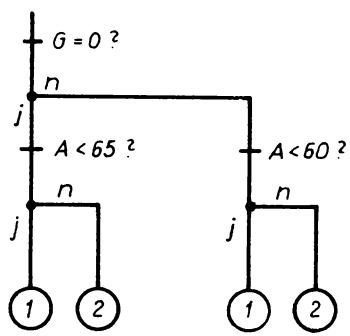


Eine EDVA kann diese Entscheidung natürlich nur in Abhängigkeit von konkreten Daten treffen. Dazu seien das Alter und ein Geschlechtskennzeichen für jede Person gegeben:

Alter	A	
Geschlechtskennzeichen	G ; G	= 0 männlich = 1 weiblich

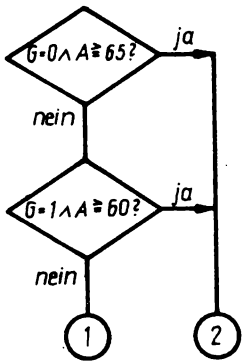
Mit diesen Angaben läßt sich die geforderte Verzweigung wie folgt realisieren:

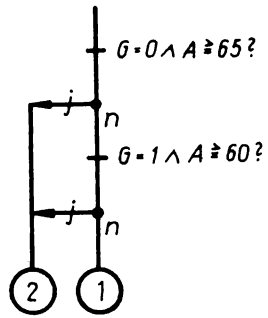




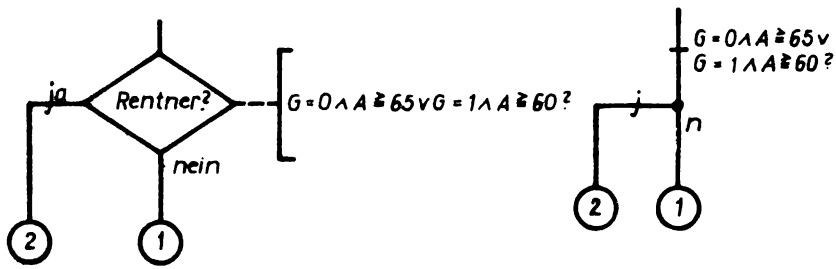
Fortsetzung bei ① bedeutet „kein Rentner",
Fortsetzung bei ② bedeutet „Rentner".

Mit Verwendung der „Konjunktion" kann man dafür mit geringerem zeichentechnischen Aufwand das Problem folgendermaßen lösen:





Benutzt man darüber hinaus auch noch die „Disjunktion" ergibt sich folgende Vereinfachung:



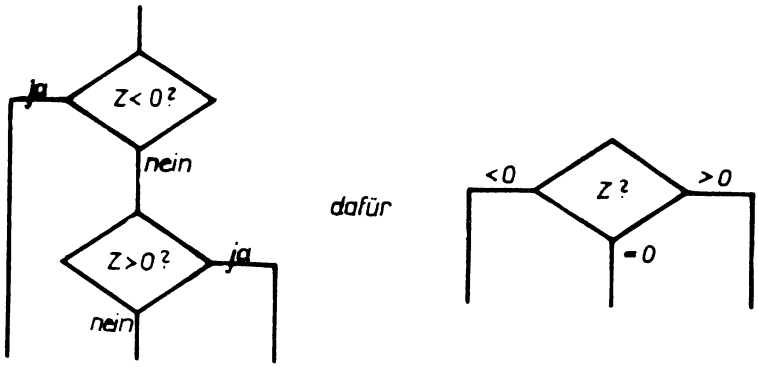
3.4.2. Dreifachverzweigung

In verschiedenen Programmiersprachen gibt es Befehle, die in Abhängigkeit vom Vorzeichen eines Zahlenwertes einen Sprung ausführen. Dabei wird unterschieden zwischen:

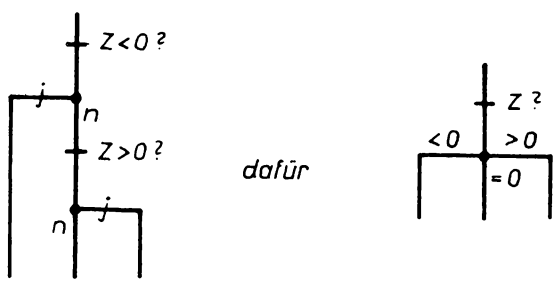
- Zahlenwert ist kleiner als null (negativ)
- Zahlenwert ist gleich null
- Zahlenwert ist größer als null (positiv)

Das entspricht praktisch einer „Dreifachverzweigung".
Es wird empfohlen, in Anlehnung an die Darstellung nach TGL 22451 folgende Symbolik zu verwenden:

(1) Kästchenmethode



(2) Programmlinienmethode



3.5. Übungen zu verzweigten PAP

Es sind jeweils mathematisches Modell und PAP auszuführen.

(4) Gegeben ist die Funktion

$$y = f(x) = \frac{a}{b \cdot \sqrt{a + b}}$$

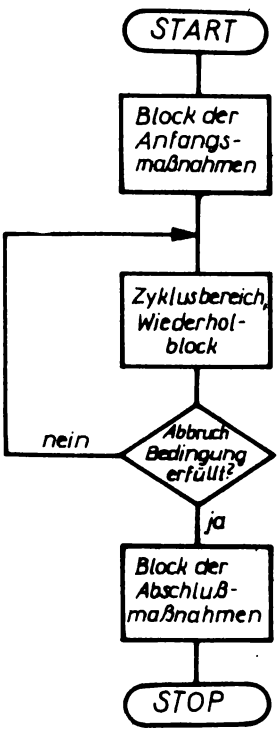
Programmieren Sie die Berechnung des Funktionswertes, wenn die Werte für a und b auf Lochband zur Verarbeitung bereit stehen. Diese beiden Werte und der berechnete Funktionswert sind zu drucken. Wird der Nenner null oder der Radikand negativ, ist der Text „Funktion nicht definiert" zu drucken.

(5) Auf Lochband stehen die Koordinaten eines Punktes P(x; y) zur Verarbeitung bereit (x und y ungleich null).
Ermitteln Sie, in welchem Quadranten der Punkt liegt und drucken Sie diese Angabe.

3.6. Zyklische PAP

3.6.0. Allgemeines

Bei vielen Problemen der Praxis (z. B. Lohnrechnung, Materialbestandsrechnung, Kontenverwaltung) sind die gleichen Anweisungen für eine Vielzahl gleichartiger Objekte (z. B. Beschäftigte, Materialarten, Konten) durchzuführen. Es ist dann rationell, die sich wiederholenden Anweisungen nur einmal zu programmieren und durch organisatorische Maßnahmen zu gewährleisten, daß deren Wiederholung jeweils mit veränderten Informationen – mit den Daten des nächsten gleichartigen Objekts – erfolgt und daß die Folge der Wiederholungen nach Erreichen eines bestimmten Zieles abgebrochen wird. So entsteht ein Zyklus mit folgender allgemeinen Struktur:



Der Block der Anfangsmaßnahmen beinhaltet alle **einmalig** abzuarbeitenden Anweisungen, die **vor** Bearbeitung des **ersten** Objekts der Objektgruppe erforderlich sind.

Der Zyklusbereich umfaßt alle die Anweisungen, die für jedes Objekt der Objektgruppe in gleicher Weise abzuarbeiten sind. Dieser Wiederholblock kann lineare Struktur haben, verzweigt sein oder auch selbst wieder ein vollständiger Zyklus sein.

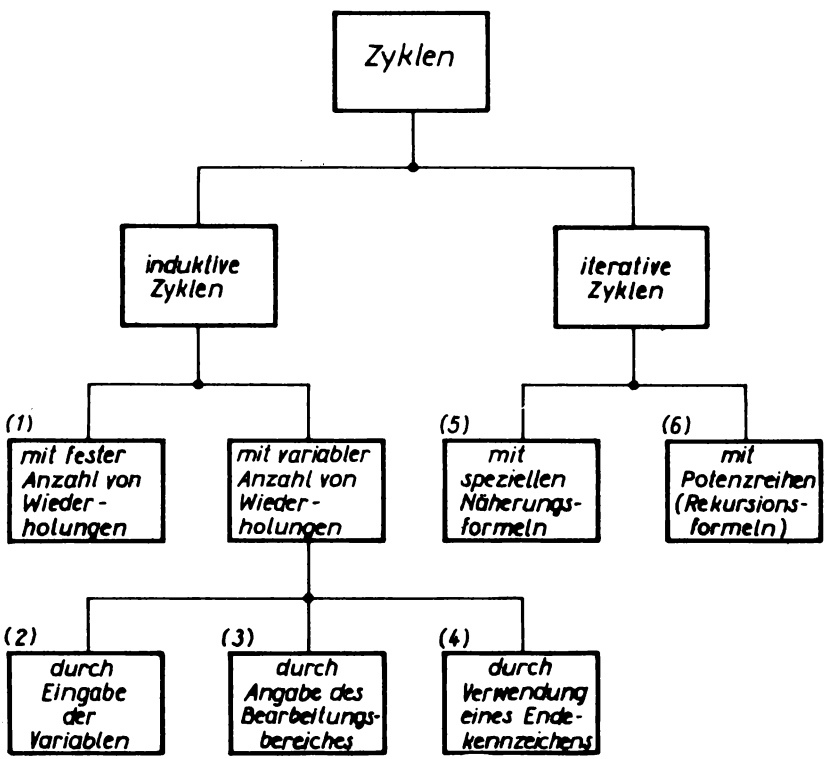
Der Block der Abschlußmaßnahmen beinhaltet alle **einmalig** abzuarbeitenden Anweisungen, die **nach** Bearbeitung des **letzten** Objekts der Objektgruppe erforderlich sind.

Die Abbruchbedingung überprüft, ob das durch die Aufgabenstellung festgelegte Ziel durch die erfolgten Wiederholungen erreicht wurde. In Abhängigkeit von zwei Grundsatzzielen unterscheidet man zwei Hauptarten zyklischer PAP:

(1) Ziel: Sind alle Objekte der Objektgruppe bearbeitet?
Art: Induktiver Zyklus

(2) Ziel: Ist das Ziel mit hinreichender Genauigkeit erreicht?
Art: Iterativer Zyklus

An dieser Stelle soll eine Übersicht über die einzelnen Zyklusarten gegeben werden.



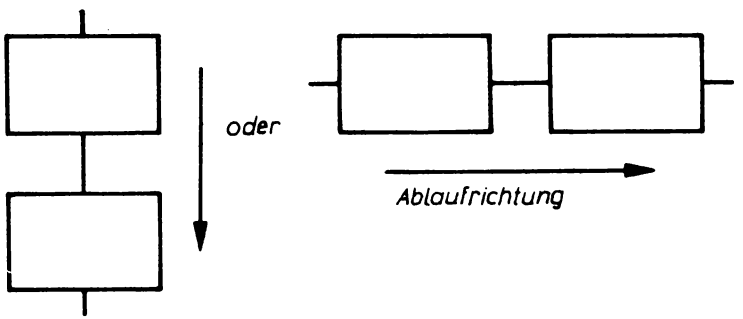
Die sechs bezeichneten Arten werden nachfolgend genauer behandelt.

3.6.1. Besonderheiten der Darstellung

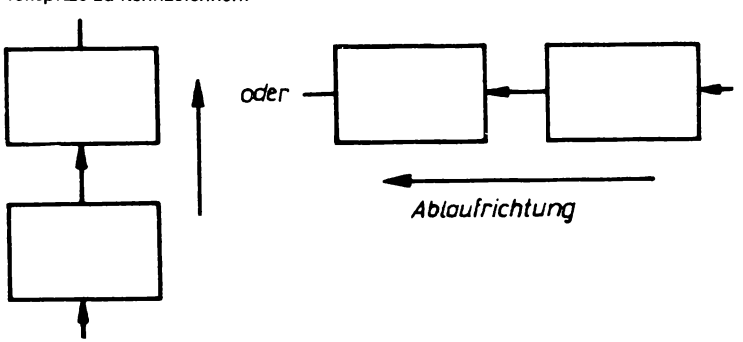
(1) **Flußlinie**

Die Flußlinie bei der Kästchenmethode ist eine senkrechte oder waagerechte Gerade, die die Sinnbilder der einzelnen Operationen zum eigentlichen Programmablauf verbindet. Dabei sind zwei Vorzugsrichtungen definiert:
(1) von oben nach unten und
(2) von links nach rechts.

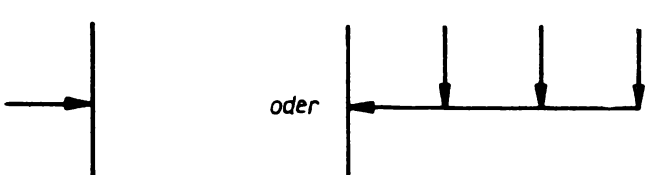
Bei Einhaltung dieser Vorzugsrichtungen mündet die Linie **ohne** Pfeilspitze in das nächste Sinnbild ein:



Bei Abweichung von diesen Vorzugsrichtungen ist die Ablaufrichtung jedoch durch eine Pfeilspitze zu kennzeichnen:



Zusammenführungen von Flußlinien sind **immer** durch Pfeilspitzen zu kennzeichnen:

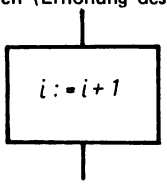


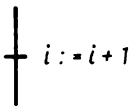
Um Irrtümer auszuschließen, sind Kreuzungen von Flußlinien möglichst zu vermeiden. Die Hauptflußlinie darf niemals gekreuzt werden.

(2) **Ergibt-Anweisung**

Bei zyklischen Abarbeitungen werden häufig Summenbildungen auf einem Speicherplatz vorgenommen. Dazu zwei Beispiele:

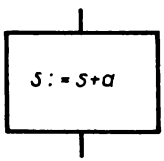
1. Zählen (Erhöhung des Wertes um 1)

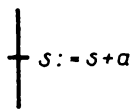




i ergibt sich aus i plus 1, d. h., der Variablen i wird der Wert der Summe aus i und 1 zugewiesen. Zum Wert der Speicherzelle mit der symbolischen Adresse i wird eins addiert. Danach wird die Speicherzelle mit der symbolischen Adresse i mit dem Wert der Summe belegt.

2. Addieren des Wertes a





s ergibt sich aus s plus a, d. h., der Variablen s wird der Wert der Summe aus s und a zugewiesen. Die Werte der Speicherzellen mit den symbolischen Adressden s und a werden addiert. Danach wird die Speicherzelle mit der symbolischen Adresse s mit dem Wert der Summe belegt.

Diese beiden Beispiele zeigen den Sonderfall einer kumulativen Berechnung, bei der der Wert einer Speicherzelle auf beiden Seiten der Ergibt-Anweisung vorkommt. Auf der rechten Seite handelt es sich um den „alten" Wert, um den Wert, der **vor** der Verknüpfung auf der Speicherzelle steht. Auf der linken Seite handelt es sich um den „neuen" Wert, um den Wert, der **nach** der Verknüpfung auf der Speicherzelle steht. Selbstverständlich geht der „alte" Wert dabei verloren, er wird vom „neuen" Wert überspeichert.

3.6.2. Induktive Zyklen mit fester Anzahl von Wiederholungen (Fall (1) der Übersicht auf Seite 39)

An dieser einfachsten, allerdings selten vorkommenden Zyklusart sollen einige Grundsätze der Konstruktion von Zyklen demonstriert werden. Hierzu ist in fast allen Fällen nötig, eine Hilfsvariable (Zähler genannt) einzuführen. Häufig werden für einen solchen Zähler die Variablen i, j und k benutzt.

Wahl des Anfangswertes des Zählers, Formulierung der Bedingung und die Stelle des Weiterzählens des Zählers hängen dabei eng zusammen. Drei Varianten sollen durch folgendes Beispiel erläutert werden:

Auf einem Lochband stehen 130 Zahlenwerte zur Verarbeitung bereit. Aus ihnen ist der arithmetische Mittelwert zu berechnen und zu drucken.

Das mathematische Modell

$$M = \frac{1}{130} \cdot \sum_{i=1}^{130} Z_i$$

beinhaltet die Addition von 130 Zahlen, die mit der Anweisung S : = S + Z im Zyklus 130mal ausgeführt wird. Dazu muß die Speicherzelle mit der Bezeichnung S den Anfangswert null erhalten

Variante (1):

- Vor dem Zyklusbereich wird der Zähler auf den Anfangswert gesetzt.
- erst nach der Verzweigung (im nein-Zweig) wird der Zähler weitergestellt,
- im Test wird gefragt, ob der Endwert erreicht wurde.

Bild siehe Seite 43

Variante (2):

- Vor dem Zyklusbereich wird der Zähler mit einem Wert belegt, der um die Schrittweite geringer ist als der Anfangswert,
- zu Beginn des Zyklusbereichs wird der Zähler weitergestellt – beim ersten Mal wird damit der Anfangswert erreicht.
- im Test wird gefragt, ob der Endwert erreicht wurde.

Bild siehe Seite 44

Variante (3):

- Vor dem Zyklusbereich wird der Zähler auf den Anfangswert gesetzt,
- zwischen Zyklusbereich und Verzweigung wird der Zähler weitergestellt,
- im Test wird gefragt, ob der Endwert des Zählers überschritten wurde.

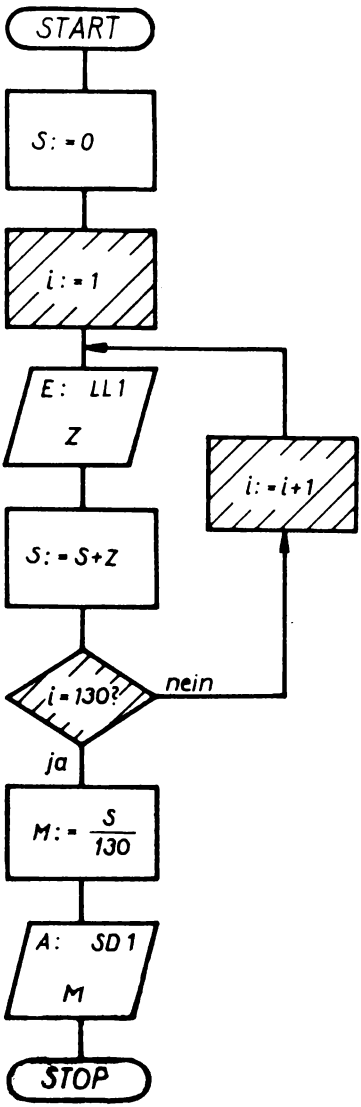
Bild siehe Seite 45

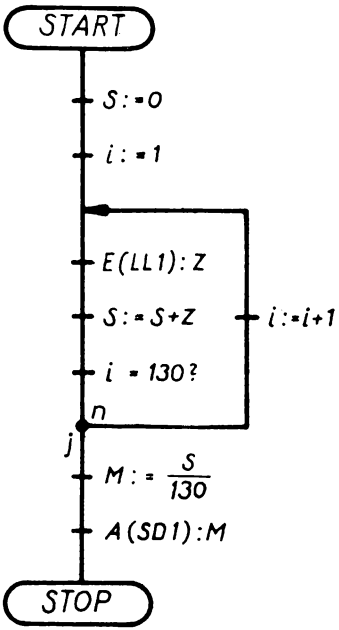
3.6.3. Laufanweisungen

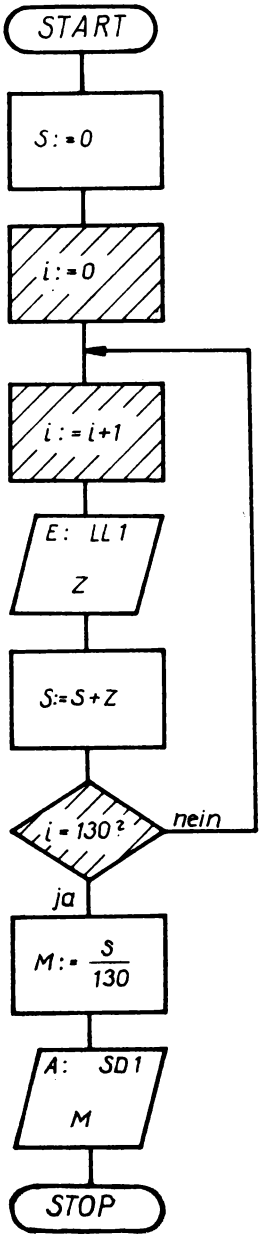
Natürlich ist jedem Programmierer überlassen, welchen Weg er wählt. Es hat sich durch die praktischen Erfahrungen jedoch ergeben, den Weg der Variante 3 als „Standardzyklus" zu betrachten. In Hinblick auf Formulierungen in den problemorientierten Programmiersprachen kann dafür ein besonderes Symbol benutzt werden. Man nennt es **Programmodifikation.** Es symbolisiert die Änderung eines Befehls durch das Programm selbst.
Dieses Symbol ist nach /3/ nur für die Kästchenmethode vorgesehen. Eine entsprechende Darstellung in der Programmlinienmethode gibt es **nicht.**

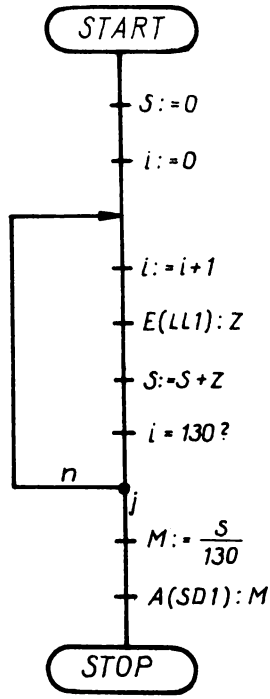
In verschiedener Literatur werden von den Autoren eigene Lösungen vorgestellt, um den Vorteil einer Laufanweisung zu nutzen. Der standardisierten Darstellung der Kästchenmethode soll für die Programmlinienmethode die geringfügig abgewandelte Darstellung nach /7/ in diesem Lehrbrief gegenübergestellt werden.

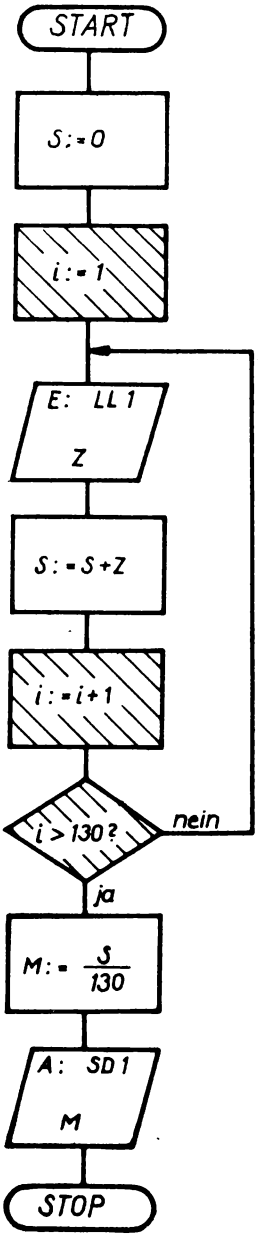
Fortsetzung des Textes zu 3.6.3. **Laufanweisungen** auf Seite 46!

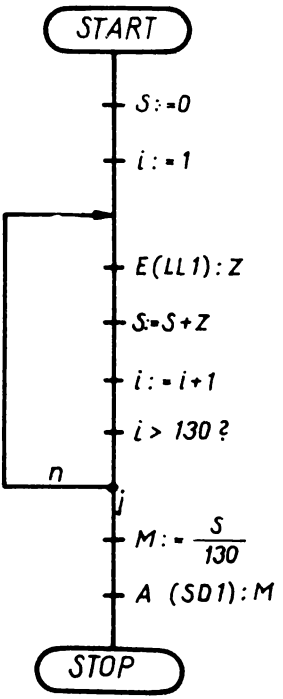

START
S: = 0
i: = 1
E: LL1
Z
i: = i+1
S: = S+Z
i = 130?
nein
ja
M: = S/130
A: SD1
M
STOP


START
S: = 0
i: = 1
E(LL1): Z
S: = S+Z
i: = i+1
i = 130?
n
j
M: = S/130
A(SD1): M
STOP

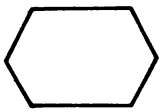

START
S:=0
i:=0
i:=i+1
E: LL1
Z
S:=S+Z
i=130?
nein
ja
M:= S/130
A: SD1
M
STOP

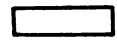

START
S:=0
i:=0
i:=i+1
E(LL1):Z
S:=S+Z
i=130?
n
j
M:= S/130
A(SD1):M
STOP


START
S:=0
i:=1
E: LL1
Z
S:=S+Z
i:=i+1
i>130?
nein
ja
M:= S/130
A: SD1
M
STOP

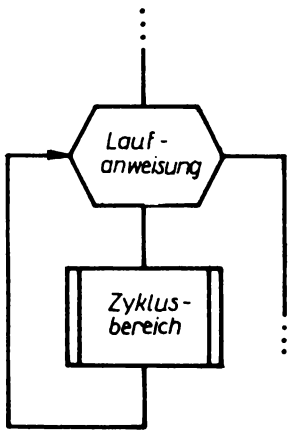

START
S:=0
i:=1
E(LL1):Z
S:=S+Z
i:=i+1
i>130?
n
j
M:= S/130
A (SD1):M
STOP

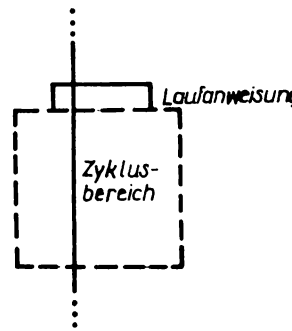
– Symbol:





Diese Symbole stehen immer in Verbindung mit einem Zyklusbereich, deren Darstellung dann allgemein folgendermaßen erfolgt:

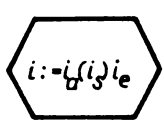


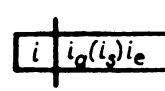


In einer Laufanweisung werden einer Laufvariablen (z. B. Zählgröße) ihr Anfangswert, ihr Endwert und die Schrittweite zugeordnet. In der üblichen mathematischen Schreibweise steht zuerst der Anfangswert, dann in Klammern die Schrittweite und danach der Endwert:

$i: = i_a\ (i_s)\ i_e$

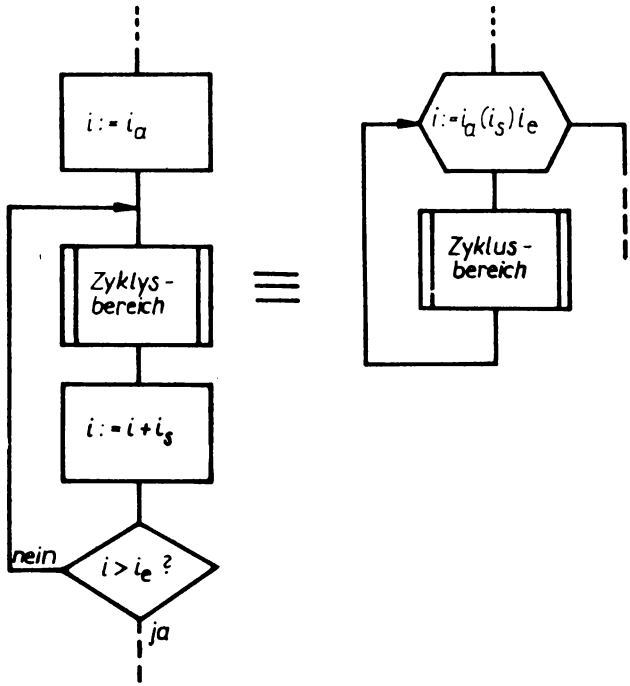
Da es allgemein üblich ist, bei der PAP-Beschriftung die mathematischen Schreibweisen anzuwenden, soll das auch hier so durchgeführt werden.



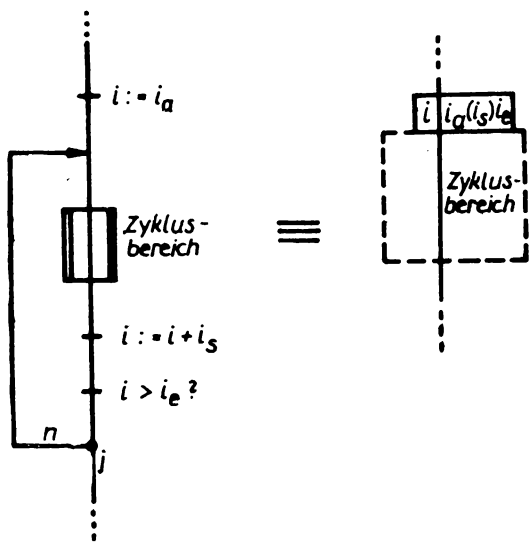


Durch die Anwendung des Symbols für die Laufanweisung ergibt sich eine Vereinfachung der Darstellung des „Standardzyklus".

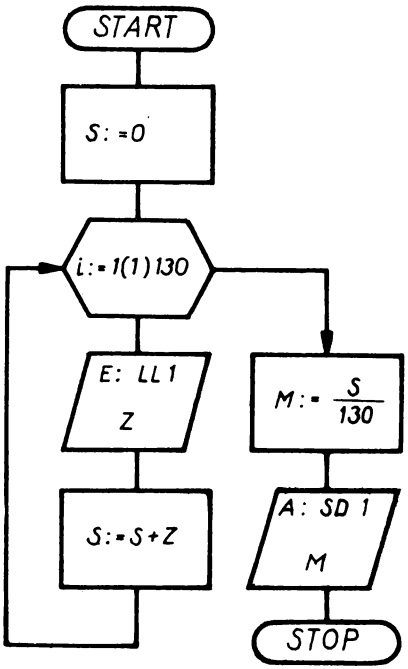
(Kästchenmethode)

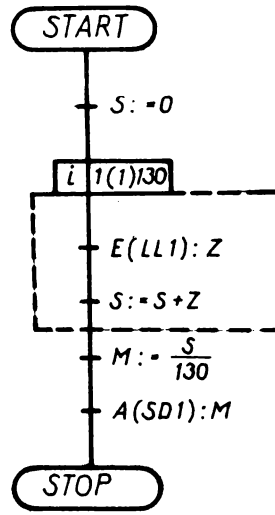


(Programmlinienmethode)



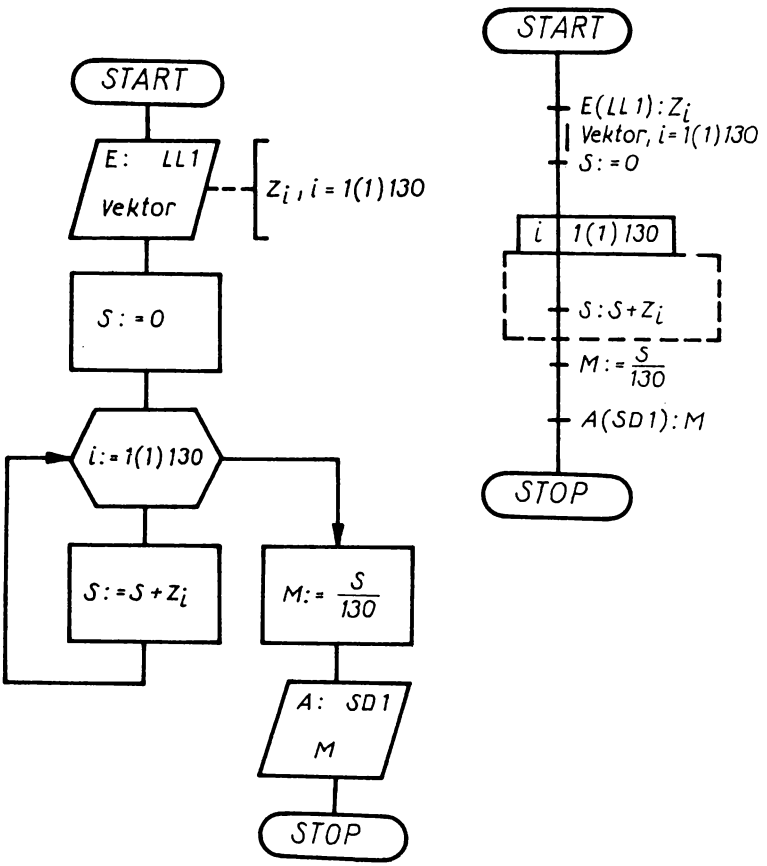
Für die Darstellung des unter 3.6.2. behandelten Beispiels erhält man damit eine 4. Variante:





Bei der Betrachtung dieses Beispiels waren wir davon ausgegangen, daß die Zahlenwerte einzeln zur Verarbeitung bereitstehen, wodurch es möglich ist, dieselben jeweils im Zyklusbereich einzulesen. Stehen diese Werte jedoch als geschlossene Zahlenmenge (Feld) zur Verfügung, erfolgt die Eingabe als Feld zu Beginn des Programms, also vor dem Zyklusbereich im Block der Anfangsmaßnahmen.

Damit ergibt sich eine 5. Variante des behandelten Beispiels:



Die in diesem Kapitel vorgestellten Zyklen in direkter Angabe der Anzahl der Wiederholungen gestatten lediglich, Spezialfälle zu behandeln.

Durch die ausführliche Darstellung sollten die grundsätzlichen Methoden eines Zyklusaufbaus erklärt werden.

3.6.4. Induktive Zyklen
mit variabler Anzahl von Wiederholungen durch Eingabe der Variablen
(Fall (2) der Übersicht auf Seite 39)

Die letzten beiden Kapitel behandelten als Beispiel die Mittelwertberechnung aus genau 130 Zahlenwerten. Ein Programm wird aber erst dann leistungsfähig, wenn es die Bearbeitung aller Aufgaben einer Aufgabenklasse (z. B. Berechnung des Mittelwertes einer Zahlenfolge) ermöglicht. Das erfordert eine variable Anzahl der zu bearbeitenden Objekte. Dabei muß die Variable im Block der Anfangsmaßnahmen zur Verfügung gestellt werden. Dazu ein Beispiel:

Für die tägliche Ermittlung des Einzelhandelsumsatzes einer Verkaufsstelle stehen die erforderlichen Daten auf Lochband zur Verarbeitung bereit.

Variante 1:

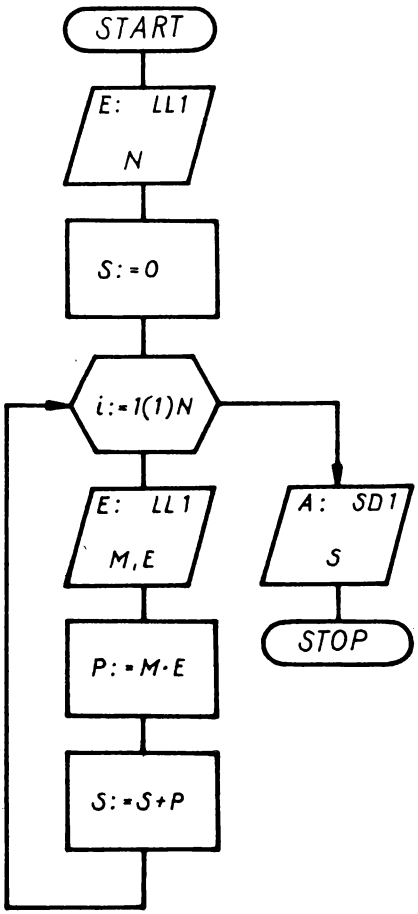
Das Lochband hat folgende Struktur:

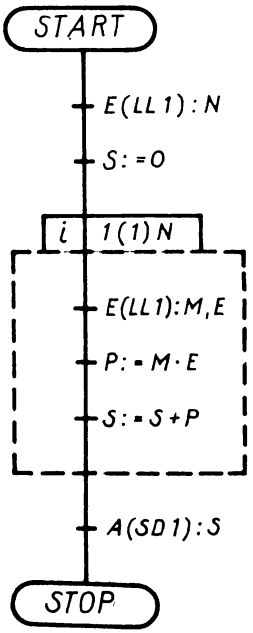
d. h. nach der Variablen N stehen N Zahlenpaare M (Anzahl der verkauften Exemplare eines Artikels) und E (Einzelverkaufspreis des Artikels) zur Verarbeitung bereit. Die Gesamteinnahme S ist zu berechnen und zu drucken.

Mathematisches Modell:

$$S = \frac{1}{N} \cdot \sum_{i=1}^{N} M \cdot E$$

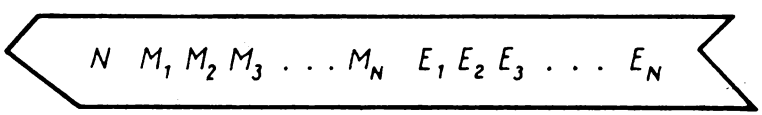
Bild siehe Seite 51



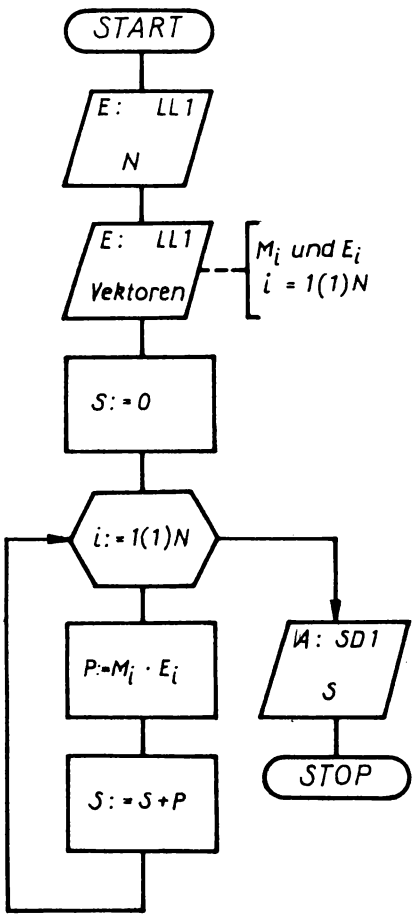


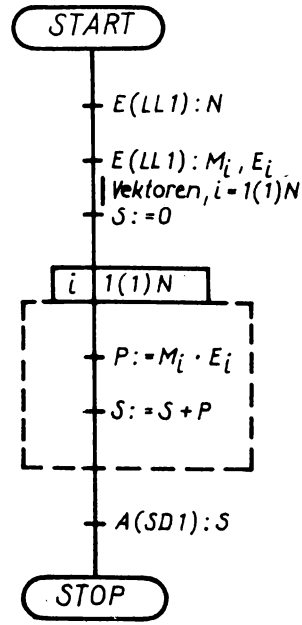
Variante 2:

Das Lochband hat folgende Struktur:



d. h. nach der Variablen N stehen erst alle M-Werte, danach alle E-Werte als Vektoren auf dem Lochband. In diesem Falle erfolgt die gesamte Eingabe im Block der Anfangsmaßnahmen vor Beginn des Zyklus.





Erarbeiten Sie für beide Varianten einen Trockentest und machen Sie sich daran Vor- und Nachteile beider Varianten klar!

3.6.5. Induktive Zyklen
mit variabler Anzahl von Wiederholungen durch Angabe des Bearbeitungsbereiches (Fall (3) der Übersicht auf Seite 39)

Während die bisher behandelten Zyklen immer den Wert 1 als Anfangswert und als Schrittweite benutzten (reine Zählzyklen), können bei dieser Art beide Werte von 1 verschieden sein. Man spricht in diesem Falle auch von **Wertetabellenprinzip.**

Dazu ein Beispiel:

Für die Funktionen

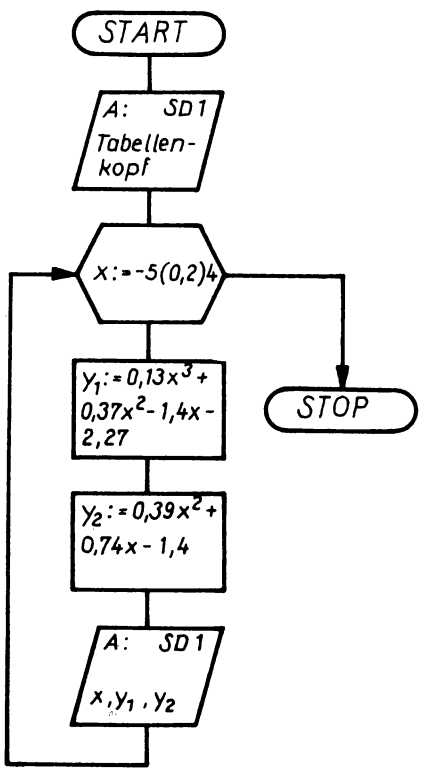
$y_1 = 0{,}13\,x^3 + 0{,}37\,x^2 - 1{,}4\,x - 2{,}27$ und

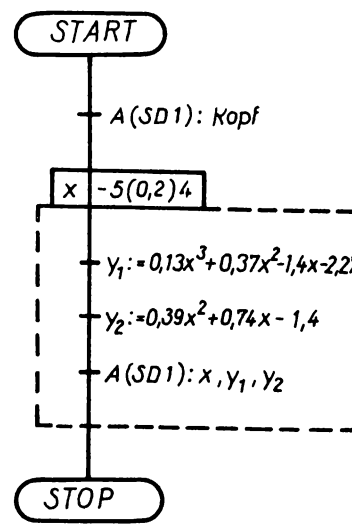
$y_2 = 0{,}39\,x^2 + 0{,}74\,x - 1{,}4$

ist im Bereich

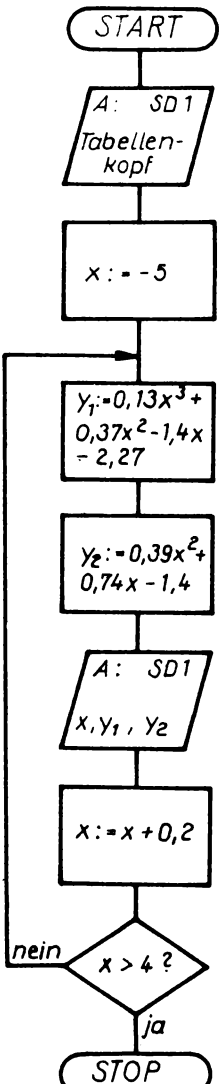
$x = -5\ (0{,}2)\ 4$

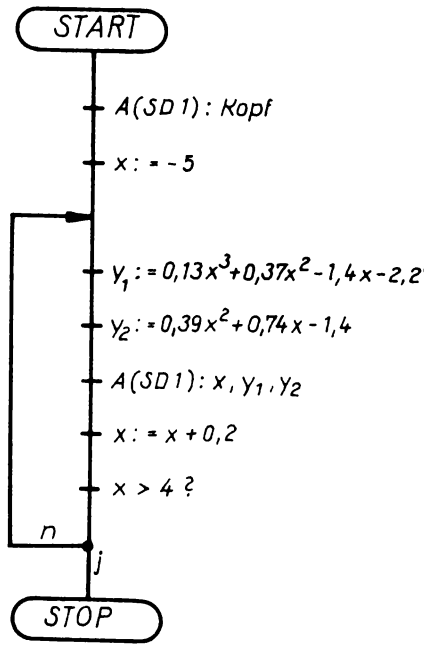
eine Wertetabelle zu berechnen und zu drucken.





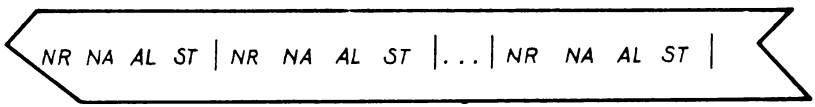
Dieser Zyklus läßt sich auch ausführlich darstellen:





3.6.6. Induktive Zyklen
mit variabler Anzahl von Wiederholungen durch Verwendung eines Endekennzeichens
(Fall (4) der Übersicht auf Seite 39)

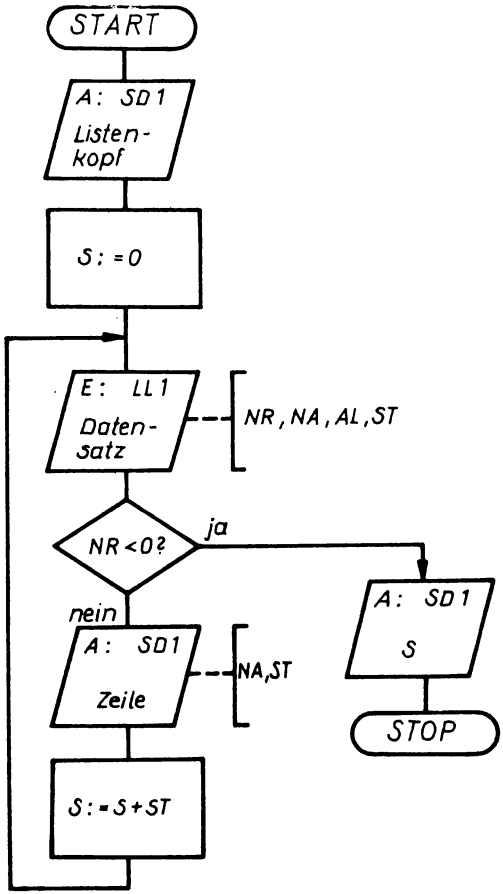
Auch diese Zyklusart gestattet die Verarbeitung von Zahlenmengen unterschiedlichen Umfanges. Dabei wird die zu verarbeitende Zahlenmenge mit einem Endekennzeichen abgeschlossen. Als solches verwendet man eine Zahl, die außerhalb des Definitionsbereiches der Zahlen der zu verarbeitenden Zahlenmenge liegt. Besteht z. B. die zu verarbeitende Zahlenmenge aus positiven Zahlen, verwendet man als Endekennzeichen eine negative Zahl.
Das soll an einem Beispiel gezeigt werden.
Für eine Studentenstatistik stehen auf einem Datenband folgende Angaben zur Verarbeitung bereit

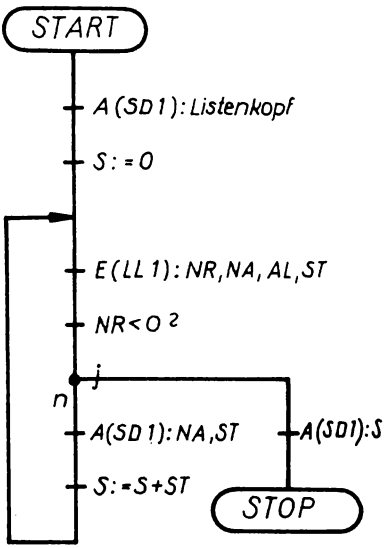


Es bedeuten:

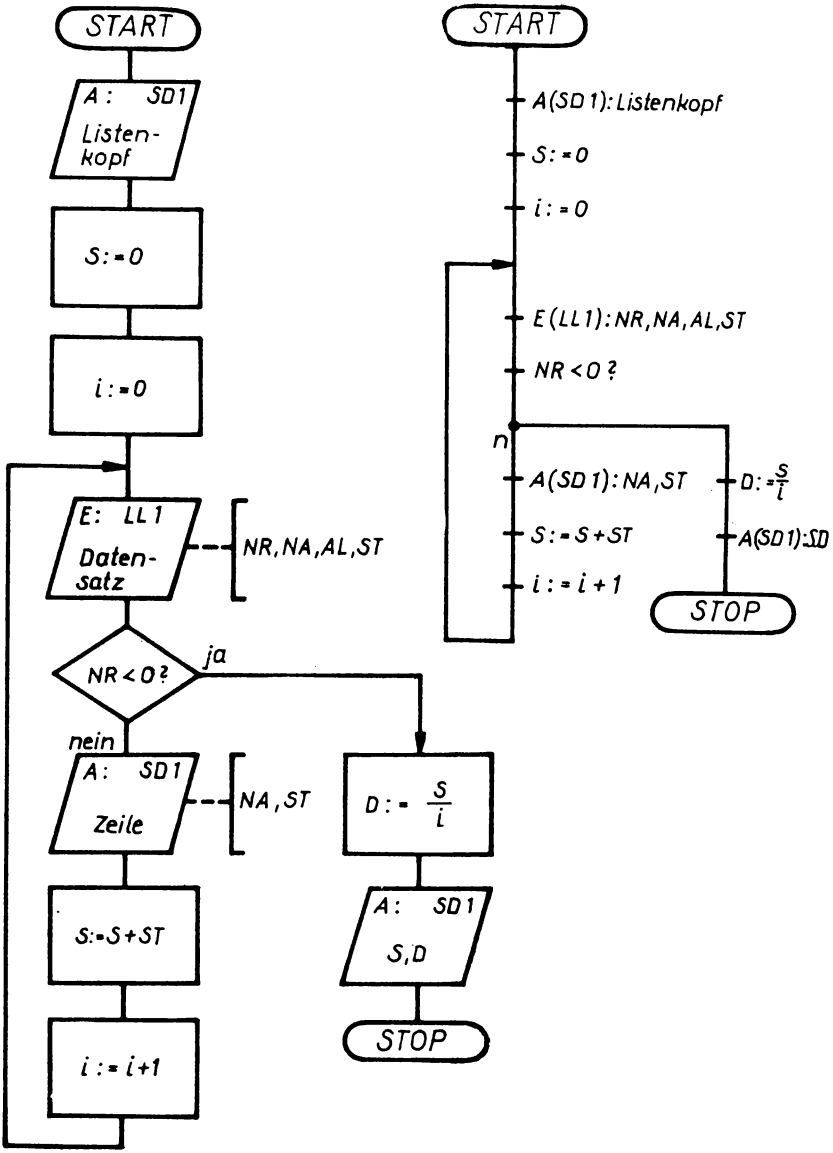
NR Studentennummer
NA Name des Studenten
AL Alter des Studenten
ST Höhe des Stipendiums
| Datensatzmarke

Am Ende der Datenfolge steht ein Pseudodatensatz, der an der Stelle der Studentennummer eine negative Zahl enthält. Es ist eine Liste zu drucken, bei der in jeder Zeile der Name und die Höhe des Stipendiums steht. Zum Abschluß dieser Liste ist die Gesamtsumme des Stipendiums zu drucken.





Sollte zur Ermittlung bestimmter Größen auch die Anzahl der Objekte wichtig sein, ist zusätzlich wieder ein Zähler einzubauen. Um das zu zeigen, wird das eben behandelte Beispiel dahingehend erweitert, daß zusätzlich noch das durchschnittliche Stipendium pro Student berechnet und gedruckt werden soll. Dann erhält man folgenden PAP:


START
A: SD1
Listen-
kopf
S:=0
i:=0
E: LL1
Daten-
satz
NR,NA,AL,ST
NR<0?
ja
nein
A: SD1
Zeile
NA,ST
S:=S+ST
i:=i+1
D:= S/i
A: SD1
S,D
STOP
START
A(SD1):Listenkopf
S:=0
i:=0
E(LL1):NR,NA,AL,ST
NR<0?
n
A(SD1):NA,ST
S:=S+ST
i:=i+1
D:=S/i
A(SD1):SD
STOP

3.6.7. Iterative Zyklen mit speziellen Näherungsformeln (Fall (5) der Übersicht auf Seite 39)

Die Ermittlung der Funktionswerte irrationaler Funktionen kann in EDVA nur dann erfolgen, wenn ein Algorithmus zur näherungsweisen Berechnung vorliegt, der die Berechnung der gesuchten Größe auf die Grundrechenarten zurückführt. Dabei ist die Anzahl der Wiederholungen des Zyklusbereiches nicht bekannt.
Der Abbruch eines solchen Zyklus wird deshalb von der geforderten Genauigkeit der zu berechnenden Größe abhängig gemacht.
Der Zyklusdurchlauf wird abgebrochen, wenn die gesuchte Größe mit hinreichender Genauigkeit ermittelt wurde. Das ist der Fall, wenn der Betrag der Differenz zweier aufeinander folgender Näherungswerte kleiner wird als eine vorgegebene Genauigkeit ϵ. Diese Größe ϵ gibt durch eine Zehnerpotenz mit negativem Exponenten an, in welcher Stelle nach dem Komma keine Veränderung mehr auftreten soll.

Dazu ein Beispiel:

Berechnung der Funktion $y = \sqrt{x}$ nach der **Iterationsformel**

$$y_{n+1} = 0{,}5 \left(\frac{x}{y_n} + y_n\right) \text{ für } x > 0 \qquad n = 1, 2, \ldots$$

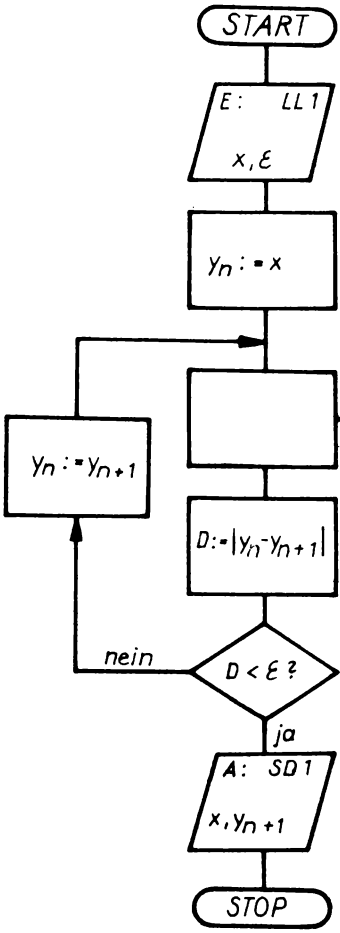
Die Werte für den Radikanden x und die Genauigkeit ϵ sollen über Lochband eingegeben werden. Radikand und Ergebnis sind zu drucken. Als Anfangswert wird mit $y_0 = x$ gerechnet.

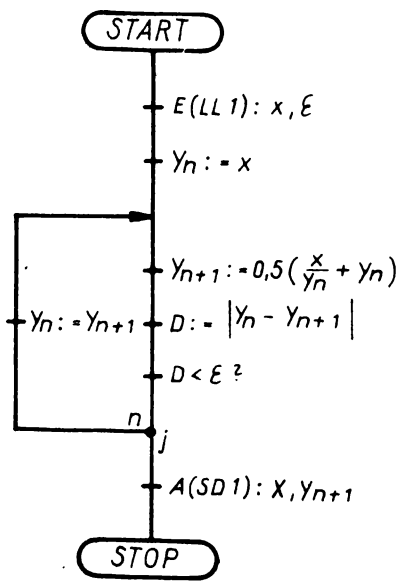
Die Berechnung vollzieht sich nach folgendem Algorithmus:

1. Wähle den ersten Näherungswert für y_n (Anfangswert $y_o = x$)
2. Berechne den verbesserten Näherungswert y_{n+1} nach der gegebenen Formel
3. Berechne den Betrag der Differenz der beiden Näherungswerte
4. Prüfe, ob dieser Wert schon kleiner ist, als die vorgegebene Genauigkeit ϵ

- **wenn ja,** ist die geforderte Genauigkeit erreicht und das Ergebnis wird gedruckt.
- **wenn nein,** ist der berechnete Näherungswert als Eingangsgröße für die erneute Verwendung der Näherungsformel zu benutzen und danach mit 2. fortzusetzen.

Damit erhält man folgenden PAP:


START
E: LL 1
x, ε
yn := x
yn+1 := 0,5 · (x/yn + yn)
yn := yn+1
D := |yn - yn+1|
nein
D < ε ?
ja
A: SD 1
x, yn+1
STOP


START
E(LL 1): x, ε
yn := x
yn+1 := 0,5 (x/yn + yn)
yn := yn+1
D := |yn - yn+1|
D < ε ?
n
j
A(SD 1): X, yn+1
STOP

Trockentest für x = 5 und $\epsilon = 10^{-4}$:

y_n	y_{n+1}	D	Test
5,00000	3,00000	2,00000	nein
3,00000	2,33333	0,66667	nein
2,33333	2,23810	0,09523	nein
2,23810	2,23607	0,00203	nein
2,23607	2,23607	0,00000	ja
	======		

3.6.8. Iterative Zyklen mit Potenzreihen (Fall (6) der Übersicht auf Seite 39)

Eine zweite Variante der näherungsweisen Berechnung von Funktionswerten irrationaler Funktionen ist die mittels Reihenentwicklung.
Auch das soll an einem Beispiel gezeigt werden:

Programmieren Sie die Berechnung eines Funktionswertes der Exponentialfunktion $y = e^x$ bei einer vorgegebenen Genauigkeit ϵ.
Die Zahlenwerte für x und ϵ werden eingegeben, x und y sind zu drucken.

Mathematisches Modell:

– gegebene Formel

$$y = e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!} \qquad (|x| \leq 1)$$

– Da 0! nicht berechnet werden kann, bezieht man das 1. Glied der Reihe nicht in die Betrachtung ein und erhält:

$$y = e^x = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots = 1 + \sum_{i=1}^{\infty} \frac{x^i}{i!}$$

– Die Anfangswertbelegung der Summenzelle s im Block der Anfangsmaßnahmen erfolgt mit dem Wert 1

– Wie aus der Reihe ersichtlich ist, wird jedes Nachfolgeglied kleiner als sein Vorgänger. Es werden deshalb so viele Glieder zu berechnen und zu summieren sein, bis alle nachfolgenden Glieder im Rahmen der geforderten Genauigkeit keinen spürbaren Einfluß mehr auf das Endergebnis haben.

– Die unabhängige Berechnung jedes Reihengliedes nach der Formel

$$G_i = \frac{x^i}{i!}$$

wäre sehr umständlich, da nur über weitere Zyklen realisierbar. Es ist einfacher, jedes Nachfolgeglied G_i aus dem Vorgängerglied G_{i-1} durch eine Rekursionsformel zu berechnen:

$$G_i = f(G_{i-1})$$

Für die gegebene Reihe gilt:

$$G_i = \frac{x^i}{i!} = \frac{x^i}{1 \cdot 2 \cdot 3 \ldots (i-2) \cdot (i-1) \cdot i}$$

und

$$G_{i-1} = \frac{x^{i-1}}{(i-1)!} = \frac{xi-1}{1 \cdot 2 \cdot 3 \ldots (i-2) \cdot (i-1)}$$

Daraus erhält man den Quotienten

$$\frac{G_i}{G_{i-1}} = \frac{x^i \cdot 1 \cdot 2 \cdot 3 \ldots (i-2) \cdot (i-1)}{x^{i-1} \cdot 1 \cdot 2 \cdot 3 \ldots (i-2) \cdot (i-1) \cdot i} = \frac{x}{i}$$

und die gesuchte Rekusionsformel

$$G_i = G_{i-1} \cdot \frac{x}{i} \qquad \text{für } i = 1, 2, 3 \ldots$$

Es ist sicher zu erkennen, daß für i= 1 der Anfangswert $G_o = 1$ gesetzt werden muß.

Bild siehe Seite 62

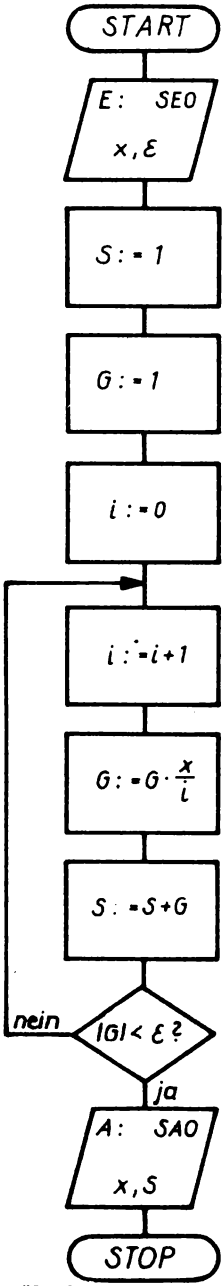
3.7. Übungen zu zyklischen PAP

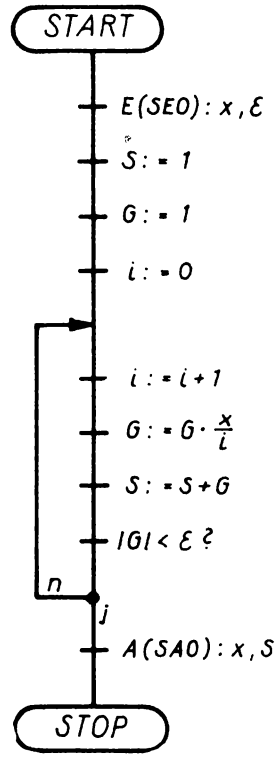
(6) Ein regelmäßiges Vieleck hat seinen Mittelpunkt im Koordinatenursprung und ist symmetrisch zur x-Achse, d. h. ein Eckpunkt liegt auf dieser Achse im Abstand a vom Ursprung.
Programmieren Sie die Ermittlung und den Druck der Koordinaten aller n Eckpunkte!
Die Zahlenwerte für n und a werden über Lochband eingelesen.

(7) Gegeben ist ein Lochband mit folgenden Angaben als Datensatz pro Student:
- Studenten-Nr.
- Name des Studenten
- Alter des Studenten
- Personenstandskennzeichen
  (0 ≙ ledig, 1 ≙ verheiratet, 2 ≙ geschieden)
- Anzahl der Kinder
- Höhe des Stipendiums

Das Datenband ist mit einem Pseudosatz abgeschlossen, der anstelle der Studenten-Nr. eine negative Zahl enthält.
Programmieren Sie die Berechnung der Summe des Stipendiums und den Druck einer Auszahlungsliste.

(8) Gegeben ist die Steigung s einer Fahrbahn in Prozent
(Eingabe über Bedienkonsole).
Programmieren Sie die Berechnung des Steigungswinkels der Fahrbahn in Grad und Minuten
(Ausgabe über Bedienkonsole)!





Überprüfen Sie diesen PAP mit einem Trockentest für x = 1 und $\epsilon = 10^{-5}$!

Für die Berechnung kleiner Winkel im Bogenmaß gilt die Reihe:

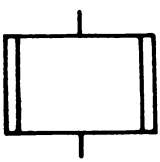
$$y = \arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} \quad - + \ldots$$

Die Genauigkeit soll 5 Stellen nach dem Komma betragen.

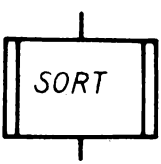
4. Unterprogramme

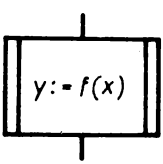
Unter einem Unterprogramm versteht man eine Folge von mehreren Operationen, die an anderer Stelle ausführlich behandelt, im vorliegenden PAP jedoch als Einheit betrachtet werden soll.
Dazu gehören z. B. mathematische Standardlösungen oder ständig wiederkehrende Befehlsfolgen.

– Symbol



Der Name des Programmteils wird bei der Kästchenmethode in das Symbol hinein, bei der Programmlinienmethode rechts neben das Symbol geschrieben, z. B.:





In den Programmiersprachen unterscheidet man dabei grundsätzlich zwei Arten von Unterprogrammen

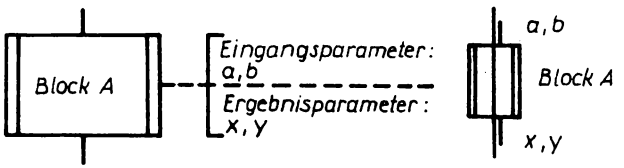
– Funktionsprogramme
– Subroutinen

In beiden Fällen werden dem Unterprogramm vom Hauptprogramm die Eingangsparameter übergeben. Funktionsprogramme berechnen daraus nur **einen** Funktionswert und geben diesen an das Hauptprogramm zurück. Mit Subroutinen können **mehrere** Werte ermittelt werden, die als Rückgabeparameter dem Hauptprogramm übergeben werden.

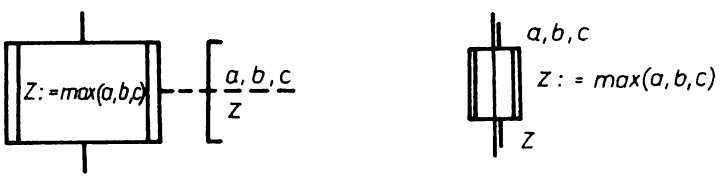
Die zwischen Hauptprogramm und Unterprogramm zu übergebenden Parameter sind als „Bemerkung" einzutragen.

Beispiele

(1) Unterprogramm zur Berechnung einer Befehlsfolge A



(2) Unterprogramm zur Bestimmung des Maximums der Werte von a, b und c



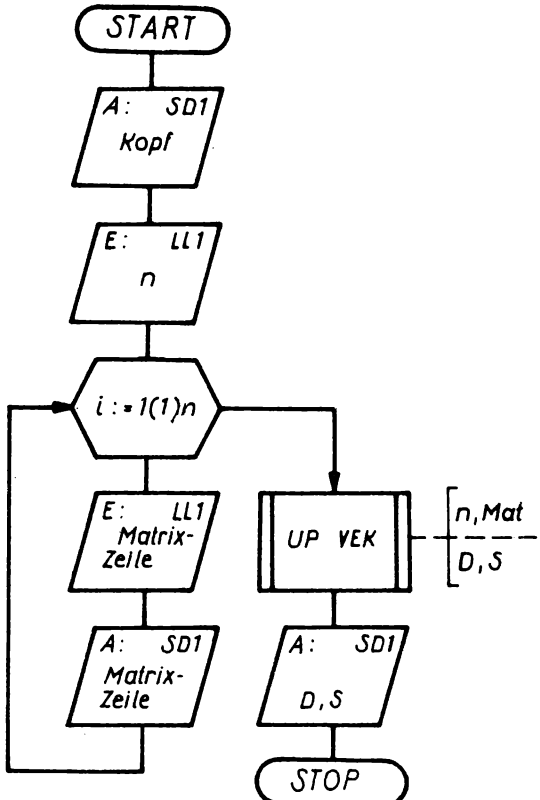
Für einfache, in arithmetischen Ausdrücken verwendete mathematische Grundfunktionen wie sin(x), log(x), arctan(x) u. a. wird dieses Symbol nicht benutzt.
Diese werden als „Standardfunktionen" vom Hersteller der EDVA mitgeliefert und können auch bei der Programmierung in arithmetischen Ausdrücken (rechte Seite einer Ergibtanweisung) geschrieben werden.

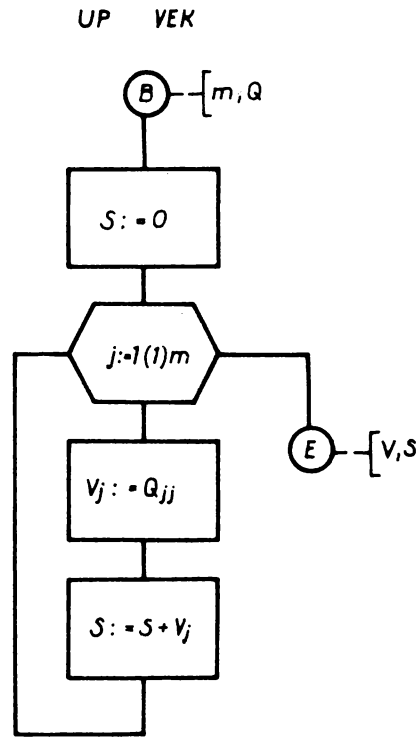
Die Operationsfolge des Unterprogrammes selbst muß an anderer Stelle der Dokumentation dargestellt sein. Beginn und Ende dieses UP werden durch Konnektoren mit den Buchstaben Ⓑ und Ⓔ gekennzeichnet (Beginn und Ende).

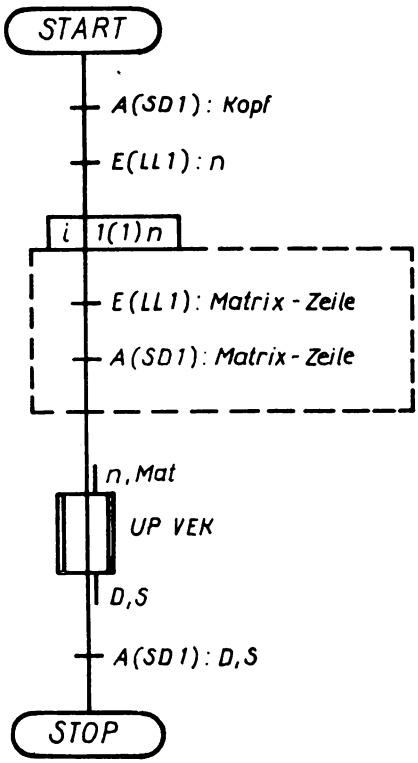
Dazu ein Beispiel:

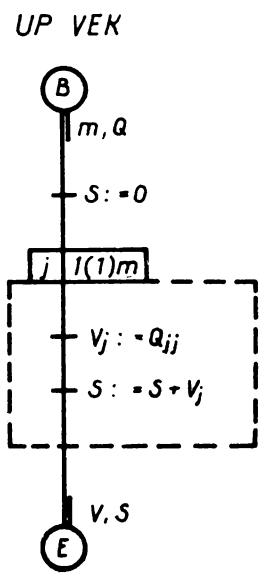
Von einer quadratischen Matrix sind mit einem Unterprogramm der Vektor der Hauptdiagonalen zu bilden und die Summe der Elemente zu berechnen. Auf dem Datenband sind die Anzahl der Zeilen n und danach zeilenweise die Elemente der Matrix abgelocht.
Die Matrix, der Vektor der Hauptdiagonalen und die Summe der Diagonalelemente sind zu drucken.

(Kästchenmethode)






START
A(SD1): Kopf
E(LL1): n
i 1(1)n
E(LL1): Matrix - Zeile
A(SD1): Matrix - Zeile
n, Mat
UP VEK
D, S
A(SD1): D, S
STOP


UP VEK
B
m, Q
S: = 0
j 1(1)m
Vj: = Qjj
S: = S + Vj
V, S
E

5. Anwendungen

Die Praxis der Programmerarbeitung zeigt, daß die Aufgabenstellungen so gestaltet sind, daß die entstehenden Programmablaufpläne komplexer und komplizierter sind als die in den vorangegangenen Kapiteln behandelten Grundstrukturen; sie stellen Kombinationen dieser dar. Die Mannigfaltigkeit der dabei auftretenden Kombinationen läßt sich in einem Lehrbriefkapitel selbstverständlich nicht allumfassend darstellen.
An Hand von Beispielen sollen deshalb einige ständig wiederkehrende Standardprobleme behandelt werden.

(1) Auswertung von Personenstammdaten

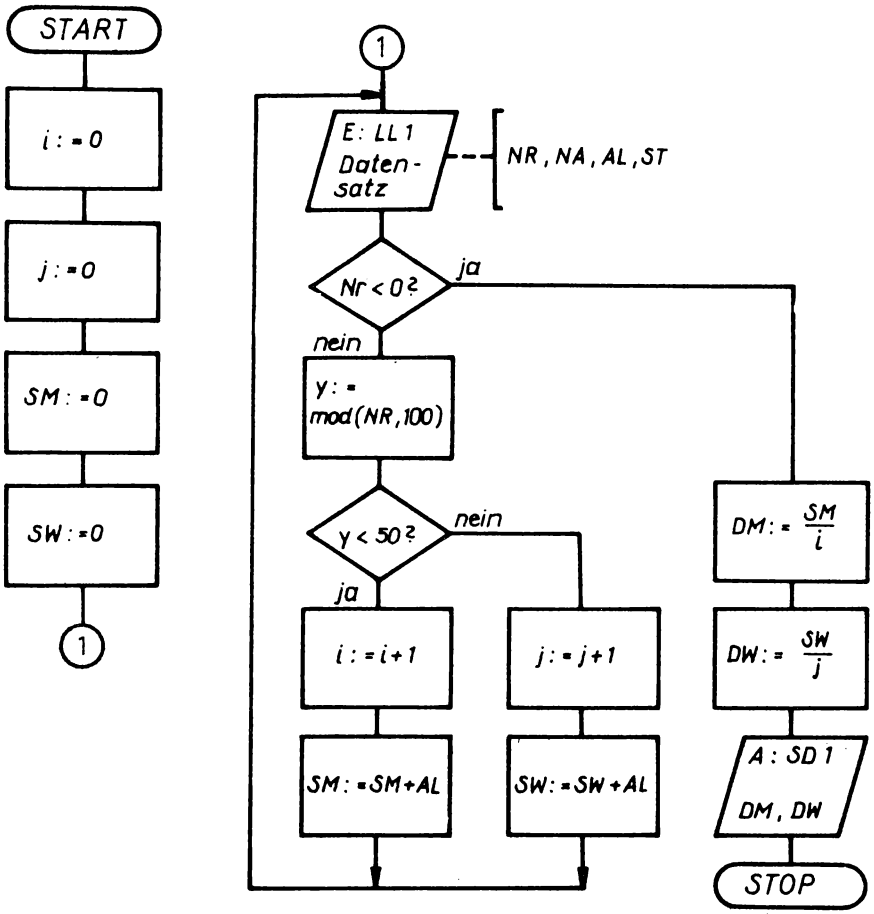
Ausgehend von dem unter 4.5.5. beschriebenen Datenband einer Studentenstatistik soll getrennt für weibliche und männliche Studenten das Durchschnittsalter ermittelt und gedruckt werden.
Die beiden letzten Ziffern der Studentennummer geben einen Hinweis auf das Geschlecht der Studenten. Angaben < 50 bedeuten männlich, Angaben ≥ 50 bedeuten weiblich.
Zur Abspaltung der letzten beiden Ziffern benutzt man die Modulus-Funktion, die den Rest einer ganzzahligen Division ermittelt.
y = mod (a, b) bedeutet, daß y der Rest der ganzzahligen Division a : b ist.

Es werden folgende Symbole verwendet:

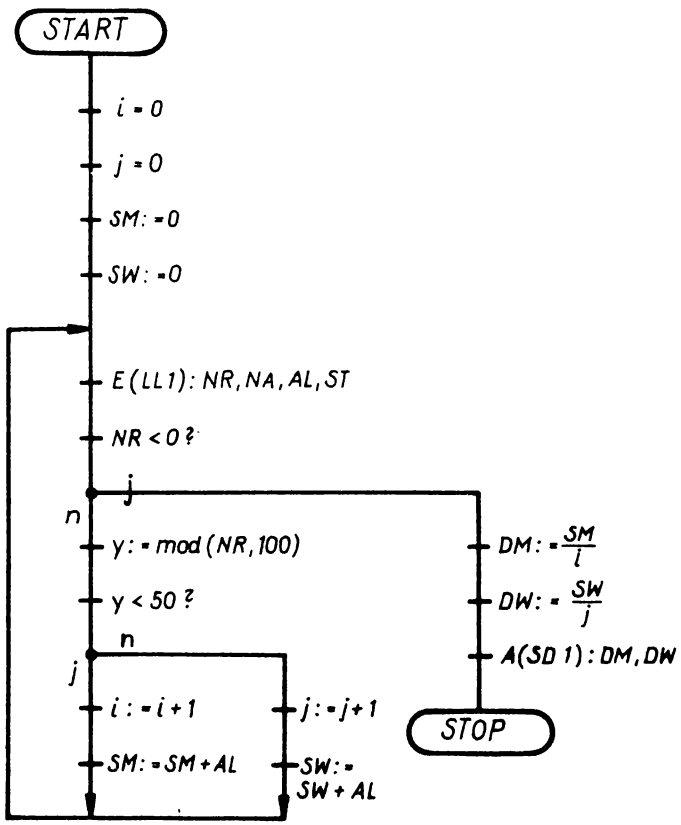
i	– Zähler für männliche Studenten
j	– Zähler für weibliche Studenten
SM	– Summe der Altersangaben männl. Studenten
SW	– Summe der Altersangaben weibl. Studenten
DM	– Durchschnittsalter der männl. Studenten
DW	– Durchschnittsalter der weibl. Studenten

Bilder siehe Seite 68 und 69

(Kästchenmethode)



(Programmlinienmethode)



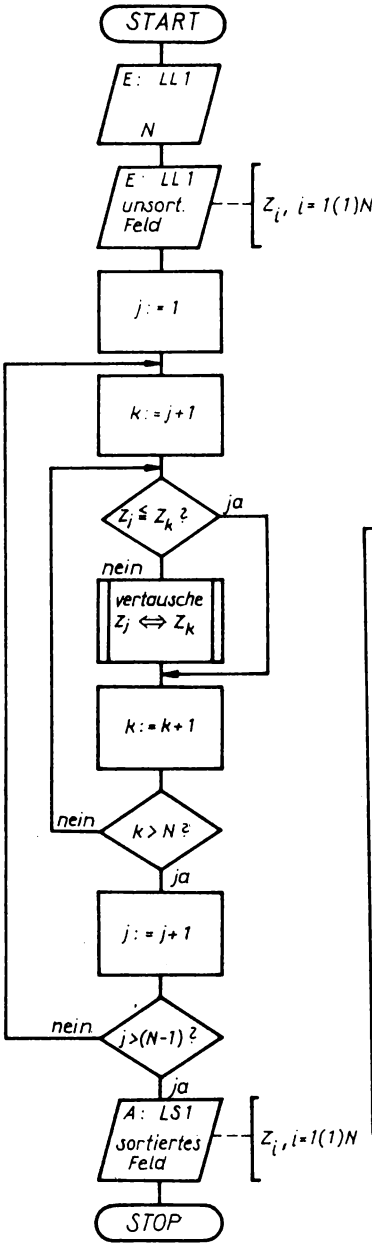
(2) Sortieren einer Datenmenge

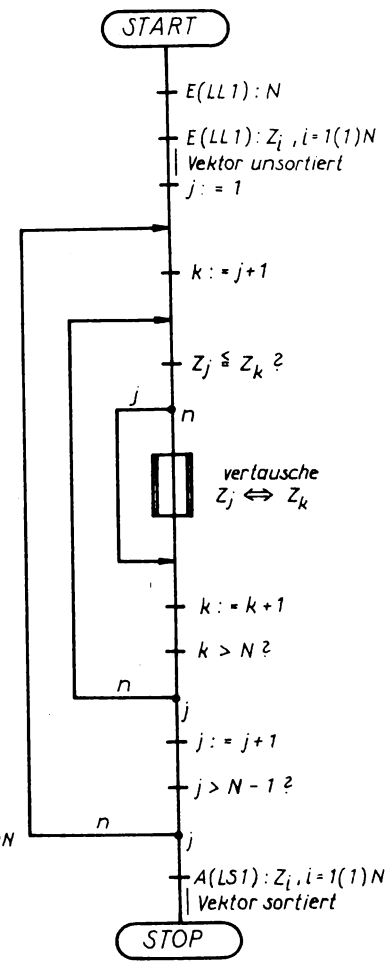
Auf einem Lochband stehen n Zahlen zur Verarbeitung bereit (der Wert für n befindet sich als erste Zahl auf dem Lochband).

Diese Zahlen sollen als Feld eingelesen, aufsteigend sortiert und wieder auf Lochband ausgegeben werden.

Derartige speicherplatzsparende Sortiervorgänge können häufig vorkommen. Sie lassen sich auch auf alphanumerische Daten anwenden.

Bild siehe Seite 70


START
E: LL1
N
E: LL1
unsort. Feld
Z_i, i = 1(1)N
j := 1
k := j+1
$Z_j \leqq Z_k$?
ja
nein
vertausche $Z_j \Leftrightarrow Z_k$
k := k+1
nein
k > N ?
ja
j := j+1
nein
j>(N-1) ?
ja
A: LS1
sortiertes Feld
Z_i, i=1(1)N
STOP

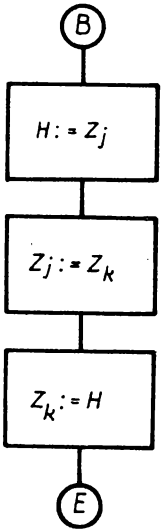

START
E(LL1) : N
E(LL1) : Z_i , i = 1(1)N
Vektor unsortiert
j := 1
k := j+1
$Z_j \leqq Z_k$?
j
n
vertausche $Z_j \Leftrightarrow Z_k$
k := k+1
k > N ?
n
j
j := j+1
j > N - 1 ?
n
j
A(LS1) : Z_i , i = 1(1)N
Vektor sortiert
STOP

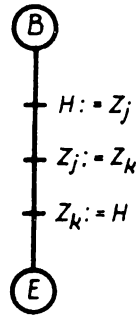
Die Wirksamkeit soll an einem Trockentest nachgewiesen werden. Für N = 6 sollen folgende Zahlen gegeben sein:

j	k	z_1	z_2	z_3	z_4	z_5	z_6
umsortiert:		20	18	24	24	15	25
1	2 3 4 5 6 7	18 15	20			 18	
2	3 4 5 6 7		 18			 20	
3	4 5 6 7			 20		 24	
4	5 6 7						
5	6 7						
sortiert		15	18	20	24	24	25

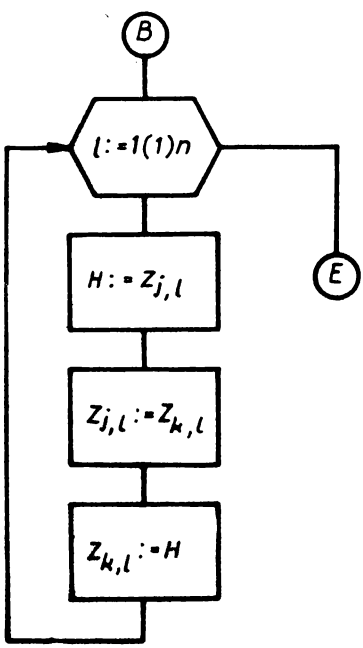
Das als Unterprogramm gekennzeichnete „Vertauschen" zweier Zahlen bedarf einer besonderen Untersuchung.

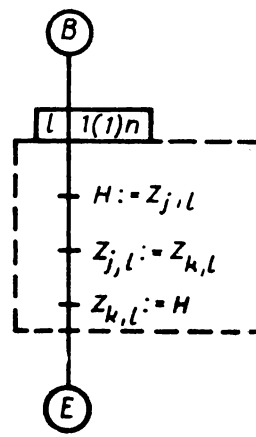
Beim Vertauschen zweier Zahlenwerte ist zu beachten, daß dazu ein Hilfsspeicherplatz gebraucht wird, auf den der Wert des einen Speicherplatzes ausgelagert werden muß. Im einzelnen ist die Vertauschung in drei Schritten realisierbar:





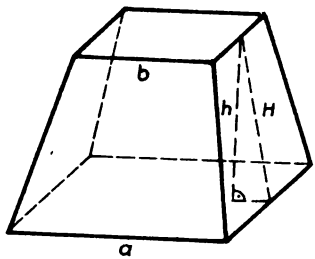
Sollen ganze Matrix-Zeilen (z. B. Zeichenketten) vertauscht werden, ist zu beachten, daß **alle** Elemente der Zeile (Kette) vertauscht werden müssen. D. h., die Vertauschung läuft innerhalb eines Zyklus ab. Für n Elemente müßte man schreiben:





6. Lösungen der Übungen

(1) **Skizze:**



Mathematisches Modell:

Grundfläche: $A_G = a^2$

Deckfläche: $A_D = b^2$

Parallelflächen: $A_p = A_G + A_D$

Trapezhöhe: $H = \sqrt{h^2 + (\frac{a-b}{2})^2}$

Trapezfläche: $A_T = \frac{a+b}{2} \cdot H$

Mantelfläche: $A_M = 4 \cdot A_T$

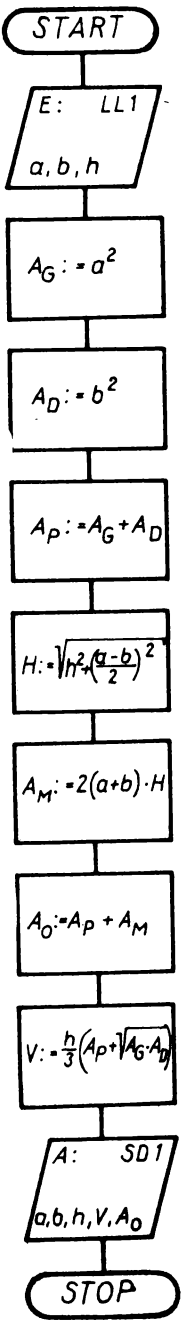
Oberfläche: $A_o = A_p + A_M$

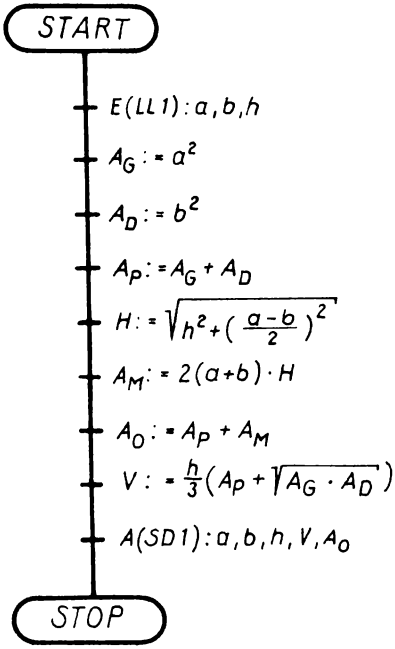
Volumen: $V = \frac{h}{3} (A_p + \sqrt{A_G \cdot A_D})$

Organisation der Ein- und Ausgabe:

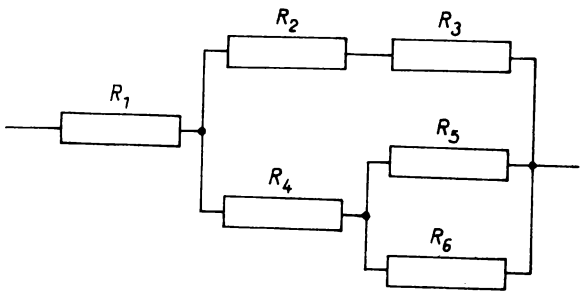
- Die Größen a, b und h sollen in dieser Reihenfolge über LL1 eingegeben werden.
- Zur Kontrolle sollen diese gegebenen Werte protokolliert, außerdem sollen die berechneten Werte über den SD1 gedruckt werden.

PAP:


START
E: LL1
a, b, h
$A_G := a^2$
$A_D := b^2$
$A_P := A_G + A_D$
$H := \sqrt{h^2 + \left(\frac{a-b}{2}\right)^2}$
$A_M := 2(a+b) \cdot H$
$A_O := A_P + A_M$
$V := \frac{h}{3}\left(A_P + \sqrt{A_G \cdot A_D}\right)$
A: SD1
a, b, h, V, A_O
STOP


START
E(LL1): a, b, h
$A_G := a^2$
$A_D := b^2$
$A_P := A_G + A_D$
$H := \sqrt{h^2 + \left(\frac{a-b}{2}\right)^2}$
$A_M := 2(a+b) \cdot H$
$A_O := A_P + A_M$
$V := \frac{h}{3}\left(A_P + \sqrt{A_G \cdot A_D}\right)$
A(SD1): a, b, h, V, A_O
STOP

(2) Skizze:



Mathematisches Modell:

$$R_A = R_5 \parallel R_6 = \frac{R_5 \cdot R_6}{R_5 + R_6}$$

$$R_B = R_2 + R_3$$

$$R_C = R_4 + R_A$$

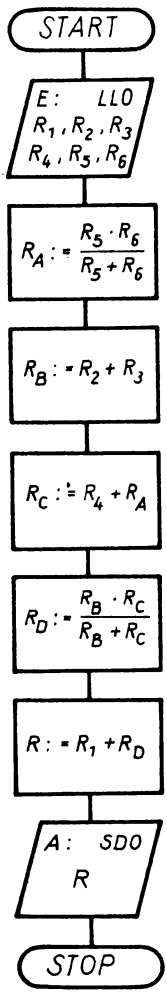
$$R_D = R_B \parallel R_C = \frac{R_B \cdot R_C}{R_B + R_C}$$

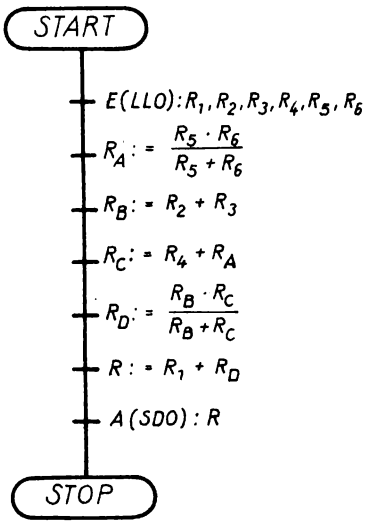
$$R = R_1 + R_D$$

Organisation der Ein- und Ausgabe:

- Die gegebenen Widerstandswerte stehen auf Lochband bereit und werden über LLO eingelesen.
- Der ermittelte Wert des Gesamtwiderstandes ist über den SDO zu drucken.

PAP:


START
E: LLO
R1, R2, R3
R4, R5, R6
RA := (R5 · R6) / (R5 + R6)
RB := R2 + R3
RC := R4 + RA
RD := (RB · RC) / (RB + RC)
R := R1 + RD
A: SDO
R
STOP


START
E(LLO): R1, R2, R3, R4, R5, R6
RA := (R5 · R6) / (R5 + R6)
RB := R2 + R3
RC := R4 + RA
RD := (RB · RC) / (RB + RC)
R := R1 + RD
A (SDO) : R
STOP

(3) **Mathematisches Modell:**

Zeit des freien Falls: x_1

Zeit des Schalles: x_2

gemessene Gesamtzeit: $x = x_1 + x_2 \qquad (*)$

Weg beim freien Fall: $s = \frac{g}{2} x_1^2$

daraus: $x_1 = \sqrt{\frac{2s}{g}}$

Weg des Schalles: $s = v \cdot x^2$

daraus: $x_2 = \frac{s}{v}$

Damit erhält man aus (*)

$$x = \sqrt{\frac{2s}{g}} + \frac{s}{v}$$

Durch Umformung nach s (quadratische Gleichung) und Berücksichtigung, daß nur die Lösung mit positivem Ergebnis möglich ist, erhält man

$$s = \frac{v^2}{2g} \left(\sqrt{1 + \frac{2gx}{v}} - 1\right)^2 .$$

Mit $y = \frac{v}{2g}$ wird dann:

$$s = v \cdot y \cdot \left(\sqrt{1 + \frac{x}{y}} - 1\right)^2$$

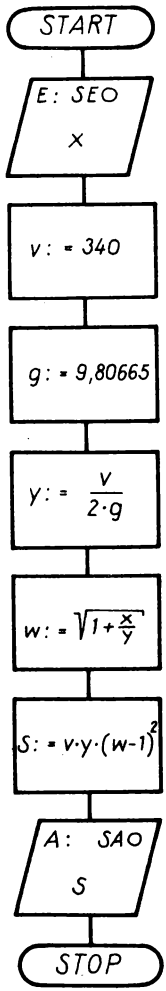
mit den Konstanten $v = 340\ ms^{-1}$

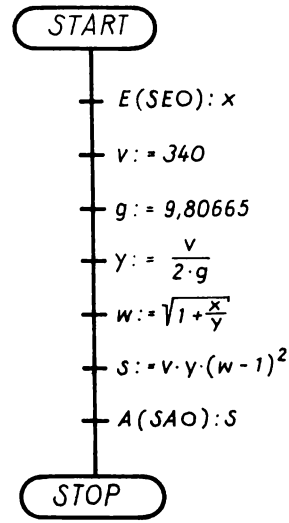
und $g = 9{,}80665\ ms^{-2}$

Organisation der Ein- und Ausgabe:

Der gegebene Wert x und der ermittelte Wert s sind über die Bedienschreibmaschine ein- und auszugeben.

PAP:


START
E: SEO
x
v: = 340
g: = 9,80665
y: = v / 2·g
w: = √(1 + x/y)
S: = v·y·(w-1)²
A: SAO
S
STOP


START
E(SEO): x
v: = 340
g: = 9,80665
y: = v / 2·g
w: = √(1 + x/y)
S: = v·y·(w - 1)²
A(SAO): S
STOP

(4) **Mathematisches Modell:**

Die Funktion $y = f(a, b) = \frac{a}{b \cdot \sqrt{a+b}}$ ist dann nicht definiert, wenn

- entweder der Nenner null wird:

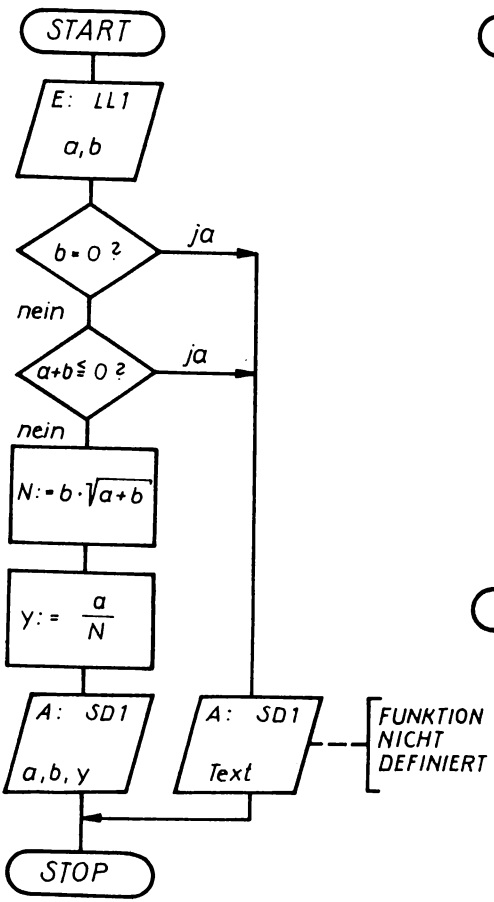
  $b \cdot \sqrt{a+b} = 0$

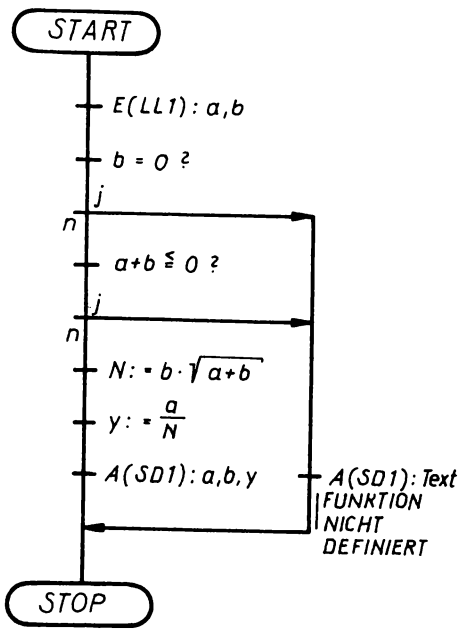
  daraus folgt: $b = 0$ oder $\sqrt{a+b} = 0$, d. h. $a + b = 0$

- oder der Radikand negativ wird:

  daraus folgt: $a + b < 0$

PAP:





(5) **Mathematisches Modell:**

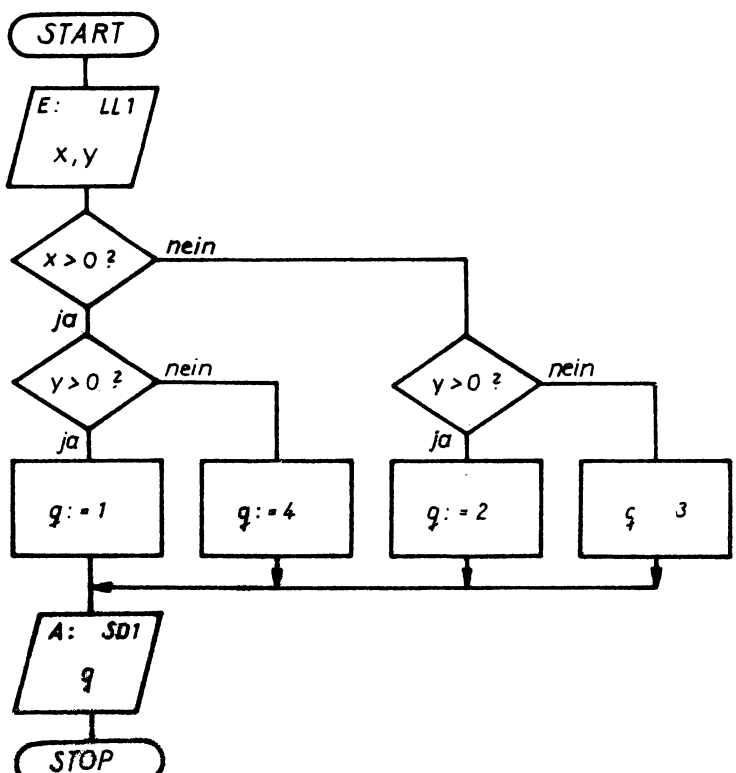
Vorzeichenfestlegung der Quadranten:

Quadrant / Koordinate	I	II	III	IV
x	+	–	–	+
y	+	+	–	–

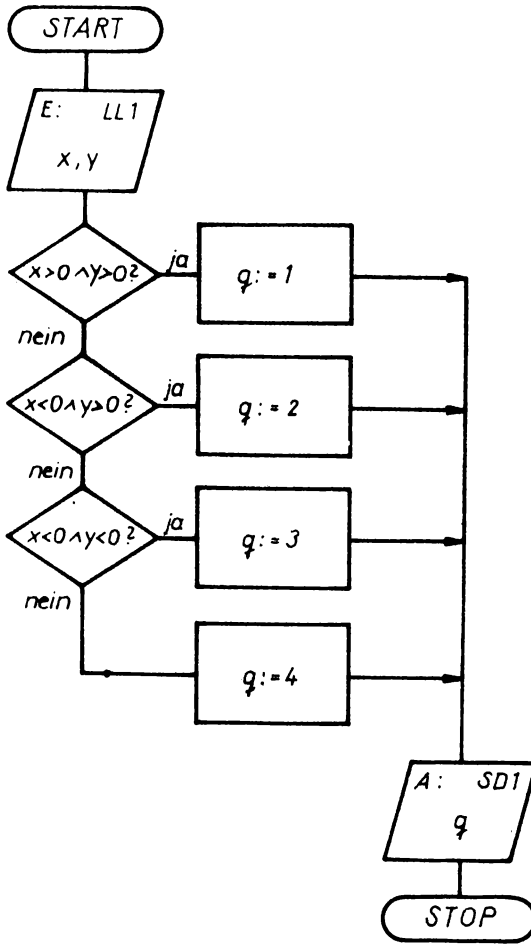
PAP

(Kästchenmethode)

1. Variante:

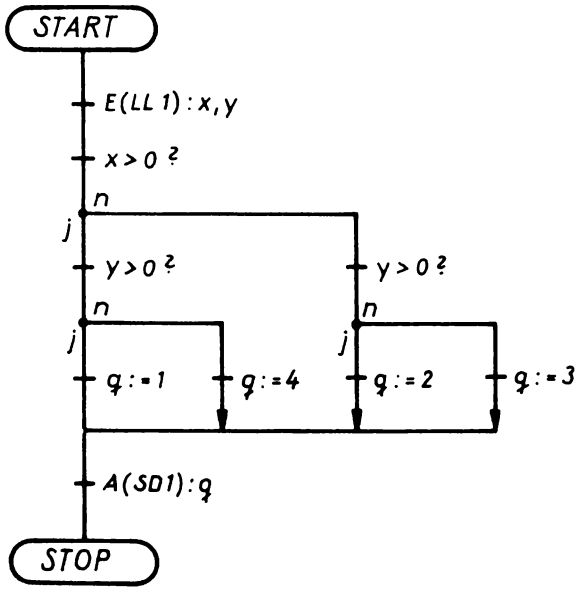


2. Variante:

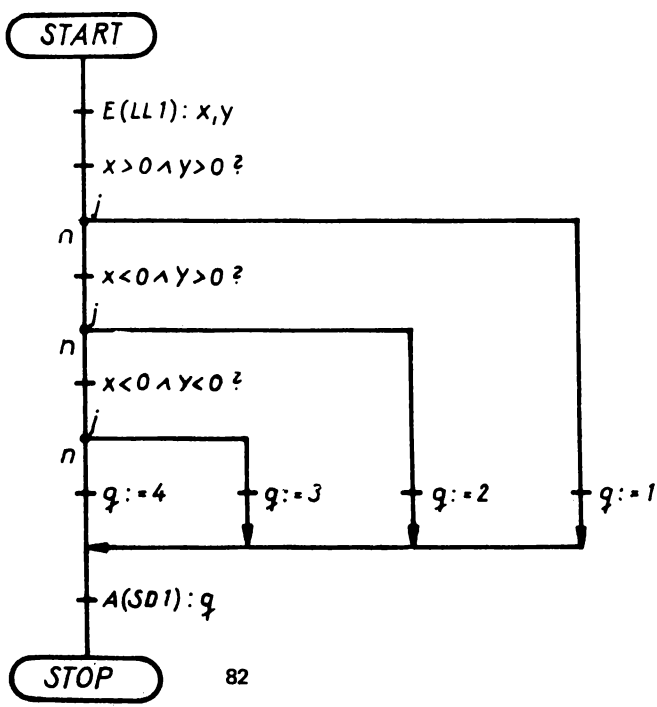

START
E: LL1
x, y
x>0 ∧ y>0 ?
ja
q := 1
nein
x<0 ∧ y>0 ?
ja
q := 2
nein
x<0 ∧ y<0 ?
ja
q := 3
nein
q := 4
A: SD1
q
STOP

(Programmlinienmethode)

1. Variante:



2. Variante:



(6) **Mathematisches Modell:**

1. Der Zentriwinkel des Vielecks berechnet sich nach

$$\alpha = \frac{360^\circ}{n}$$

2. Für den laufenden Winkel $\varphi = \varphi + a$ berechnen sich die Koordinaten der n Punkte nach den Beziehungen $x = a \cdot \sin \varphi$ und $y = a \cdot \cos \varphi$

Es handelt sich also nach der Einteilung im Kapitel 4.5. um einen induktiven Zyklus mit indirekter Vorgabe der Anzahl der Durchläufe durch eine Variable (Kapitel 4.5.3.).

Bild siehe Seite 84

(7) **Mathematisches Modell**

- Festlegung der Variablenbezeichnungen

Studenten Nummer	NR
Name des Studenten	NA
Stipendium	ST
Summe	S

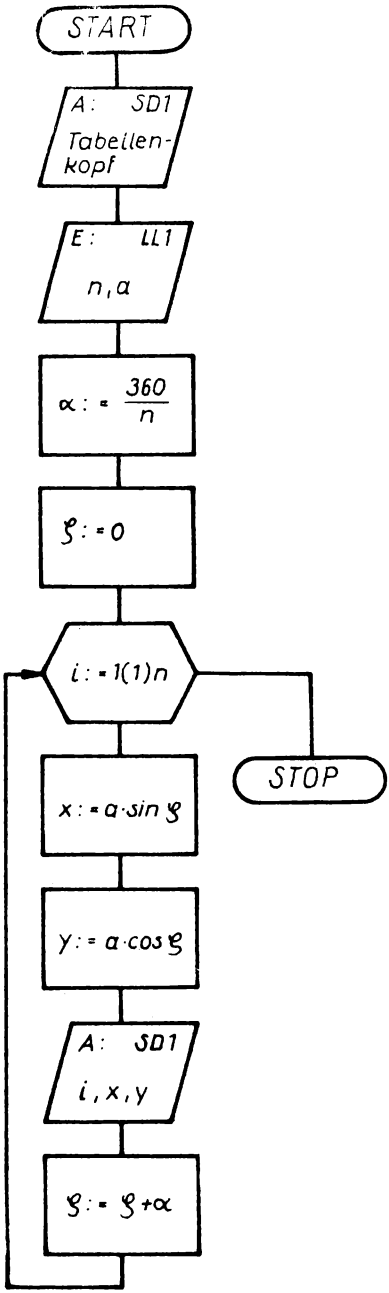
- Es handelt sich nach der Einteilung im Kapitel 4.5. um einen induktiven Zyklus mit indirekter Vorgabe der Anzahl der Durchläufe durch ein Endekennzeichen (Kapitel 4.5.5.).

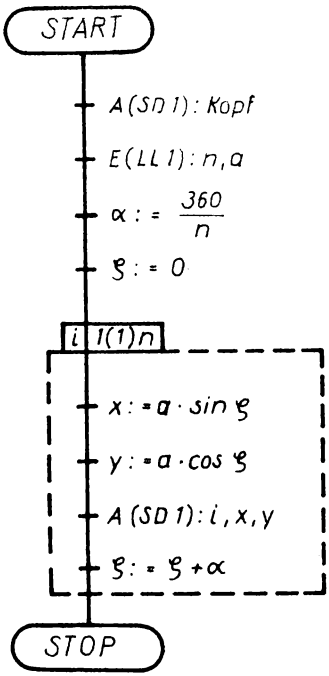
Bild siehe Seite 85

(8) **Mathematisches Modell:**

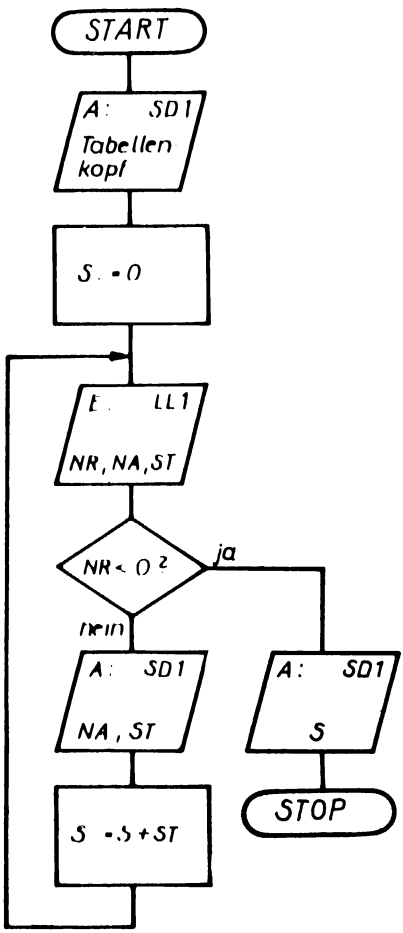
- Durch Division durch 100 erhält man aus der in Prozent angegebenen Steigung den Tangens des Winkels
- Durch Iteration mit Hilfe der gegebenen Formel erhält man den Winkel im Bogenmaß
- Nach Multiplikation mit $\frac{180}{\pi} = 57{,}3$ ergibt sich daraus der Winkel als Dezimalzahl
- Nach Multiplikation mit 60 erhält man daraus den Winkel in Minuten als Dezimalzahl
- Durch ganzzahlige Division durch 60 ergibt sich der Gradanteil des Winkels
- Durch Anwendung der mod-Funktion ergibt sich schließlich der Rest dieser ganzzahligen Division als Minutenanteil des Winkels.
- Es handelt sich um einen iterativen Zyklus gemäß Kapitel 4.5.6..

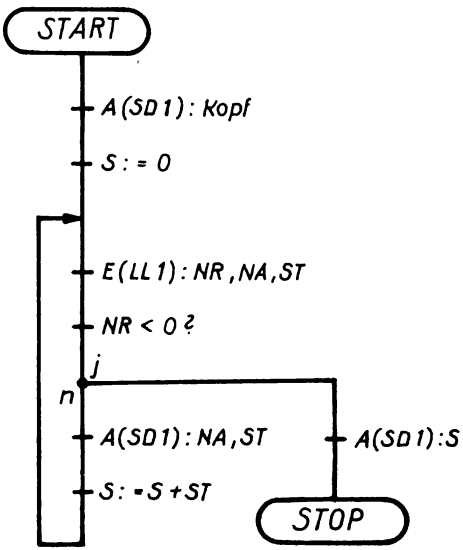
PAP


START
A: SD1
Tabellen-kopf
E: LL1
n, a
α := 360 / n
ϕ := 0
i := 1(1)n
STOP
x := a · sin ϕ
y := a · cos ϕ
A: SD1
i, x, y
ϕ := ϕ + α

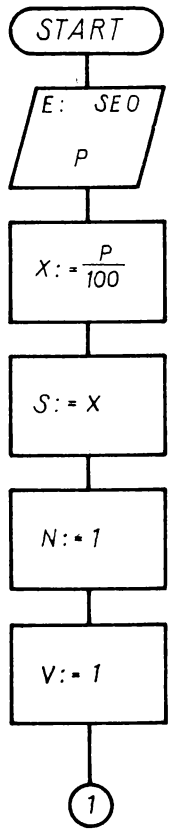

START
A(SD 1): Kopf
E(LL 1): n, a
α := 360 / n
ϕ := 0
i 1(1)n
x := a · sin ϕ
y := a · cos ϕ
A (SD 1): i, x, y
ϕ := ϕ + α
STOP

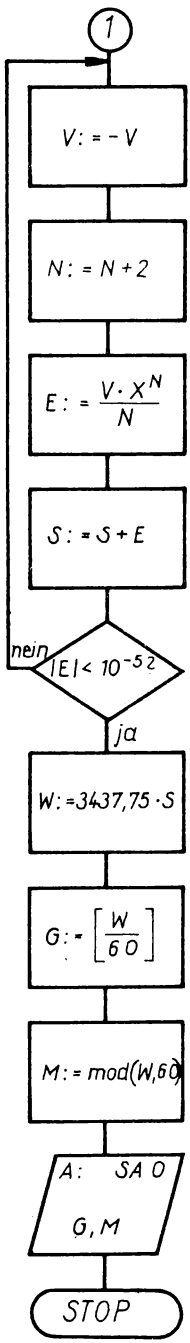
PAP:

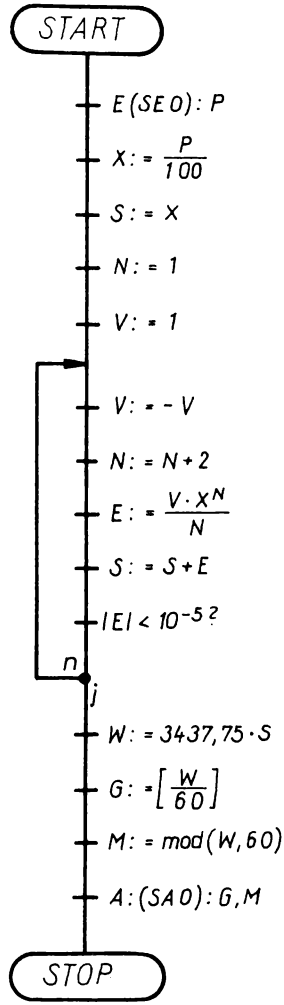

START
A: SD1
Tabellen kopf
S := 0
E: LL1
NR, NA, ST
NR < 0 ?
ja
nein
A: SD1
NA, ST
S := S + ST
A: SD1
S
STOP


START
A(SD1): Kopf
S := 0
E(LL1): NR, NA, ST
NR < 0 ?
j
n
A(SD1): NA, ST
S := S + ST
A(SD1): S
STOP

PAP:


START
E: SEO
P
X: = P/100
S: = X
N: = 1
V: = 1
1


1
V: = - V
N: = N + 2
E: = V · X^N / N
S: = S + E
nein
|E| < 10^-5 ?
ja
W: = 3437,75 · S
G: = [W/60]
M: = mod(W,60)
A: SA O
G, M
STOP


START
E(SEO): P
X: = P/100
S: = X
N: = 1
V: = 1
V: = - V
N: = N + 2
E: = V · X^N / N
S: = S + E
|E| < 10^-5 ?
n
j
W: = 3437,75 · S
G: = [W/60]
M: = mod(W,60)
A: (SAO): G, M
STOP

Literaturverzeichnis:

/1/ Rahmenmethodik der Datenverarbeitungsprojektierung
VEB Kombinat Robotron Dresden 1973

/2/ Aktualisierung und Ergänzung zur Rahmenmethodik der Datenverarbeitungsprojektierung vom März 1973
—Methodischer Leitfaden zur DV-Projektierung —
VEB Kombinat Robotron Dresden 1976

/3/ TGL 22 451
Informationsverarbeiung
Datenfluß- und Programmablaufpläne
Sinnbilder
Verbindlich ab 01. 01. 1976

/4/ Wörterbuch der Kybernetik
Herausgegeben von Georg Klaus und Heinz Liebscher
Dietz Verlag, Berlin 1976

/5/ Meyer
Neues Lexikon in 8 Bänden
VEB Bibliografisches Institut, Leipzig 1961

/6/ Kleine Enzyklopädie Mathematik
VEB Bibliografisches Institut, Leipzig 1967

/7/ EDV Aufgabensammlung (Lösungsteil)
von R. Schmidt
Lehrbrief 026.10 herausgegeben vom
HF der DDR, Karl-Marx-Stadt 1974

FACHSCHUL–
FERNSTUDIUM

Dipl.-Math. Arnd Haustein

GRUNDLAGEN DER INFORMATIONSVERARBEITUNG FÜR INGENIEURE

2/2

Übungen zur Programmablaufplanung

Herausgeber:
Institut für Fachschulwesen der
Deutschen Demokratischen Republik
Karl-Marx-Stadt

03 0160 02 2

Dieser Lehrbrief wurde
verfaßt von:

Dipl.-Math. Arnd Haustein
Ingenieurschule für Verkehrstechnik „Erwin Kramer"
Dresden

lektoriert von:

Dipl.-Ing. Klaus Paschmionka
Ingenieurschule für Maschinenbau Leipzig

bearbeitet von:

Dipl.-Phys. Werner Seifert
Institut für Fachschulwesen der DDR
Karl-Marx-Stadt

Redaktionsschluß: 15. 6. 1984

Druck und buchbinderische Verarbeitung:
Zentralstelle für Lehr- und Organisationsmittel des Ministeriums
für Hoch-und Fachschulwesen, Zwickau
1.Auflage 1984
2.unveränderter Nachdruck 1987
Ag 613/4020/87/5200
Vorzugsschutzgebühr: 2,00 M

Inhaltsverzeichnis

Literaturverzeichnis

[1] Bormann, Dr. Jürgen u. a.
Programmierung und Nutzung von Rechenanlagen
Verlag Die Wirtschaft Berlin
Teil 3: FORTRAN
Teil 7: Aufgabensammlung für ökonomische und ingenieurökonomische Fachrichtungen

0. Vorbemerkungen

Im Lehrabschnitt „Funktionsprinzipien programmgesteuerter digitaler Rechenautomaten" wurden Ihnen Grundkenntnisse über den Aufbau und Wirkungsweise von elektronischen Informationsverarbeitungssystemen vermittelt. Bei der gerätetechnischen Erläuterung einzelner Baugruppen erkannten Sie bereits die enge Wechselwirkung von Gerätetechnik und Programmen. Die Programme enthalten in maschinenverständlicher Schreibweise die Befehlsfolge für die Verarbeitung der Informationen durch eine Informationsverarbeitungsanlage. Diese Befehlsfolge stellt eine besondere Form eines Algorithmus dar. Das Aufstellen des Algorithmus wird somit zum Kernproblem bei der Programmierung jeder Informationsverarbeitungsanlage, z. B. auch des Taschen- oder Tischrechners. Bekanntlich erfordert aber jede Lösung eines Problems in der Praxis die Verarbeitung von Informationen. Hierbei ist es zunächst völlig unwichtig, ob die Problemlösung manuell oder mit Hilfsmitteln erfolgt. Ein guter Ingenieur zeichnet sich durch die Anwendung einer optimalen Lösungsstrategie aus, d. h., er arbeitet einen gedanklich vorgegebenen Lösungsweg, einen Algorithmus, ab. Während bei der manuellen Verarbeitung dieser Algorithmus oft routinemäßig und damit unbewußt angewendet wird, steht vor dem Anwender der EDVA und Mikrorechner das Problem, der Anlage jeden einzelnen Lösungsschritt mit allen seinen Bedingungen exakt vorzugeben. Dieses erfordert ein strenges logisches algorithmisches Denken durch den Programmierer. Ziel des Kapitels „Algorithmierung von Prozessen" innerhalb dieses Lehrgebietes ist es, Sie mit den Grundlagen der Algorithmierung vertraut zu machen und Sie zu befähigen, selbständig Algorithmen aufzustellen. Sie sollen dabei die Möglichkeiten und Grenzen der Programmierung erkennen. Die Realisierung dieser Zielstellung ist jedoch nur dann möglich, wenn Sie sich aktiv mit der Problematik auseinandersetzen.

Während der Lehrbrief „Programmablaufplanung" die theoretischen Grundlagen enthält, ist die vorliegende Aufgabensammlung ein wesentlicher Bestandteil für die aktive Wissensaneignung. Diese Aufgabensammlung ist für alle Studienrichtungen vorgesehen und umfaßt allgemeine und spezielle fachtypische Aufgaben. Sie setzt dabei mathematische und physikalisch-technische Grundkenntnisse voraus.

Die Darstellung der Algorithmen in Form der Programmablaufpläne erfolgt in Analogie zum Lehrbrief „Programmablaufplanung".

Die wichtigsten vorkommenden Abkürzungen sind hier nochmals zusammengestellt.

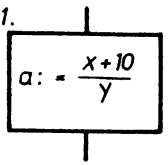
E:	Eingabeoperation (von der Peripherie zum Hauptspeicher)
A:	Ausgabeoperation (vom Hauptspeicher zur Peripherie)
LL0	Lochbandleser 0 (Eingabegerät)
LL1	Lochbandleser 1 (Eingabegerät)
SE0	Bedienschreibmaschine (Eingabegerät)
SE1	Schreibmaschine 1 (Eingabegerät)
LS0	Lochbandstanzer 0 (Ausgabegerät)
LS1	Lochbandstanzer 1 (Ausgabegerät)
SA0	Bedienschreibmaschine 1 (Ausgabegerät)
SA1	Schreibmaschine 1 (Ausgabegerät)
SD0	Seriendrucker 0 (Ausgabegerät)
SD1	Seriendrucker 1 (Ausgabegerät)

Für die mit dem Symbol (*) gekennzeichneten Aufgaben wurde am Ende des Lehrbriefes die Lösung angegeben. Auf Hinweise für den Lösungsweg wurde aus Platzgründen verzichtet. Wie allgemein bekannt ist, sind für die Lösung stets mehrere verschiedene PAP möglich. Dieser Lehrbrief enthält immer nur eine mögliche Lösungsvariante.

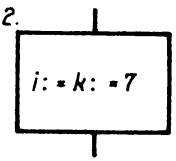
1. Allgemeine PAP-Übungen

1.1. Prüfen Sie nachstehende PAP-Symbole auf ihre richtige Anwendung.

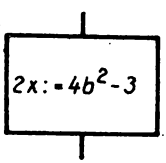
1.1.1.



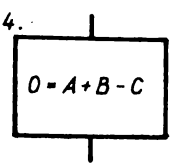
1.1.2.



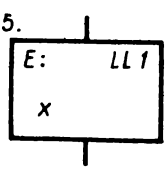
1.1.3.



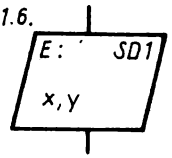
1.1.4.



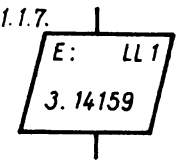
1.1.5.



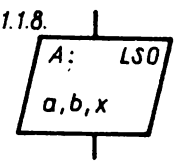
1.1.6.



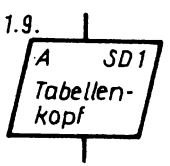
1.1.7.



1.1.8.

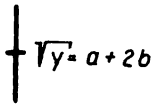


1.1.9.

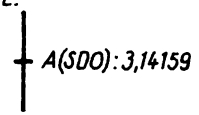


1.1.10.

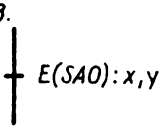
1.1.11.



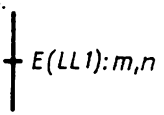
1.1.12.



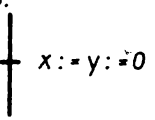
1.1.13.



1.1.14.

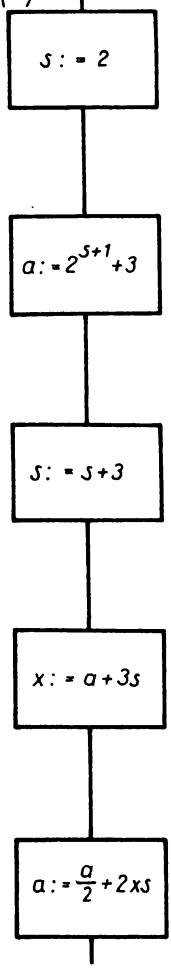


1.1.15.

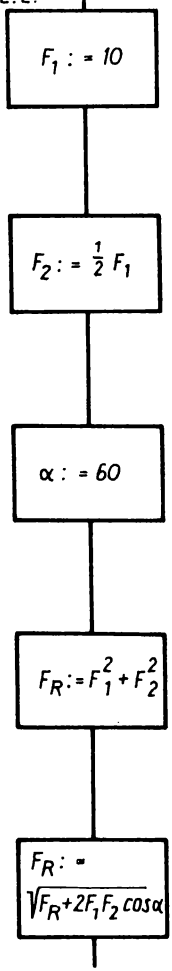


1.2. Arbeiten Sie die nachstehenden Operationsfolgen ab, und geben Sie die letzte Speicherplatzbelegung für alle im PAP-Ausschnitt enthaltenen Variablen an!

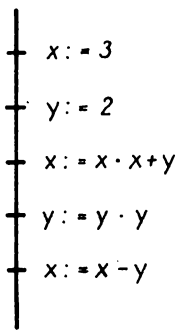
1.2.1.(•)



1.2.2.

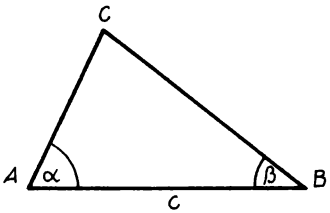


1.2.3.



2. Übungen zu linearen PAP

2.1. Gegeben sind die Winkel α und β (in Grad) und die Seite c (in cm) eines beliebigen Dreiecks.



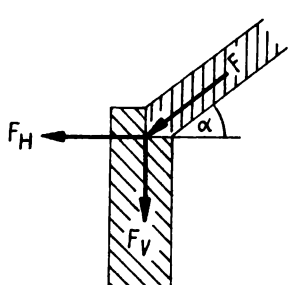
Erstellen Sie einen PAP zur Berechnung des Flächeninhaltes A und des Umfangs U des Dreiecks. Die Eingabe der Werte soll in obiger Reihenfolge über LL1 erfolgen.
Über Seriendrucker 1 sind die Werte α, β, c, A (in cm^2) und U (in cm) auszugeben.

2.2. Stellen Sie einen linearen PAP zur Berechnung der Oberfläche O_K und des Volumens V einer Kugel und der Oberfläche O_H einer Halbkugel auf. Der Radius r in cm der Kugel ist über die Bedienschreibmaschine 0 abzufordern. Die Ausgabe der berechneten Werte soll über dasselbe Gerät erfolgen. Bei der Berechnung von O_H ist die Kreisfläche zu berücksichtigen.

2.3. (*) Die Gesamtrechenzeit eines Programms wird in der EDV wesentlich durch die Anzahl der auszuführenden Multiplikationen und Divisionen beeinflußt.
Formulieren Sie den PAP der Aufgabe 2.2. so, daß eine minimale Anzahl von Multiplikationen und Divisionen durchgeführt werden muß.

2.4. Stellen Sie einen linearen PAP zur Berechnung der Resultierenden auf, wenn zwei Kräfte F_1 und F_2 (in N) einen gemeinsamen Angriffspunkt haben und einen Winkel α einschließen! Die Kräfte F_1, F_2 und der Winkel α sind in dieser Reihenfolge über den Lochbandleser 1 einzugeben. Es sollen die Werte F_1, F_2, α und F_R (in N) über Seriendrucker 1 ausgegeben werden.

2.5. Ein Sparren schließt mit der Horizontalebene einen Winkel α ein und erzeugt in seiner Richtung eine Kraft F (in N). Der Horizontalschub F_H und die Vertikaldruckkraft F_V in der Mauer am Angriffspunkt des Sparrens sind zu ermitteln. Erstellen Sie einen PAP für die Berechnung von F_H und F_V (in N)! Die Eingabe von F und α soll über die Bedienschreibmaschine erfolgen.
F_H und F_V sind über dasselbe Gerät auszugeben.

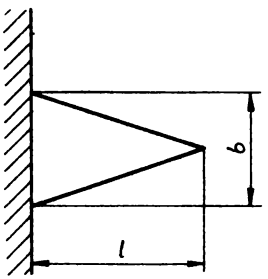


2.6. Von einer Dreiecksfeder sind folgende Angaben bekannt:

(*) Blattbreite b (in mm),
Blattdicke s (in mm),
Blattlänge l (in cm),
zulässige Biegespannung σ_{zul} (in N/m²) und Elastizitätsmodul E (in N/m²).

Erstellen Sie einen PAP zur Berechnung der Federkonstanten c (in N/m) und der maximalen Durchbiegung f (in mm)! Die Eingabe der Werte soll über Lochbandleser 1 erfolgen. Die berechneten Werte und die Eingangsdaten sind auf dem Seriendrucker 1 auszugeben.

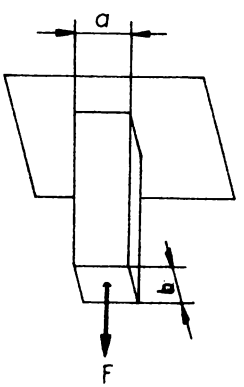
Hinweis:



$$c = \frac{b s^3 E}{6 l^3}$$

$$f = \frac{l^2 \sigma_{zul}}{s E}$$

2.7. Eine rechteckige Stange mit einer maximalen Zugfestigkeit von σ_{max} (in N/m²) wird durch eine Zugkraft F (in N) auf Zug beansprucht. Berechnen Sie mit Hilfe eines PAP die Mindestmaße (in cm) für die Kantenlängen der Stange, wenn die Kanten ein Verhältnis von 1 : 2 aufweisen sollen. Die Eingabe von σ_{max} und F soll über Lochbandleser 0 erfolgen. Als Ausgabegerät für die Eingangsgrößen und für die Ergebnisse ist der Seriendrucker 1 vorzusehen.

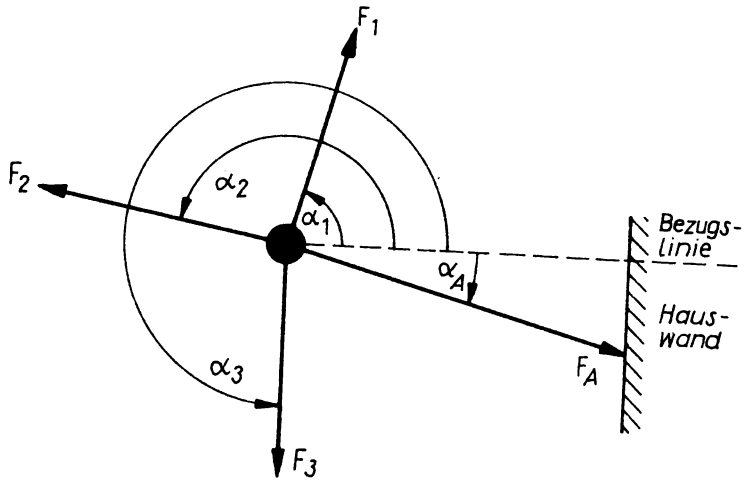


Es soll gelten:

a : b = 1 : 2

2.8. Am Kopf eines Fahrleitungsmastes sollen 2 Spanndrähte befestigt werden, die den Mast mit den Kräften F_1, F_2 und F_3 (jeweils in N) auf Zug beanspruchen.
Ein Ankerdraht soll vom Mastkopf zu einer Hauswand so gezogen werden, daß der Mast nicht auf Biegung beansprucht wird. Stellen Sie einen PAP zur Berechnung der Richtung (Winkel in Grad) und der Spannkraft (in N) des Ankerdrahtes auf! Die Wirkungslinien der Kräfte F_1, F_2 und F_3 bilden mit einer Bezugslinie die Winkel α_1, α_2 und α_3 (in Grad). Alle Drähte sind horizontal mit dem Mast verbunden.

Skizze:



Bemerkungen: 1. Es gilt $-90° \leqq \alpha_A \leqq 90°$.

2. Es stehen die Winkelfunktionen sin x, cos x und arctan x zur Verfügung.
3. Bei diesen drei Funktionen erfolgt jeweils die Winkelangabe in Bogenmaß.
4. Für die Lösung dieser Aufgabe wird empfohlen, die Kräfte F_i in ihre Teilkräfte in x-y-Richtung zu zerlegen.

Die Größen F_i und α_i sind über Lochbandleser einzugeben. Die Ausgabe soll über einen Drucker realisiert werden

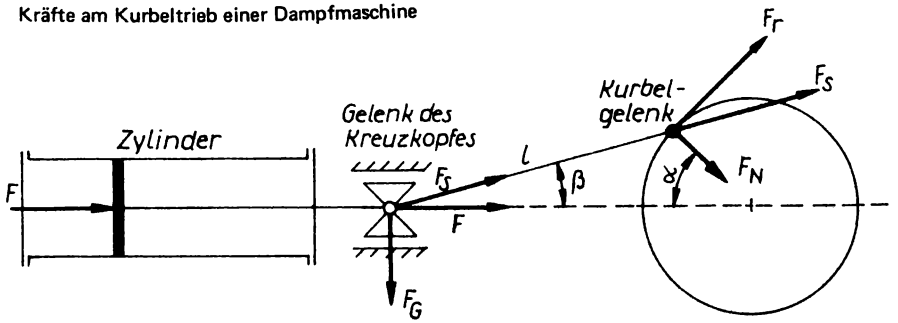
2.9. Stellen Sie einen PAP für die Berechnung der Seilkraft F_s (in kN) für ein Hebezeug auf. Es gilt

$$F_s = \frac{F_L}{2 \cdot \eta_{FL} \cdot \eta_R^2}$$

Hierbei ist F_L die maximale Seillast (in N), η_{FL} der Flaschenzugswirkungsgrad und η_R der Rollenwirkungsgrad. Die Eingabe dieser drei Größen soll über Lochbandleser und die Ausgabe über Drucker erfolgen.

2.10. Bei einer Dampfmaschine ist die Kolbenkraft F gegeben. Für eine bestimmte Kurbelstellung, also bei bekannten Winkeln α und β sind die Tangentialkraft F_T und die Normalkraft F_N an der Kurbel zu ermitteln.

Kräfte am Kurbeltrieb einer Dampfmaschine

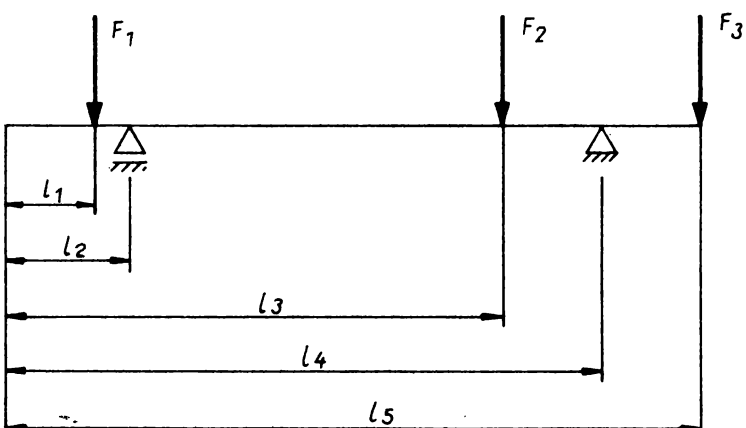


Es gilt:

Schubstangenkraft	F_s	$= F/\cos\beta$
Gleitbahnkraft	F_G	$= F \cdot \tan\beta$
Normalkraft	F_N	$= F_s \cdot \cos(\alpha+\beta)$
Tangentialkraft	F_T	$= F_s \cdot \sin(\alpha+\beta)$

Stellen Sie einen PAP zur Berechnung der Größen F_G, F_N und F_T (alle in kN) auf, wenn die Größen F (in kN), α (in Grad), die Länge der Schubstange l (in mm) und der Kurbelradius r (in mm) auf einem Lochband stehen.
Die Ausgabe der Größen soll über die Bedienschreibmaschine realisiert werden.

2.11. Gegeben ist das Belastungssystem eines Trägers auf zwei Stützen mit drei Einzellasten. Stellen Sie einen PAP zur Berechnung der Auflagerkräfte F_A und F_B auf.



Die Eingabe der Größen F_1 bis F_3 (in N) und l_1 bis l_5 (in mm) soll über einen Lochbandleser und die Ausgabe über Drucker erfolgen

2.12. Beim Übergang elektrischer Feldlinien von einem Dielektrikum zum anderen gilt das Brechungsgesetz

$$\frac{\tan \alpha_1}{\tan \alpha_2} = \frac{\epsilon_1}{\epsilon_2}$$

Stellen Sie für folgende Bedingungen einen PAP zur Berechnung von α_2 auf:
- ϵ_1, ϵ_2 und α_1 (in Grad) sind gegeben und werden über Lochbandleser 1 eingelesen.
- Zur Berechnung stehen die Funktionen sin x, cos x und arctan x zur Verfügung. Bei der Verwendung dieser Funktionen beziehen sich alle Winkelangaben auf Bogenmaß.
- Die Ausgabe der Eingangswerte und von α_2 (in Grad) hat über Seriendrucker 1 zu erfolgen.

2.13. Die Kapazität eines Plattenkondensators berechnet sich nach der Formel

$$C = \frac{\epsilon \cdot A}{s} \quad \text{mit } \epsilon = \epsilon_0 \cdot \epsilon_r \ .$$

Erstellen Sie einen PAP zur Berechnung des Durchmessers (in cm) der Kondensatorplatten auf, wenn die Werte von C (in pF) und s (in mm) über Schreibmaschine 0 bereitgestellt werden! Für ϵ_r ist 3,54 einzusetzen.
Die Eingangsdaten sind zusammen mit dem Ergebnis über Lochbandstanzer 1 auszugeben.

3. Übungen zu verzweigten PAP

3.1. Führen Sie für nachstehende Aufgaben einen Trockentest durch und geben Sie die Werte an, die ausgegeben werden. Die Eingangsdaten sind durch Sie so zu wählen, daß mindestens jeder Programmzweig einmal durchlaufen wird.

3.1.1. (*) Gegeben ist nachstehender PAP zur Berechnung des SV- und FZR-Beitrages für einen Werktätigen mit dem Bruttolohn L. Die Kennziffer KZ gibt Auskunft über die Zugehörigkeit zur FZR, wobei

$$KZ = \begin{cases} 0 \text{ nicht in der FZR} \\ 1 \text{ in der FZR ohne Höchstgrenze} \end{cases}$$

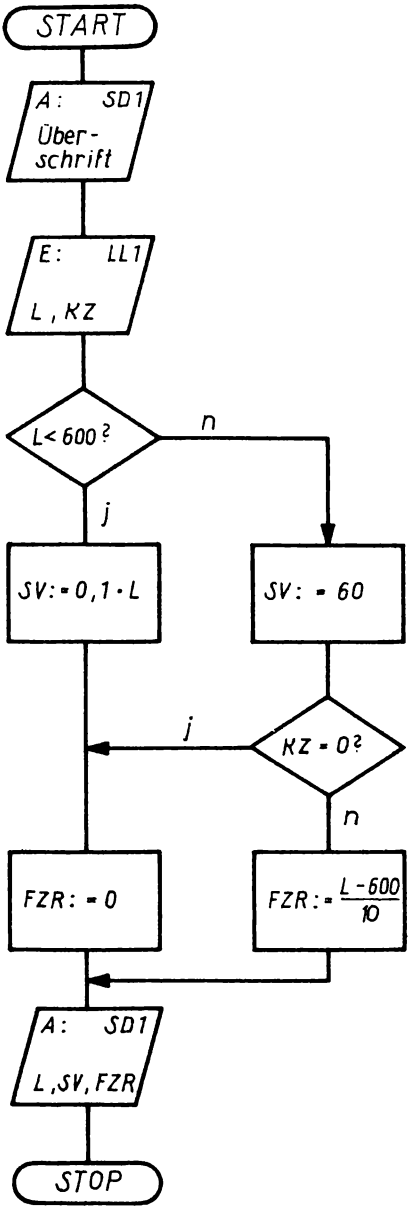
gilt.

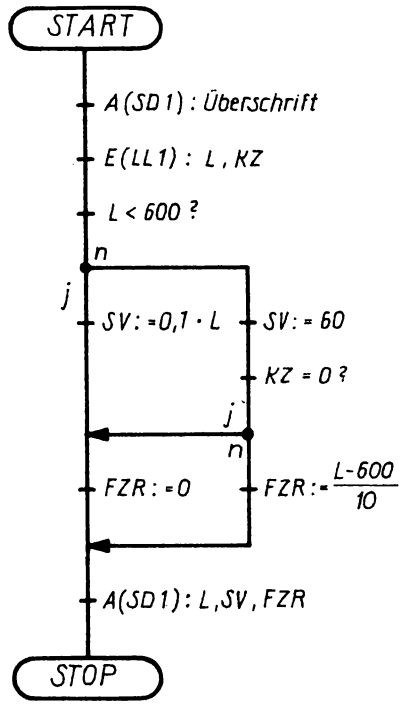
1. Mathematisches Modell:

(a) $L < 600\,M$: $SV = 0{,}1 \cdot L$
$FZR = 0$

(b) $L \geqq 600\,M$: $SV = 60$
$KZ = 0$: $FZR = 0$
$KZ = 1$: $FZR = 0{,}1 \cdot (L - 600)$

2. **PAP**


START
A: SD1
Über-
schrift
E: LL1
L, KZ
L< 600?
n
j
SV:=0,1·L
SV: = 60
j
KZ = 0?
n
FZR: = 0
FZR:= L-600/10
A: SD1
L, SV, FZR
STOP


START
A(SD1): Überschrift
E(LL1): L, KZ
L < 600?
n
j
SV:=0,1·L
SV:=60
KZ = 0?
j
n
FZR:=0
FZR:= L-600/10
A(SD1): L, SV, FZR
STOP

3.1.2. Für einen beliebigen x-Wert soll der reelle Funktionswert y nach der Vorschrift

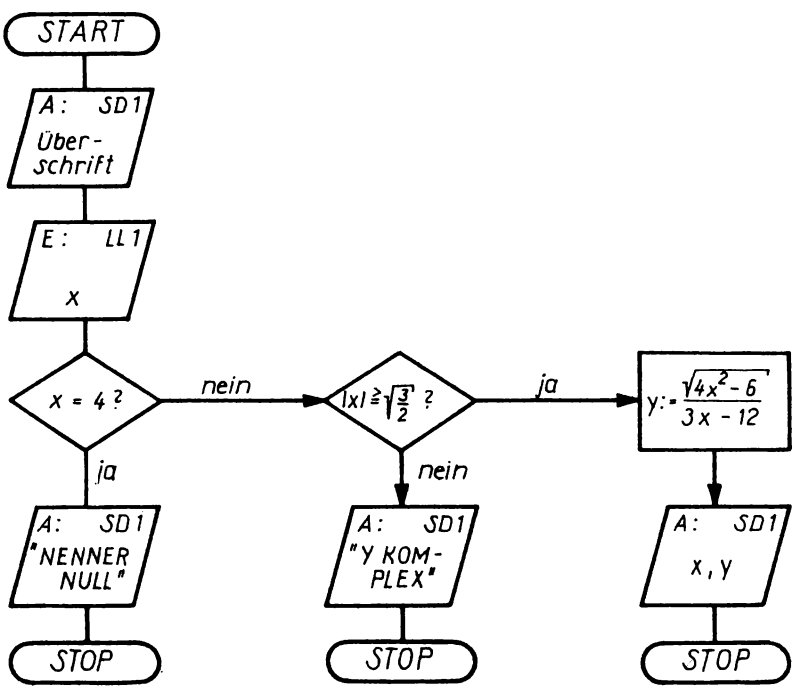
$$y = f(x) = \frac{\sqrt{4x^2 - 6}}{3x - 12}$$

berechnet werden. Ist der Funktionswert y nicht definiert, so soll der Text „NENNER NULL" gedruckt werden. Erhalten wir für y eine komplexe Zahl, so ist der Text „Y KOMPLEX" zu drucken.

1. **Mathematisches Modell**
   (a) y nicht definiert für $3x - 12 = 0$,
   folglich für $x = 4$.
   (b) y komplex für $4x^2 - 6 < 0$, folglich für

$$-\sqrt{\frac{3}{2}} < x < \sqrt{\frac{3}{2}}$$

2. **PAP**



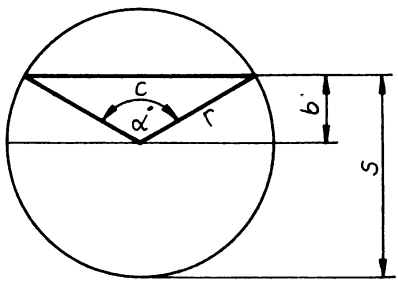
3.1.3. In einem liegenden Benzinfaß (Zylinder) mit dem Innendurchmesser d (in mm) und der lichten Länge l (in cm) wird mit Hilfe einer eingetauchten Meßlatte der Flüssigkeitsstand s (in mm) abgelesen. Mit dem nachfolgenden PAP soll die sich noch im Faß befindliche Restmenge Benzin in Litern ermittelt werden.
Bemerkung: Bei der Anwendung der arctan-Funktion erhält man in der EDV den Winkel in Bogenmaß.

1. Mathematisches Modell

$V = A \cdot l$

$r = \frac{d}{2}$

$\tan \frac{\alpha}{2} = \frac{c}{2-b}$



1. Fall
Es gilt $s \leqq r$
$b = r - s$

$A = \frac{\alpha^\circ}{360^\circ} \cdot \pi r^2$

$A = \frac{1}{2} \alpha r$

2. Fall
Es gilt $s \geqq r$
$b = s - r$

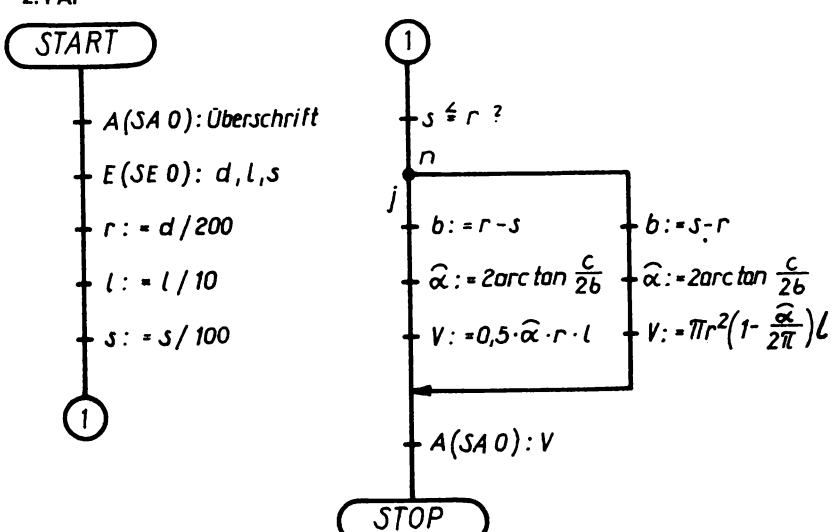
$A = \pi r^2 - \frac{\alpha^\circ}{360^\circ} \pi r^2$

$A = \pi r^2 \left(1 - \frac{\alpha^\circ}{360^\circ}\right)$

$A = \pi r^2 \left(1 - \frac{\alpha}{2\pi}\right)$

Aus der Maßeinheitenkontrolle folgt,
daß alle Längenangaben in dm erforderlich sind.

2. PAP



3.2.1. Gegeben sind über Lochbandleser 1 die Koeffizienten a, b und c einer Funktion
$y = f(x) = ax^2 + bx + c$
Erstellen Sie einen PAP zur Berechnung der reellen Nullstellen dieser Funktion, falls $a \neq 0$ gilt! Treten komplexe Nullstellen auf, so ist der Text „KOMPLEXE NULLSTELLEN" zu drucken. Die Ausgabe der Ergebnisse und der Protokolldruck der Eingangsdaten sollen über den Seriendrucker 1 erfolgen.

3.2.2. Gegeben sind über Lochbandleser 1 die Koeffizienten a, b und c einer Funktion
$y = f(x) = ax^2 + bx + c$
Erstellen Sie einen PAP zur Berechnung der reellen Nullstellen dieser Funktion, falls $a^2 + b^2 \neq 0$ gilt (d. h., a und b sind nicht gleichzeitig Null)!
Treten komplexe Nullstellen auf, so ist der Text „KOMPLEXE NULLSTELLEN" zu drucken.
Die Ausgabe der Ergebnisse und der Protokolldruck der Eingangsdaten soll über den Seriendrucker 1 erfolgen.

3.2.3. (*) Gegeben sind die Koeffizienten a, b und c einer Funktion
$y = f(x) = ax^2 + bx + c$
Erstellen Sie einen PAP zur Berechnung der reellen und komplexen Nullstellen dieser Funktion, falls $a^2 + b^2 \neq 0$ gilt!
Die Ein- und Ausgabe soll über Bedienschreibmaschine erfolgen.

3.3. Erstellen Sie einen PAP zur Berechnung des reellen Funktionswertes

$$y = \frac{1}{\sqrt{1{,}2x^2 - 30}}$$

für einen beliebigen x-Wert, der über eine Bedienschreibmaschine abzufordern ist. Die Ausgabe von y soll über dasselbe Gerät erfolgen!

3.4. (*) Gegeben sind über Lochbandleser 1 die 3 Seiten a, b und c eines Dreiecks in cm. Mit Hilfe eines PAP soll ermittelt werden, ob das Dreieck
a) spitzwinklig
b) rechtwinklig oder
c) stumpfwinklig
ist. Der entsprechende Text ist über Seriendrucker 1 zu drucken.

Hinweise:
1. Zur Abarbeitung des Programms innerhalb der EDVA stehen nur die Winkelfunktionen sin(x), cos(x) und arctan(x) zur Verfügung.
2. Wollen Sie die Winkelwerte mittels der Funktionen arcsin(x) und arccos(x) ermitteln, so müssen Sie diese Funktionen auf die Funktion arctan(x) zurückführen. Die Formeln für die Umrechnungen sind dem Tafelwerk zu entnehmen.
3. Bei der Anwendung des Sinussatzes ist zu beachten, daß Sie als Ergebnis zwei Winkel erhalten.
4. Es wird empfohlen obiges Problem über den Cosinussatz und den Quadrantenbeziehungen zu lösen

$\cos\alpha > 0$ für $0° \leqq \alpha < 90°$
$\cos\alpha = 0$ für $\alpha = 90°$
$\cos\alpha < 0$ für $90° < \alpha \leqq 180°$

3.5. Bekanntlich werden an die Datensicherheit bei der Anwendung der EDV große Anforderungen gestellt. Es kommen in der Praxis spezielle Verfahren zur Kontrolle der Eingangsdaten zur Anwendung, z. B. das Bilden von Prüfziffern. Prüfen Sie mit Hilfe eines PAP folgende siebenziffrigen Zahlen (einschließlich Prüfziffer) auf ihre Richtigkeit, wenn die Zahlen folgenden Aufbau haben:

$z_1\ z_2\ z_3\ z_4\ z_5\ z_6 - z_p$

(z_1 bis z_6 = Ziffern der Zahl; z_p = Prüfziffer)
Die Prüfziffer z_p wird nach folgender Vorschrift gebildet:

1. $z = z_1 + 2z_2 + z_3 + 2z_4 + z_5 + 2z_6$
2. z_p ist dann die Differenz von z bis zur nächsten vollen Zehnerstelle. Ist z eine volle Zehnerstelle, so ist $z_p = 0$.

Über Seriendrucker 1 ist die Zahl und der entsprechende Text zu drucken.

Die Eingabe der Zahl erfolgt über Lochbandleser 1 in zwei Teilen:

1. Zahl besteht aus den Ziffern z_1 bis z_6 als eine Zahl
2. Zahl ist die Ziffer z_p.

Beispiele:

$z_1 - z_6$	z_p	
200017	3	$:z = 2 + 2 \cdot 0 + 0 + 2 \cdot 0 + 1 + 2 \cdot 7 = 17$
		$z_p = 20 - 17 = 3 \longrightarrow$ richtig
581264	6	$:z = 5 + 2 \cdot 8 + 1 + 2 \cdot 2 + 6 + 2 \cdot 4 = 40$
		$z_p = 40 - 40 = 0 \neq 6 \longrightarrow$ falsch

Hinweise:

1. In der EDV gibt es zwei Arten der Division.
   1. Das Ergebnis der Division zweier Zahlen liefert eine reelle Zahl, z. B. 23,0 : 10,0 = 2,3
   2. Das Ergebnis der Division zweier ganzer Zahlen ist wieder eine ganze Zahl, z. B. 23 : 10 = 2
2. Als mathematische Standardfunktion steht in den meisten Fällen in der EDV die Modulusfunktion zur Verfügung. Der Funktionwert der Modulusfunktion MOD (a,b) ist der Rest der ganzzahligen Division a : b, z. B.
   MOD (23,10) = 3
   MOD (17,10) = 7
   MOD (8,2) = 0

3.6. Gegeben ist eine 6-stellige Studentennummer über Lochbandleser 1. Mit Hilfe eines PAP soll entschieden werden, ob der Student im 1., 2. oder 3. Studienjahr ist und welcher Abteilung er angehört. Die Studentennummer hat folgenden Aufbau:

$x_1\ x_2\ x_3\ x_4\ x_5\ x_6$. Es gilt:

x_1 – Abteilungsnummer
x_2 – Fachrichtungsnummer
x_3 – letzte Ziffer der Matrikel
x_4 – lfd. SG-Nr. innerhalb der Fachrichtung und des Studienjahres
x_5 – Studienformnummer
x_6 – Studienortnummer

Beispiel: 2 2 9 2 0 0

- 0 — Dresden
- 0 — Direktstudium
- 2 — 2
- 9 — 29. Matrikel
- 2 — Kraftfahrzeugtechnik
- 2 — Fahrzeugtechnik

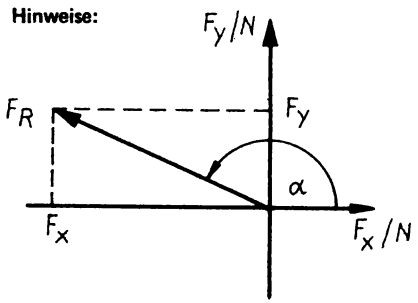
Hinweise: siehe Aufgabe 3.5.

3.7. Gegeben ist eine Wassermasse (in kg). Diese Menge ist von einer Anfangstemperatur t_1 (in °C) auf eine Endtemperatur t_2 (< 100 °C) zu erwärmen.
Berechnen Sie die erforderliche Wärmemenge (in Joule) mit Hilfe eines PAP!
x, t_1 und t_2 sind in dieser Reihenfolge über Lochbandleser 1 einzugeben. Die Ausgabe soll über Seriendrucker 1 erfolgen.

Hinweis: t_1 und t_2 können auch negative Werte annehmen, sind aber ungleich Null.

3.8. Gegeben sind über Lochbandleser 1 zwei Kräfte F_x und F_y, die einen gemeinsamen Angriffspunkt besitzen und einen Winkel von 90° einschließen. Berechnen Sie mit Hilfe eines PAP den Betrag der resultierenden Kraft F_R und den Winkel α. α ist der mathematisch positive Winkel zwischen F_R und einer Wirkungslinie, die parallel zur Kraft F_x verläuft. Die Richtungen von F_x und F_y werden durch das Vorzeichen bestimmt. Über Seriendrucker 1 sind die Werte F_x, F_y, F_R (in N) und α (in Grad) auszugeben.

Hinweise:



1. Zur Berechnung von α ist die arctan-Funktion zu verwenden. Sie gilt in der EDV aber nur für die Hauptwerte $(-\frac{\pi}{2} < \arctan x < \frac{\pi}{2})$ d. h., zur endgültigen Ermittlung von α ist der Quadrant zu berücksichtigen. Als Ergebnis der arctan-Funktion erhalten wir den Winkel im Bogenmaß.
2. Es gilt F_x und F_y ungleich Null.

3.9. In einem Wechselstromkreis werden der Strom (in A), die Spannung (in V) und die Wirkleistung (in W) gemessen. Formulieren Sie das mathematische Modell und stellen Sie einen PAP zur Beurteilung des Leistungsfaktors des Netzes auf! Dabei gelten folgende Bedingungen:

1. Die gemessenen Werte werden über Lochbandleser 1 eingelesen.
2. Ist der Leistungsfaktor $\cos \varphi = 0{,}9$ und besser, sind Wirkleistung, Scheinleistung und Blindleistung zu drucken.
3. Ist der Leistungsfaktor schlechter als $\cos \varphi = 0{,}9$, ist die kapazitive Blindleistung zu ermitteln, die eine Blindleistungskompensation bis zum Grenzwert $\cos \varphi = 0{,}9$ realisiert. Diese ist zu drucken!
4. Der Druck der Eingangsdaten und der Ergebnisse soll über Seriendrucker 0 erfolgen.

3.10. Ein Körper wird mit einer Geschwindigkeit von v_0 (in m/s) senkrecht nach oben geschossen. Berechnen Sie mit Hilfe eines PAP die Höhe h(in km)und die Geschwindigkeit v(in m/s) des Körpers zum Zeitpunkt t (in s). Geben Sie neben diesen Werten weiterhin durch einen entsprechenden Textdruck an, ob der Körper noch steigt oder ob er fällt. Die Eingabe der Werte v_0 und t soll über Lochbandleser 1 erfolgen. Der Druck ist auf dem Seriendrucker 1 zu realisieren.

3.11. Stellen Sie einen PAP zur Berechnung des SV und des FZR-Beitrages für einen Werktätigen mit dem Stundenlohnsatz s auf! Für den Werktätigen liegen auf Lochband (Eingabe über Lochbandleser 1) folgende Angaben vor:

s – Stundenlohnsatz
a – Arbeitstage zu 8,75 Stunden
v – Kennziffer über die Zugehörigkeit zur FZR
  v=0 : keine Mitgliedschaft in der FZR
  v=1 : Mitgliedschaft in der FZR mit der Grenze von 1200,– M Bruttolohn
  v=2 : Mitgliedschaft in der FZR ohne Grenze.

Die Angaben s, a, v, SV und FZR sind über Seriendrucker 1 zu drucken. SV und FZR sind außerdem noch über den Lochbandstanzer 1 auszugeben.

3.12. Für die Beschäftigten eines Betriebes soll der Grund- und Mehrleistungslohn errechnet werden.

Dazu liegen auf Lochband folgende Angaben vor:

B – Beschäftigtennummer
T – Arbeitstage zu 8,75 Stunden
G – Stundenlohnsatz laut Lohngruppe
N – Normerfüllung.

Stellen Sie einen PAP auf, der für einen Beschäftigten folgendes Druckbild (über Seriendrucker 1) realisiert:

Beschäft.-Nr. Grundlohn Mehrleistungslohn Bruttolohn

Hinweise:

1. Die Eingabe soll über Lochbandleser 1 erfolgen.
2. Der Bruttolohn errechnet sich aus Grund- und Mehrleistungslohn.
3. Der Grundlohn GL ergibt sich in Abhängigkeit von T.
4. Für den Mehrleistungslohn ML gilt:

$N < 100\,\% \Rightarrow ML = 0$
$N = 100\,\% \Rightarrow ML = 5\,\%$ vom GL
$100\,\% < N \leq 105\,\% \Rightarrow ML = 10\,\%$ vom GL
$105\,\% < N \Rightarrow ML = 20\,\%$ vom GL

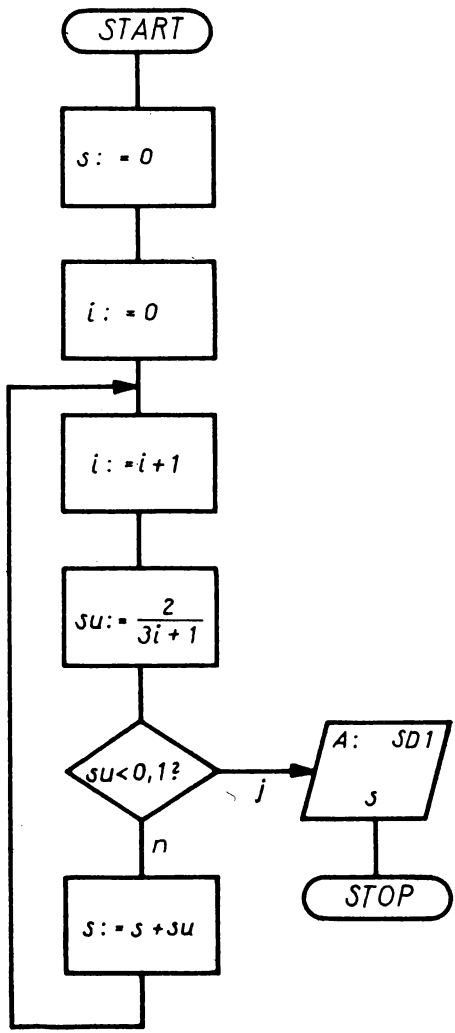
4. Übungen zu zyklischen PAP

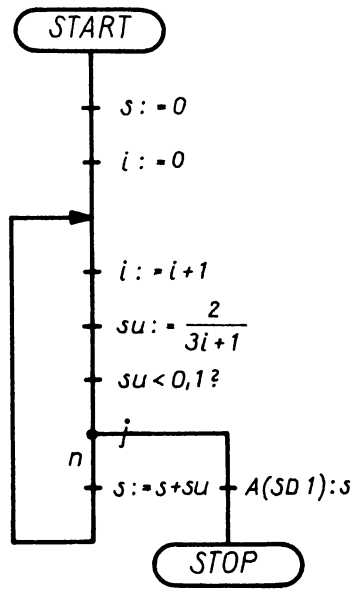
4.1. Führen Sie für nachstehende PAP einen Trockentest durch, und geben Sie die Werte an, die ausgegeben werden!

4.1.1. Gegeben ist ein PAP zur Berechnung der Summe
(*)

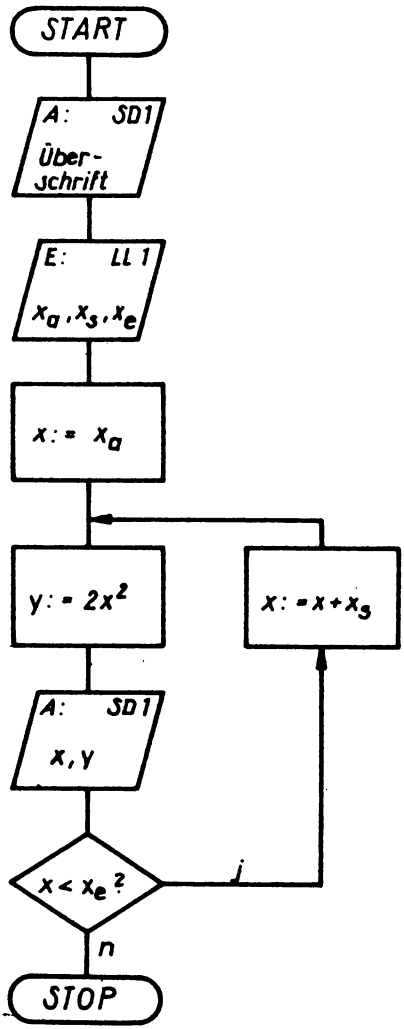
$$S = \sum_{i=1}^{\infty} \frac{2}{3i+1} \quad .$$

Die Summe ist abzubrechen, wenn die Differenz zweier aufeinanderfolgender Teilsummen kleiner als $\epsilon = 10^{-1}$ ist.





4.1.2.



Gegeben ist ein PAP zur Berechnung einer Wertetabelle für die Funktion

$$y = f(x) = 2x^2$$

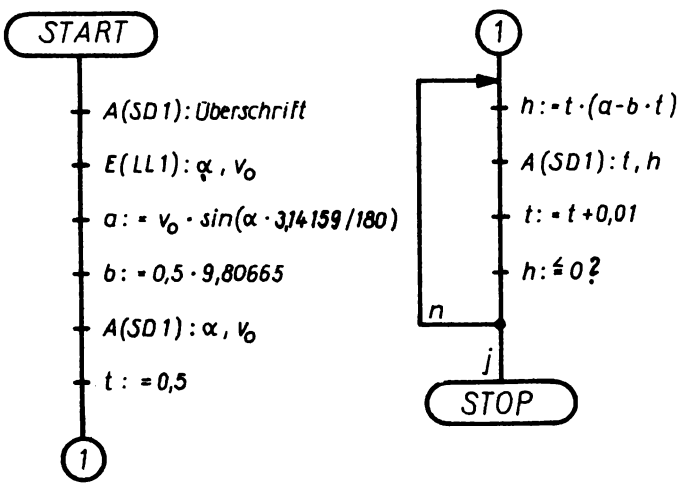
für $x = x_a(x_s)x_e$. x_a, x_s und x_e sind über Lochbandleser 1 einzugeben.
Hinweis: Für den Trockentest sollen Sie selber sinnvolle Daten für x_a, x_s und x_e wählen. Der Zyklus ist dabei mindestens 3 mal zu durchlaufen.

4.1.3. Mit dem folgendem PAP soll eine Wertetabelle erstellt werden, aus der für einen schrägen Wurf die zusammengehörenden Werte für die Zeit t (in s) und der Höhe h (in m) abgelesen werden kann. Die Wertetabelle mit der Zeit t = 0,5 s beginnen. Als Schrittweite ist Δt = 0,01 s vorgesehen. Die Tabelle soll abgebrochen werden, wenn die Höhe h den Wert 0 unterschreitet. Der Abwurfwinkel α (in Grad) und die Abwurfgeschwindigkeit v_0 (in m/s) werden über Lochbandleser eingelesen. In der EDV wird bei der Anwendung der Winkelfunktion der Winkel in Bogenmaß verlangt.

1. Mathematisches Modell

$$h = f(t) = t\,(v_0 \sin\alpha - 0{,}5\,g\,t)$$

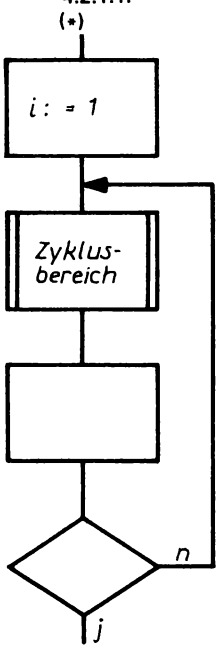
2. PAP



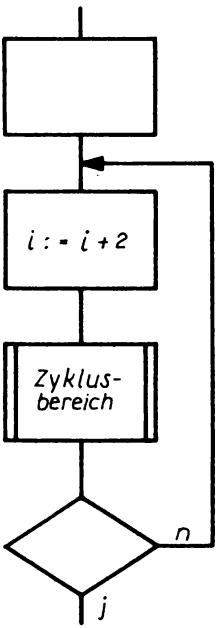
4.2. In den folgenden PAP-Ausschnitten sind die fehlenden Ergibtanweisungen und Vergleichsoperationen so einzutragen, daß der Zyklus für die angegebenen Werte durchlaufen wird.

4.2.1. Im Zyklus soll die Laufvariable i die Werte i = 1(2)15 annehmen.

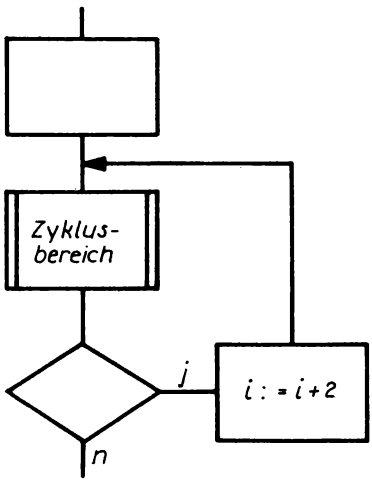
4.2.1.1.



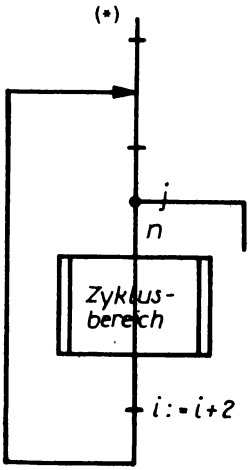
4.2.1.2.



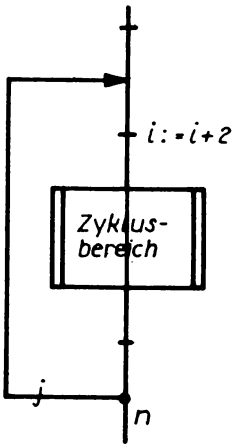
4.2.1.3.



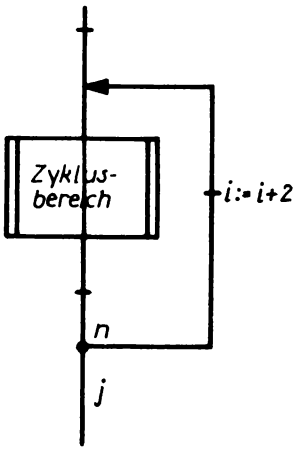
4.2.1.4.



4.2.1.5.



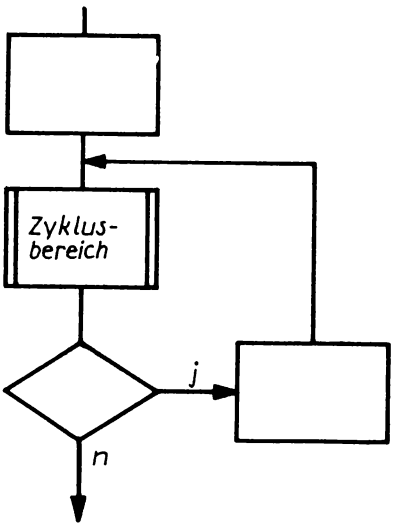
4.2.1.6.



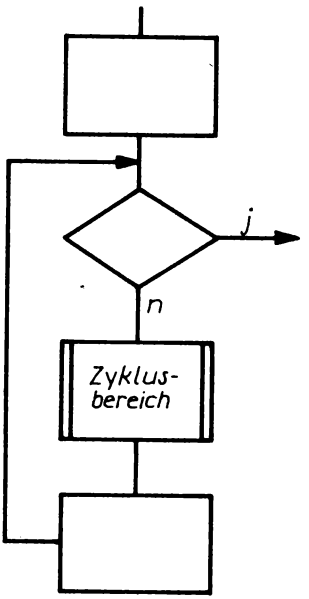
4.2.2. Im Zyklus soll die Laufvariable x die Werte

$x = x_a(x_s)x_e$ annehmen.

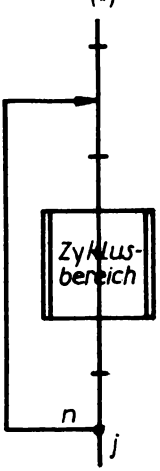
4.2.2.1.
(*)



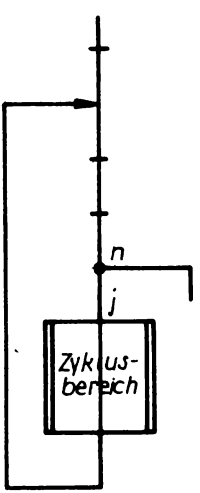
4.2.2.2.



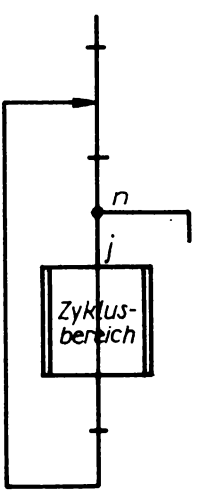
4.2.2.3.
(*)



4.2.2.4.

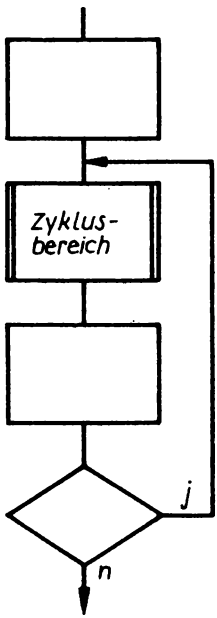


4.2.2.5.

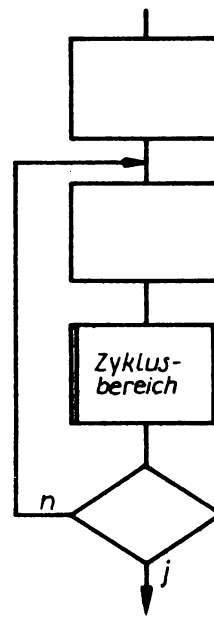


4.2.3. Im Zyklus soll die Laufvariable k die Werte k = 30(−1)0 annehmen

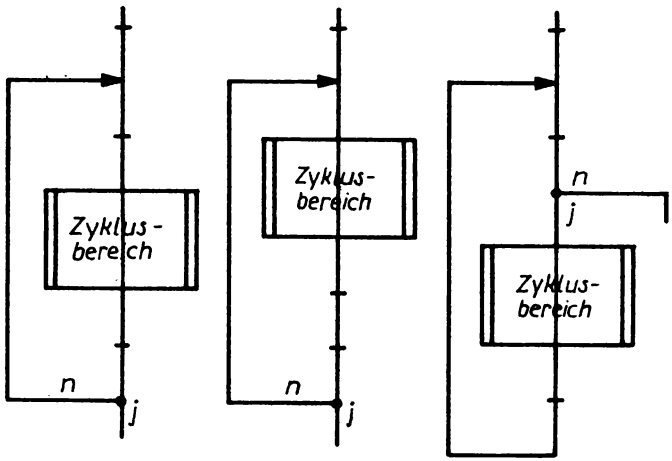
4.2.3.1.



4.2.3.2.

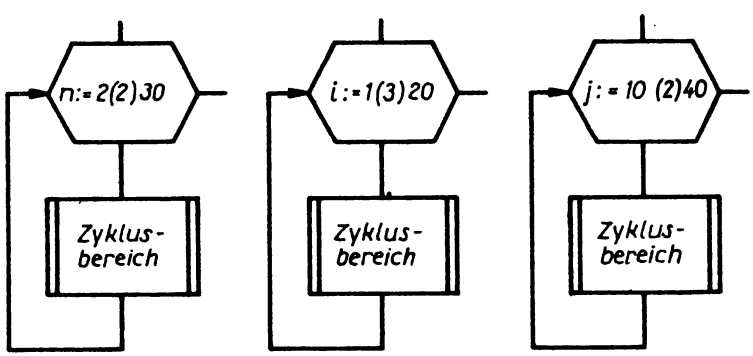


4.2.3.3. 4.2.3.4. 4.2.3.5

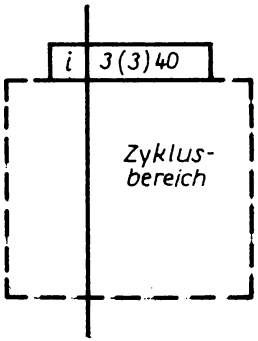


4.3. Geben Sie für die folgenden PAP-Ausschnitte die einzelnen Belegungen für die Laufvariable an. Wie oft wird der Zyklus abgearbeitet?

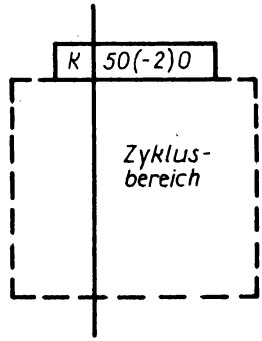
4.3.1. 4.3.2. 4.3.3.



4.3.4.



4.3.5.



4.4. (*) Stellen Sie einen PAP zur Berechnung des Produktes

$$P = n! = \prod_{i=1}^{n} i$$

auf! Die Zahl n ist über Bedienschreibmaschine einzugeben! Die Ausgabe des Ergebnisses hat ebenfalls auf der Bedienschreibmaschine zu erfolgen.

4.5. Stellen Sie einen PAP zur Berechnung der Summe

$$S = \sum_{i=0}^{10} \frac{2i + 1}{3(i + 1)}$$

auf! Der Wert S ist auf der Bedienschreibmaschine auszudrucken.

4.6. (*) Stellen Sie einen PAP zur Berechnung der Summe

$$S = \sum_{i=1}^{\infty} \frac{i}{i^2 + 1}$$

auf! Die Summe ist abzubrechen, wenn die Differenz zweier aufeinanderfolgender Teilsummen kleiner als $\epsilon = 10^{-3}$ ist. Es sind alle Partialsummen in einer Tabelle über Seriendrucker 0 zu protokollieren.

4.7. Stellen Sie einen PAP zur Berechnung der Summe

$$S = x - x^3 + x^5 - x^7 + x^9 + \ldots = \sum_{k=1}^{\infty} (-1)^{k+1} x^{2k-1}$$

auf! Der Wert von x $(-1 < x < 1)$ ist über Lochbandleser 1 einzugeben. Die Summe ist abzubrechen, wenn die Differenz zweier aufeinanderfolgender Teilsummen kleiner als $\epsilon = 0{,}5 \cdot 10^{-4}$ ist. Die Werte x und S sind über Seriendrucker 1 auszugeben.

Hinweis: Diese Aufgabe dient als Grundübung für die Aufgabe 5.9.

4.8. Der natürliche Logarithmus einer Zahl x (ln x) läßt sich für $0 < x \leq 2$ mit Hilfe der Reihe

$$\ln x = \frac{x-1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - + \ldots = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{(x-1)^k}{k}$$

berechnen. Stellen Sie einen PAP für die Berechnung des ln x mit einer Genauigkeit von $\epsilon = 10^{-4}$ auf. Die Zahl x ist über Bedienschreibmaschine einzugeben. Die Ausgabe soll ebenfalls über das Gerät erfolgen.

4.9. Stellen Sie einen PAP zur Berechnung des arithmetischen Mittelwertes von n Meßwerten auf! Ein Lochband (Eingabe über Lochbandleser 1) enthält dazu in dieser Reihenfolge folgende Angaben
1. Zahl : Anzahl der Meßwerte
2. Zahl : 1. Meßwert
3. Zahl : 2. Meßwert
usw.
Jede Zahl bildet einen Datensatz. Über Seriendrucker 0 sind alle Meßwerte und der arithmetische Mittelwert auszugeben.

4.10. Stellen Sie einen PAP zur Berechnung des arithmetischen Mittelwertes von beliebig vielen positiven Meßwerten auf! Als Endekennzeichen für die Meßwertreihe wurde die Zahl −5 vereinbart.
Die Eingabe der Meßwerte soll über Lochbandleser 1 erfolgen.
Jede Zahl bildet einen Datensatz. Die Meßwerte und der Mittelwert sind über Seriendrukker 1 auszugeben.

4.11. Gegeben sind die Koeffizienten eines Polynoms n–ten Grades

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 .$$

Stellen Sie einen PAP zur Berechnung der Koeffizienten der 1. Ableitung

$$P'(x) = b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + \ldots + b_1 x + b_0$$

auf! Die Koeffizienten $a_n, a_{n-1}, \ldots a_1$ und a_0 liegen in dieser Reihenfolge auf Lochband (Eingabe über Lochbandleser 1) vor. Vor der ersten Zahl a_n steht die Zahl n, die den Grad des Polynoms angibt. Jede Zahl bildet einen Datensatz. Das Druckbild soll alle Koeffizienten a_i und b_j enthalten. Als Ausgabegerät ist der Seriendrucker 1 zu verwenden.

4.12. Stellen Sie einen PAP zur Berechnung der Geschwindigkeit (in km/h) eines Punktes auf der Erdoberfläche für die geografischen Breiten 0°(10°)80° auf! Der Erdradius beträgt $6{,}37 \cdot 10^3$ km. Als Druckbild soll eine Wertetabelle entstehen. (Ausgabe über Seriendrucker 0.)

4.13. Für die Augenblickswerte der Leistung im Wechselstromkreis gilt

$$p = u \cdot i$$

mit

$$u = U_{max} \cdot \sin(\omega t + \varphi)$$

und

$$i = I_{max} \cdot \sin \omega t.$$

Stellen Sie einen PAP zur Berechnung und zum Druck (über Drucker) einer Wertetabelle der 3 Funktionen u, i und p in Abhängigkeit von der Zeit für eine Periode bei einer Schrittweite von 10^{-3}s auf, wenn gegeben sind:

Spannung U_{max} = 250 V
Strom I_{max} = 10 A
Frequenz f = 50 s^{-1}
Phasenwinkel φ = 0°

4.14. Beim Einschalten einer Spule (R,L) im Gleichstromkreis mit der Spannung U_0 verändern sich Strom und Spannung der Induktivität nach

$$i = \frac{U_0}{R}\left(1 - e^{\frac{-t}{\tau}}\right)$$

und

$$u = U_0 \cdot e^{\frac{-t}{\tau}}$$

mit

$$\tau = \frac{L}{R} \ .$$

Stellen Sie einen PAP zur Berechnung und zum Druck (über Seriendrucker 1) einer Wertetabelle der Funktionen i = f(t) (in A) und u = f(t) (in V) auf! Dazu gelten folgende Bedingungen

- R (in Ω), L (in mH) und U_0 (in V) werden in dieser Reihenfolge über Lochbandleser 1 eingegeben.
- Die Berechnung soll abgebrochen werden, wenn der Wert der Exponentialfunktion kleiner als 10^{-4} wird.
- Die Schrittweite der Zeit soll 1 ms betragen.

4.15. Für die Grenzgeschwindigkeit beim Rutschen eines Straßenfahrzeuges in der Kurve gilt die Beziehung

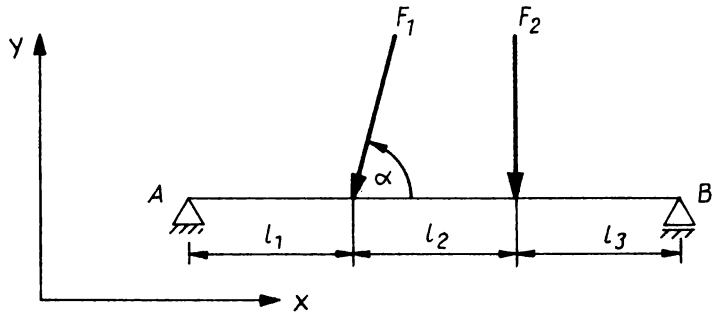
$$v = 11{,}27 \sqrt{r \cdot \tan(\beta + \rho)} \qquad \left|\frac{v}{km/h}\right| \left|\frac{r}{m}\right| \left|\frac{\beta,\rho}{grd}\right|$$

Hierin ist β der Querneigungswinkel der Straße und ρ der Haftreibungswinkel. Stellen Sie einen PAP für die Berechnung einer Wertetabelle v = f(β) für β = 0°(0,5°)10° auf! Die Größen r und ρ sind über Lochbandleser einzugeben. Die Ausgabe hat auf einem Drucker zu erfolgen.

4.16. Gegeben ist das Belastungssystem eines Trägers auf zwei Stützen mit zwei Einzellasten. Stellen Sie einen PAP für die Berechnung der Auflagerkräfte F_A und F_B auf! Der Winkel α soll die Werte $\alpha = \alpha_a\ (\alpha_s)\ \alpha_e$ annehmen. Für die Kräfte F_A und F_B sind jeweils die beiden Teilkomponenten (in x- bzw. y-Richtung) anzugeben. Die Größen F_1 und F_2 (jeweils in N), α_a, α_s und α_e (jeweils in Grad) und l_1, l_2 und l_3 (jeweils in m) stehen in dieser Reihenfolge auf einem Lochband.

Die Ausgabe soll in Form einer Wertetabelle über einen Drucker realisiert werden.

(Die Skizze auf der nächsten Seite beachten.)



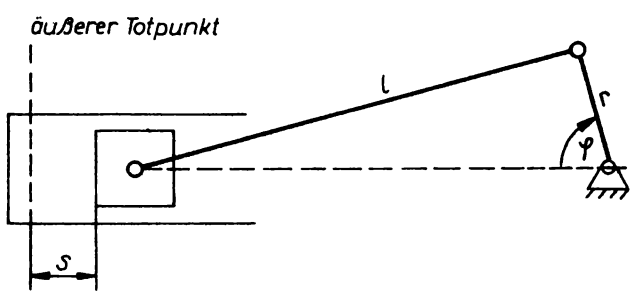
4.17. Stellen Sie einen PAP für die Berechnung des Abstandes s (in mm) eines Kolbens im Verbrennungsmotor vom äußeren Totpunkt nach der Formel

$$s = f(\varphi) = r\ [1 - \cos\varphi + \frac{1}{\lambda}\ (1 - \sqrt{1 - \lambda^2 \sin^2\varphi}\ )]$$

auf! Die Größen

r = Kurbelradius (in mm) und
l = Länge des Pleuels (in mm)

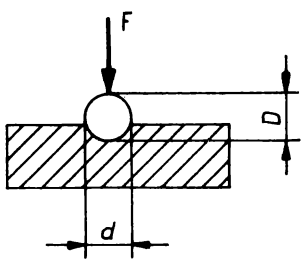
werden in dieser Reihenfolge über Lochbandleser 1 bereitgestellt. λ = r/l wird als Stangenverhältnis bezeichnet. Der Wert s ist jeweils für die Kurbelwinkel φ = 0°(5°)360° zu berechnen. Die Ausgabe soll in Form einer Wertetabelle über den Seriendrucker 1 realisiert werden.



4.18. In einem 3-dimensionalen Raum sind 2 Kräfte F_1 und F_2 (in N) gegeben. Beide Kräfte haben einen gemeinsamen Angriffspunkt.
Die Kraft F_1 bildet mit den Koordinatenachsen x, y und z die Winkel α_1, β_1 und γ_1. Die analogen Winkel für die Kraft F_2 seien α_2, β_2 und γ_2. Erstellen Sie einen PAP zur Berechnung der resultierenden Kraft F_R (in N) und ihrer Winkel α_R, β_R und γ_R. Alle Winkelangaben beziehen sich auf Grad und liegen im Bereich von 0° bis 90°.
Auf einem Lochband stehen die Größen F_1, α_1, β_1, γ_1, F_2, α_2, β_2 und γ_2 in dieser Reihenfolge. Die Ausgabe soll über Drucker erfolgen.

4.19. Auf einer geneigten Ebene mit dem Neigungswinkel α (in Grad) gleitet eine Last m (in kg). Erstellen Sie ein Programm zur Berechnung der Hangabtriebskraft F_H, der die Reibung erzeugenden Normalkraft F_N und der Reibungskraft F_R, wenn die Reibungszahl für die Gleitreibung μ ist!
Die Größe α, m und μ stehen in dieser Reihenfolge auf einem Lochband. Die Ausgabe ist über Drucker zu realisieren.

4.20. (*) Zur Bestimmung der Härte von Werkstoffen werden vorzugsweise die Verfahren nach Brinell, Vickers und Rockwell angewendet. Beim Kugeldruckversuch nach Brinell benutzt man eine gehärtete Stahlkugel, die mit einer bestimmten Kraft F (in N) in den zu prüfenden Werkstoff eingedrückt wird. Die Brinellhärte (HB) ist das Verhältnis der aufgewendeten Kraft F zur Fläche A der erzeugten Kugelhaube.



Es gilt:

$$HB = \frac{2F}{\pi D(D-\sqrt{D^2-d^2})}$$

Die Durchmesser d und D werden in mm angegeben.

Stellen Sie einen PAP zur Berechnung der Brinellhärte (arithmetischer Mittelwert) auf, wenn ein Versuch mehrmals wiederholt wurde. Für jeden durchgeführten Versuch stehen auf einem Lochband die Angaben F, D und d. Als Endekennzeichen wurde für F eine negative Zahl festgelegt. Über einen Drucker sind für jeden durchgeführten Versuch die Angaben F, D, d und HB zu protokollieren. Als letzte Angabe ist der Mittelwert für HB zu drucken.

4.21.1. In einem Materiallager eines Betriebes werden für jeden Artikel die Angaben
- Artikelnummer (AN)
- Menge (M) und
- Preis je Mengeneinheit (P)

erfaßt. Stellen Sie einen PAP zur Berechnung des Gesamtpreises (GP) jeder Artikelmenge auf. Das Druckbild soll in Tabellenform die drei Angaben AN, M und GP je Artikel enthalten. Die obigen drei Angaben (AN, M und P) bilden in dieser Reihenfolge für jeden Artikel einen Datensatz und liegen auf einem Lochband vor. Als Endekennzeichen ist eine negative Zahl für AN festgelegt worden.

4.21.2. Die Aufgabe 4.21.1. ist so zu erweitern, daß der Gesamtumsatz (GU) kumulativ zu berechnen und als letzte Zeile unter der Tabelle zu drucken ist.

4.21.3. Die Aufgabe 4.21.2. ist so zu verändern, daß als erste Spalte in der Tabelle die laufende Nummer steht.

5. Übungen zu kombinierten PAP

5.1. Stellen Sie einen PAP zur Berechnung und zum Druck einer Wertetabelle für die Funktion

$$y = f(x) = \frac{x}{x^2-9}$$

auf. x soll die Werte $x = x_a(x_s)x_e$ annehmen.
x_a, x_s und x_e sind in dieser Reihenfolge über Lochbandleser 1 einzugeben.

5.2. Stellen Sie einen PAP zur Berechnung und zum Druck einer Wertetabelle für die Funktion

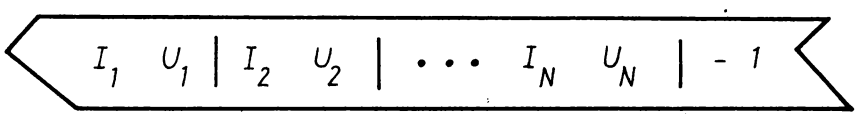
$$y = f(x) = \frac{1}{ax^2+bx+c}$$

im Bereich von [−2,5 ; 3,5] mit der Schrittweite $x_s = 0{,}1$ auf. Die Werte für a, b und c sind in dieser Reihenfolge über Lochbandleser 1 einzugeben.

5.3. Ein Körper befindet sich auf einer geneigten Ebene und gleitet auf ihr herab. Er überwindet dabei den Höhenunterschied h(in m).Stellen Sie einen PAP zur Berechnung einer Wertetabelle für die Endgeschwindigkeit v(in m/s)und die dazu benötigte Zeit t(in s)auf! Innerhalb der Wertetabelle soll sich der Bahnneigungswinkel α mit $\alpha = 40°(1°)50°$ ändern. Für jeden Winkelwert α ist die Reibungszahl μ mit den Werten $\mu = 0{,}5(0{,}05)0{,}8$ zu variieren. Der Höhenunterschied h ist über Bedienschreibmaschine abzufordern. Der Druck der Tabelle soll über Seriendrucker 1 erfolgen.

5.4. Ein Körper befindet sich auf einer geneigten Ebene und gleitet auf ihr herab. Er erreicht nach t Sekunden eine Geschwindigkeit v (in m/s). Erstellen Sie einen PAP zur Berechnung einer Wertetabelle für den Höhenunterschied h, der dabei überwunden wurde! h ist abhängig von der Gleitreibungszahl μ und vom Neigungswinkel α. Für jeden Wert μ ($\mu = 0{,}4(0{,}02)0{,}8$) ist der Neigungswinkel α mit den Werten $\alpha = \alpha_a(\alpha_s, \alpha_e)$ zu variieren v, t, α_a, α_s und α_e sind über Lochbandleser 1 einzugeben.

5.5. Für eine Widerstandsermittlung wurden mehrmals Strom und Spannung gemessen. Diese Daten wurden abwechselnd als ein Datensatz auf einem Lochband abgelocht, das über Lochbandleser 1 einzulesen ist.



Stellen Sie einen PAP zur Ermittlung des Mittelwertes des zu ermittelnden Widerstandes auf!
Dabei gelten folgende Bedingungen:
- Die gemessenen Ströme sind kleiner als 1 A.
- Die gemessenen Spannungen sind größer als 100 V.
- Genügt durch Meß- oder Lochfehler ein Wert nicht diesen Bedingungen, so ist das Wertepaar nicht in die Berechnung einzubeziehen.
- Das Ende der Meßwerte ist durch den Stromwert − 1 gekennzeichnet.

5.6. Bei der Gütekontrolle von elektrotechnischen Bauelementen erfolgt eine Sortierung in Abhängigkeit von der Abweichung zum Sollwert.
Stellen Sie einen PAP zur Ermittlung des Prozentsatzes der „guten Produktion" von Widerständen auf! Dabei gelten folgende Bedingungen:

- Als 1. Datensatz stehen auf einem Lochband (Eingabe über Lochbandleser 0) der Sollwert (in Ohm) und die zugelassene Abweichung (in %) bereit.
- Danach stehen auf diesem Lochband die Meßwerte der produzierten Widerstände (in Ohm), jeweils als ein Datensatz.
- Die Meßwertreihe wird mit einem negativen Wert abgeschlossen.
- Das Druckbild soll die Anzahl der gemessenen Bauelemente, die Anzahl der guten Widerstände und ihren Prozentsatz enthalten.

5.7. (*) Bei der Serienproduktion von Wellen werden mit Hilfe von Stichproben ständig der Wellendurchmesser d gemessen. Die gemessenen Durchmesser werden auf ein Lochband übertragen. Jede Zahl bildet einen Datensatz. Als Endekennzeichen wurde eine negative Zahl vereinbart. Erstellen Sie einen PAP zur Berechnung des arithmetischen Mittelwertes, des Minimums und des Maximums dieser Meßwerte. Das Druckbild soll außer diesen drei Angaben auch noch den Stichprobenumfang (Anzahl der Meßwerte) enthalten.

5.8. Auf einem Datenband sind beliebig viele Meßwerte gegeben. Stellen Sie einen PAP zur Berechnung des arithmetischen Mittelwertes und der absoluten und relativen Fehler für jeden Meßwert x_i auf. Die Meßwerte x_i bilden jeweils einen Datensatz. Die Meßwertreihe wird durch die Zahl 8888 abgeschlossen. Als Druckbild soll eine Tabelle mit nachstehenden Aufbau realisiert werden.

FEHLERBETRACHTUNGEN

MITTELWERT:

LFD. NR.	MESSWERT	ABS.–FEHLER	REL.–FEHLER

5.9.1. Die Werte der trigonometrischen Funktionen lassen sich mit Hilfe von Reihenentwicklungen berechnen. Für cos x gilt

$$\cos x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + - \ldots = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{x^{2k-1}}{(2k-1)!}$$

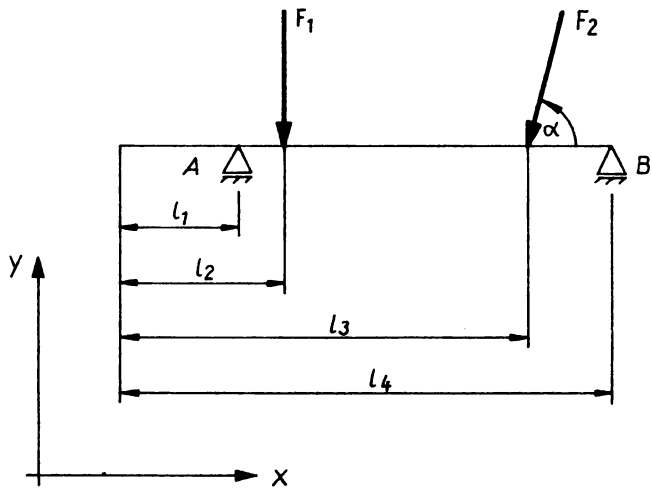
Stellen Sie einen PAP für Berechnung dieser Reihe mit einer Genauigkeit von $\epsilon = 10^{-4}$ auf! Der x-Wert ist über Bedienschreibmaschine einzugeben. Die Ausgabe hat ebenfalls über dieses Gerät zu erfolgen.

Hinweis: Der PAP der Übung 4.7. ist um die Berechnung der Fakultäten für jeden einzelnen Summanden zu ergänzen.

5.9.2. Die Abarbeitung der Reihe für cos x soll für n beliebige x-Werte realisiert werden. Auf einem Lochband steht als 1. Zahl die Größe n. Es folgen dann die n-Werte x_1, x_2 bis x_n. In Form einer Wertetabelle sollen über einen Drucker jeweils die Angaben x_i und cos x_i ausgegeben werden.

5.10. Die Aufgabenstellung der Aufgabe 4.17. ist so zu erweitern, daß der Abstand s = f(φ) mit φ = 0°(5°)360° für n verschiedene Wertepaare (r_i, l_i) berechnet wird. Auf einem Lochband stehen die Größen n, r_1, l_1, r_2, l_2, . . . , r_n, l_n in dieser Reihenfolge bereit.

5.11. Für das angegebene Belastungssystem eines Trägers auf zwei Stützen sind die Auflagerkräfte F_A und F_B (jeweils mit ihren Teilkomponenten in x- und y-Richtung) zu ermitteln.



Es sind hierbei die Angaben F_1 und l_2 zu variieren. Für jede Kraft $F_1 = F_a(F_s)F_e$ soll l_2 die Werte $l_2 = l_a(l_s)l_e$ annehmen. Die Kräfte sind in N gegeben und alle Längenangaben beziehen sich auf m. Auf einem Lochband stehen folgende Größen F_a, F_s, F_e, F_2, l_1, l_a, l_s, l_e, l_3 und l_4. Die Ausgabe soll in Form einer Wertetabelle über einen Drucker realisiert werden.

5.12. Bei einem schrägen Wurf läßt sich die Höhe y in Abhängigkeit von der Weite x nach der Formel

$$y = x \cdot \tan\alpha - \frac{g \cdot x^2}{2v_0^2 \cos^2\alpha}$$

berechnen. Stellen Sie einen PAP für die Berechnung der Bahnkurven für n verschiedene Anfangsgeschwindigkeiten v_0 (in m/s) auf. Die Weite x soll jeweils die Werte 1, 2, 3,... (in m) annehmen. Die Wertetabelle für ein v_0 ist abzubrechen, wenn $y \leqq 0$ wird. Auf einem Lochband stehen die Angaben α, n, $v_1, \ldots, v_n$.
Das Druckbild soll nachstehende Form erhalten:

Anfangsgeschwindigkeit	Weite	Höhe
xx.xx	xxx.xx	xxx.xx
	xxx.xx	xxx.xx
	.	
	.	
	.	
	xxx.xx	xxx.xx
xx.xx	xxx.xx	xxx.xx
	xxx.xx	xxx.xx
	.	
	.	
	.	

5.13. Für ein Fahrzeug mit der Masse m (in kg) wurden auf einer Versuchsstrecke folgende Angaben gemessen:

t_i (in s) und v_i (in km/h).

Die Zeitangabe t_i gibt die Zeit an, die benötigt wurde, um von der Geschwindigkeit v_{i-1} auf die Geschwindigkeit v_i zu beschleunigen bzw. abzubremsen. Ist $v_i = v_{i-1}$, so liegt für dieses Zeitintervall eine konstante Geschwindigkeit vor. Der Bewegungsablauf wird innerhalb der Zeitintervalle als gleichförmig angenommen.
Stellen Sie einen PAP für die Berechnung der Gesamtkraft (in kN) auf, die erforderlich ist, um den Bewegungsablauf unter Vernachlässigung der anderen Einflußgrößen zu realisieren. Es sollen weiterhin die maximale Beschleunigungs- und Bremskraft, die für ein Zeitintervall benötigt wurden, ausgedruckt werden.
Auf Lochband liegen folgende Informationen vor:
1. Datensatz: m
2. Datensatz: t_1 und v_1
3. Datensatz: t_2 und v_2

usw.
Ist $t_i = 0$, so ist die Meßreihe beendet.
Als Ausgagngsgeschwindigkeit ist $v_o = 0$ zu setzen.
Das Programm soll das nachstehende Druckbild realisieren:

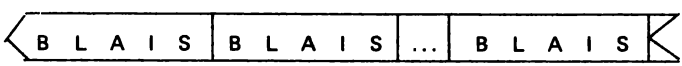
VERSUCHSAUSWERTUNG

AUSGANGSGESCHW. KM/H	ENDGESCHW. KM/H	ZEIT S	KRAFT KN
XXX.XX	XXX.XX	XXX.X	XX.XXX
XXX.XX	XXX.XX	XXX.X	XX.XXX
.	.	.	.
.	.	.	.
.	.	.	.

GESAMTKRAFT: XXX.XXX KN
MAX. BESCHL.–KRAFT: XXX.XXX KN
MAX. BREMS.–KRAFT: XXX.XXX KN

Die Druckzeilen in der Tabelle beziehen sich immer auf ein Zeitintervall.

5.14. Für die Beschäftigten einer Abteilung ist die durchschnittliche Normerfüllung und der Bruttolohn zu berechnen. Dazu liegen auf einem Lochband folgende Daten vor.



B = Beschäftigtennummer (B = 0 ist das Endekennzeichen)
L = Stundenlohnsatz laut Lohngruppe
A = Arbeitsstunden
I = Istproduktionseinheiten während der Arbeitszeit A
S = Sollproduktionseinheiten bezogen auf 43,75 Arbeitsstunden

Diese 5 Angaben bilden jeweils einen Datensatz.
Stellen Sie einen PAP zur Berechnung
– der Normerfüllung N des Beschäftigten in %
– des Lohnes G_B des Beschäftigten
– der Normerfüllung N_A der Abteilung und
– der Gesamtlohnkosten G_A für die Abteilung auf!

Das Druckbild soll folgende Form erhalten:

B	A	I	S	N	G_B
B	A	I	S	N	G_B
.					
.					
.					
B	A	I	S	N	G_B
				N_A	G_A

5.14.2 Die Aufgabe 5.14.1. ist so zu erweitern, daß sich der Lohn des Werktätigen aus einem Grundlohn und einem Mehrleistungslohn zusammensetzt. Der Grundlohn G_B berechnet sich wie in der Aufgabe 5.14.1. der Lohn. Der Mehrleistungslohn M_B ist von der Normerfüllung des Werktätigen abhängig. Es gilt

$N < 100\% \Rightarrow M_B = 0$

$N = 100\% \Rightarrow M_B = 5\%$ vom G_B

$100\% < N \leqq 105\% \Rightarrow M_B = 10\%$ vom G_B

$105\% < N \Rightarrow M_B = 20\%$ vom G_B

Im Druckbild sind die Lohnkosten getrennt nach Grundlohn, Mehrleistungslohn und Bruttolohn aufzuführen, so daß folgendes Druckbild entsteht

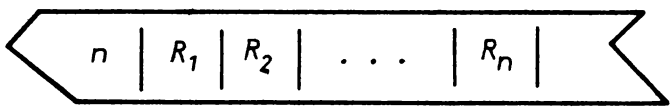
B	A	I	S	N	G_B	M_B	L_B
B	A	I	S	N	G_B	M_B	L_B
.							.
.							.
.							.
B	A	I	S	N	G_B	M_B	L_B
				N_A	G_A	M_A	L_A

6. Fehlersuche in Programmablaufplänen

Alle mathematischen Modelle und Programmablaufpläne der Aufgabengruppe 6 sind mit Hilfe eines Trockentestes auf ihre Richtigkeit zu prüfen. Evtl. erkannte Fehler sind zu korrigieren.

6.1. Auf einem LB stehen beliebige viele positive Meßwerte. Die Meßwertreihe wird durch den Zahlenwert −1 abgeschlossen. Durch den nachstehenden PAP soll der Wert des größten vorkommenden Meßwertes ermittelt werden.
(PAP bei Aufgabe 6.2. eingeordnet)

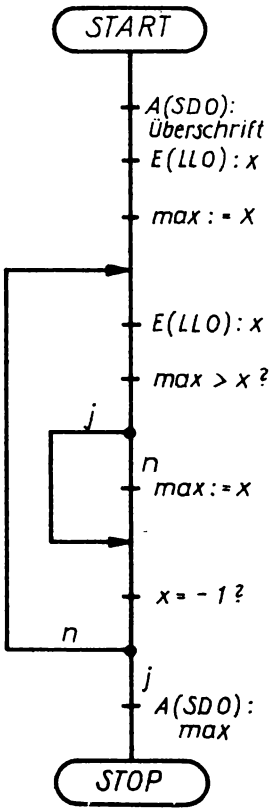
6.2. n gegebene ohmsche Widerstände R_1 bis R_n sind parallel geschaltet. Durch den nachstehenden PAP soll der Gesamtwiderstand ermittelt werden. Das gegebene Datenband hat folgenden Aufbau



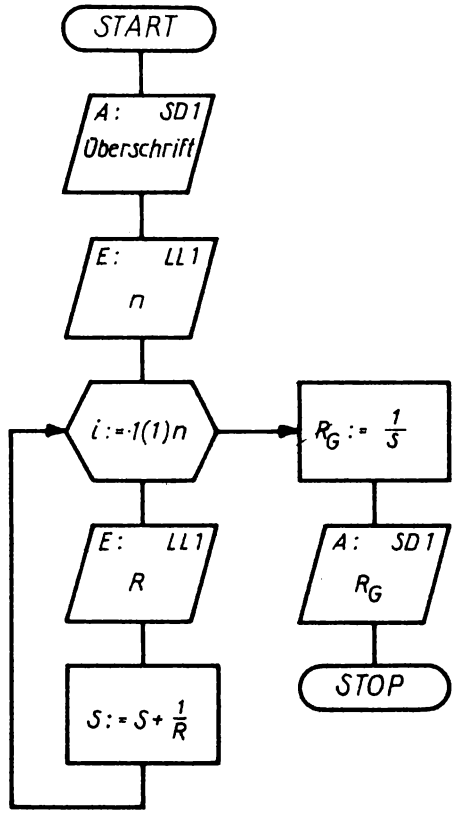
1. Mathematisches Modell

$$\frac{1}{R_G} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n} = \sum_{i=1}^{n} \frac{1}{R_i} \qquad [R_i] = \Omega$$

PAP für die Aufgabe 6.1.

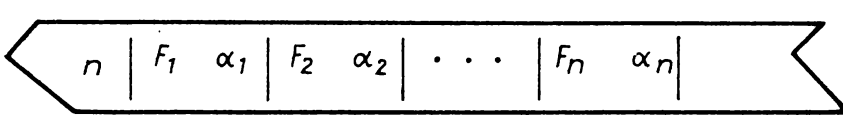


PAP für die Aufgabe 6.2.



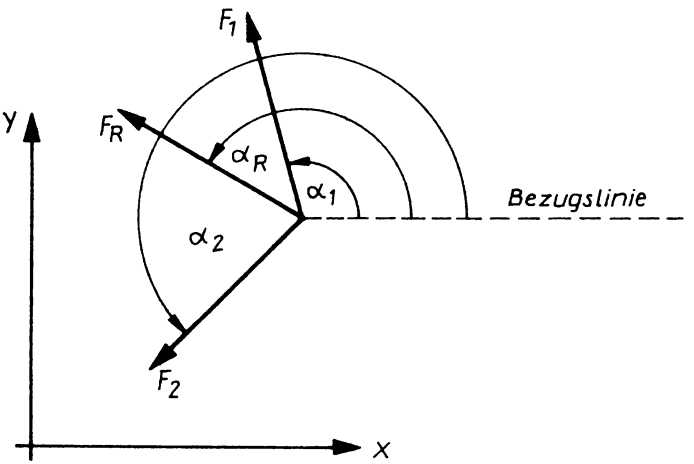
6.3. In einer Ebene sind n Kräfte $F_1, F_2, \ldots, F_n$ (in N) und ihre Richtungswinkel $\alpha_1, \alpha_2, \ldots, \alpha_n$ (in Grad) mit einer Bezugslinie gegeben. Die Kräfte besitzen einen gemeinsamen Angriffspunkt. Mit Hilfe des nachstehenden PAP soll die resultierende Kraft F_R und ihr Winkel mit der Bezugslinie berechnet werden.

Das Lochband hat folgenden Aufbau:



1. **Mathematisches Modell**

Skizze für n = 2



In der EDV sind alle Winkelangaben für Winkelfunktionen in Bogenmaß erforderlich.

$$\widehat{\alpha_i} = \frac{\alpha_i \cdot \pi}{180°}$$

$$F_R = \sqrt{F_{Rx}^2 + F_{Ry}^2} \qquad \text{mit}$$

$$F_{Rx} = \sum_{i=1}^{n} F_{ix} = \sum_{i=1}^{n} F_i \cdot \cos\widehat{\alpha_i} \quad \text{und} \quad F_{Ry} = \sum_{i=1}^{n} F_{iy} = \sum_{i=1}^{n} F_i \sin\widehat{\alpha_i}$$

$$\tan\alpha_R = \frac{F_{Ry}}{F_{Rx}} \qquad \text{falls} \quad F_{Rx} \neq 0$$

$$\alpha_R = \left(\arctan\frac{F_{Ry}}{F_{Rx}}\right) \cdot \frac{180°}{\pi} \qquad \text{mit} \quad -90° < \alpha_R < 90°$$

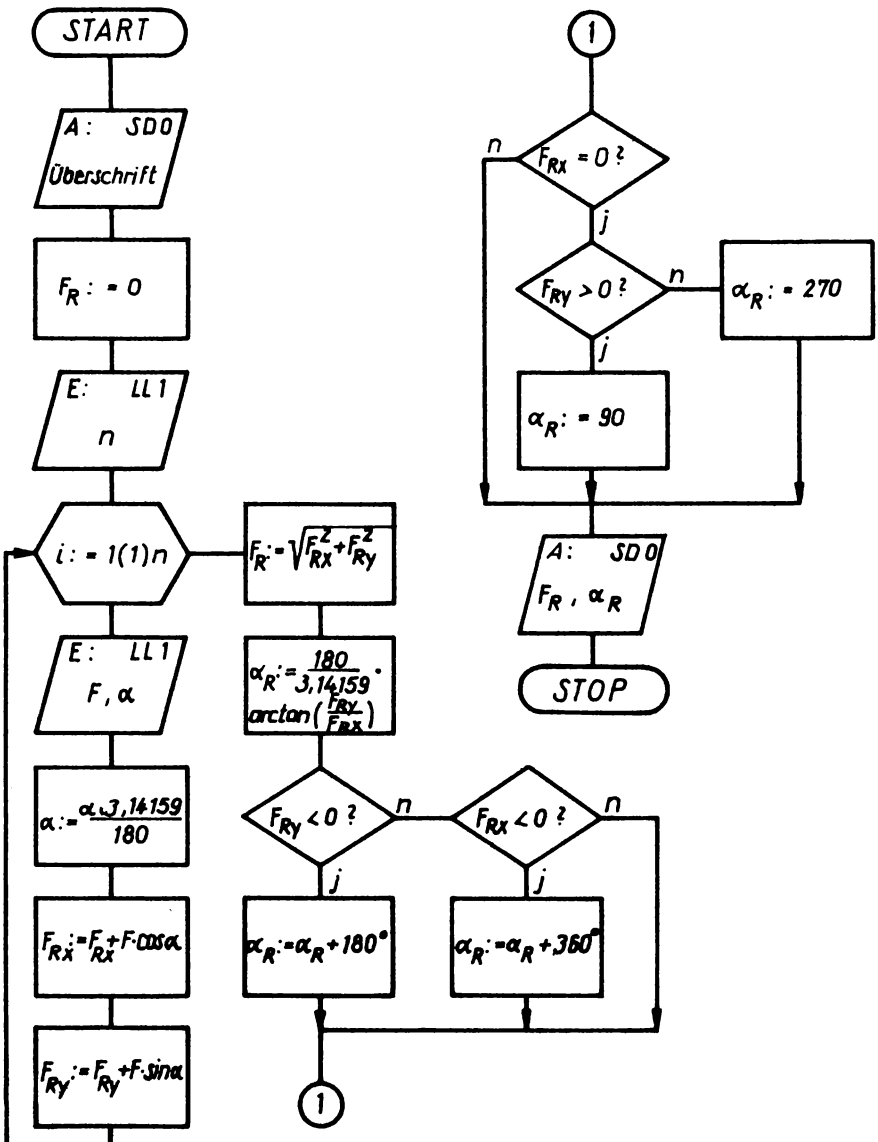
Quadrantenbetrachtungen für den Winkel α_R

F_{Rx}	F_{Ry}	α_R			
positiv	positiv	1. Quadrant			
positiv	negativ	4. Quadrant	$\Rightarrow \alpha_R :=$	α_R	$+360°$
negativ	positiv	2. Quadrant	$\Rightarrow \alpha_R :=$	α_R	$+180°$
negativ	negativ	3. Quadrant	$\Rightarrow \alpha_R :=$	α_R	$+180°$

Sonderfall: $F_{Rx} = 0$

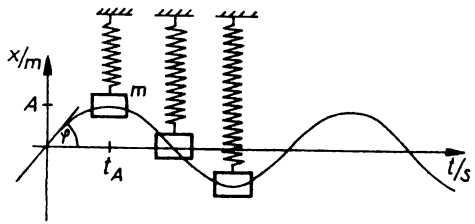
F_{Ry} positiv $\Rightarrow \alpha_R = 90°$

F_{Ry} negativ $\Rightarrow \alpha_R = 270°$


START
A: SD0
Überschrift
$F_R := 0$
E: LL1
n
i := 1(1)n
E: LL1
F, α
$\alpha := \frac{\alpha \cdot 3,14159}{180}$
$F_{Rx} := F_{Rx} + F \cdot \cos\alpha$
$F_{Ry} := F_{Ry} + F \cdot \sin\alpha$
$F_R := \sqrt{F_{Rx}^2 + F_{Ry}^2}$
$\alpha_R := \frac{180}{3,14159} \cdot \arctan\left(\frac{F_{Ry}}{F_{Rx}}\right)$
$F_{Ry} < 0$?
n
j
$\alpha_R := \alpha_R + 180°$
$F_{Rx} < 0$?
n
j
$\alpha_R := \alpha_R + 360°$
1
1
$F_{Rx} = 0$?
n
j
$F_{Ry} > 0$?
n
j
$\alpha_R := 270$
$\alpha_R := 90$
A: SD0
F_R, α_R
STOP

3.4. Eine an einer Feder befestigte Masse m (in kg) führt an ihr eine harmonische Schwingung ohne Dämpfung durch. Sie erreicht nach der Zeit t_A (in s) ihren Amplitudenwert A (in m). Mit einem PAP soll beginnend mit t = 0 s und der Schrittweite $t_A/20$ für eine volle Schwingung die Elongation x(in m), die potentielle Energie W_{pot}(in J) und die kinetische Energie W_{kin}(in J) berechnet werden. Die Federkonstante sei c(in N/m) und der Nullphasenwinkel φ (in Grad).

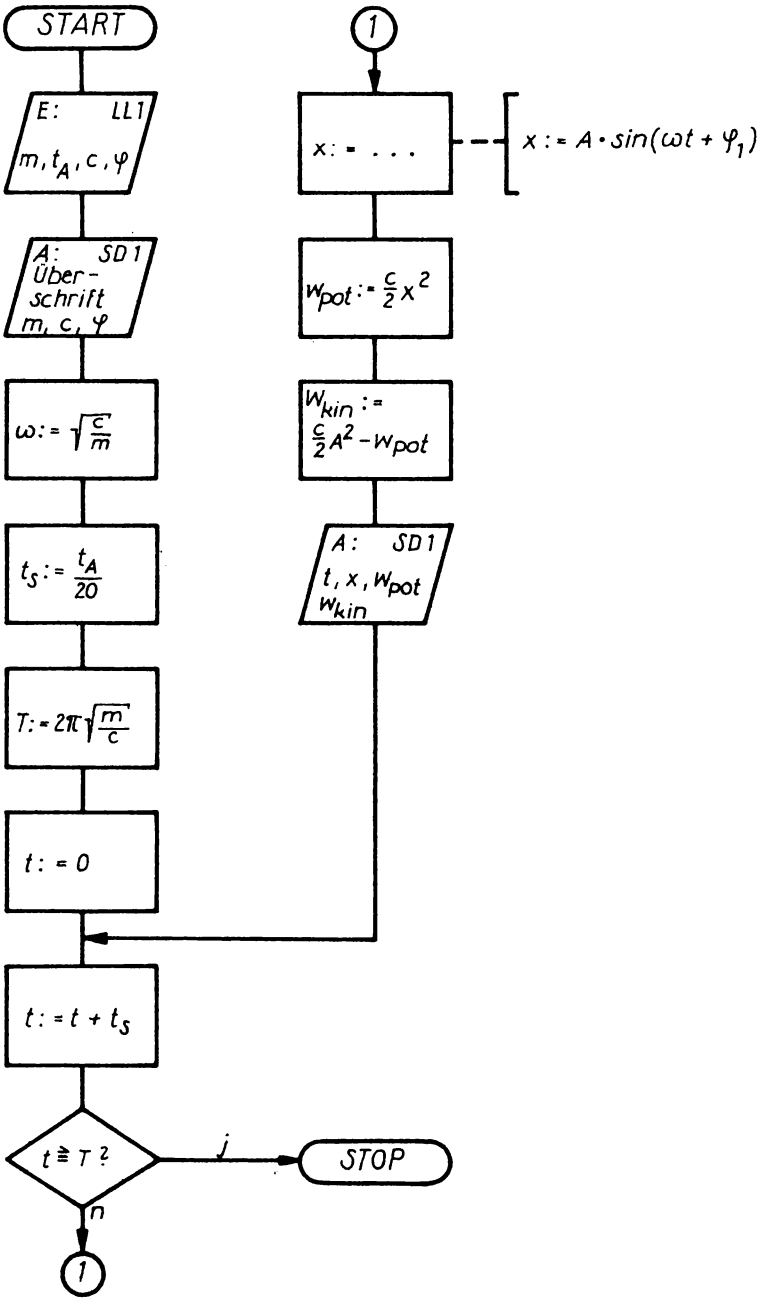
1. Mathematisches Modell



W_{pot} max W_{pot} 0 W_{pot} = max
W_{kin} = 0 W_{kin} = max W_{kin} = 0

Periodendauer	$T = 2\,\pi\sqrt{\frac{m}{c}}$	$[T] = s$
Elongation	$x = A \cdot \sin(\omega t + \varphi_1)$	$[x] = m$
	mit $\omega = \sqrt{\frac{c}{m}}$ und $\varphi_1 = \frac{\varphi}{57{,}3}$	$[\omega] = rad/s$
Potentielle Energie	$W_{pot} = \frac{c}{2}\,x^2$	$[W_{pot}] = J$
Kinetische Energie	$W_{pot} + W_{kin} = const$	$[W_{kin}] = J$
	$= W_{pot_{max}} = \frac{c}{2}\,A^2$	

$$\Rightarrow W_{kin} = W_{pot_{max}} - W_{pot}$$


START
E: LL1
m, t_A, c, φ
A: SD1
Über-
schrift
m, c, φ
ω := $\sqrt{\frac{c}{m}}$
t_S := $\frac{t_A}{20}$
T := $2\pi\sqrt{\frac{m}{c}}$
t := 0
t := t + t_S
t ≧ T ?
j
n
STOP
1
1
x := . . .
x := A · sin(ωt + φ_1)
W_{pot} := $\frac{c}{2}x^2$
W_{kin} := $\frac{c}{2}A^2 - W_{pot}$
A: SD1
t, x, W_{pot}
W_{kin}

6.5. Beim Einschalten eines Kondensators (R, C) in einem Gleichstromkreis mit der Spannung U_0 verändern sich Strom und Spannung in der Kapazität nach

$$i = \frac{U_0}{R} e^{\frac{-t}{\tau}}$$

und

$$u = U_0 (1 - e^{\frac{-t}{\tau}})$$

mit $\tau = R \cdot C$

Mit Hilfe eines PAP soll die Berechnung und der Druck einer Wertetabelle für die Funktionen $i = f(t)$ und $u = f(t)$ realisiert werden. Es gelten folgende Bedingungen:

- R (in Ω), C (in nF) und U_0 (in V) werden in dieser Reihenfolge über LL1 eingelesen.
- Die Berechnung soll abgebrochen werden, wenn der Funktionswert der Exponentialfunktion kleiner als 10^{-5} ist.
- Die Schrittweite der Zeit soll 10 ms betragen.
  Es ist mit t = 0 s zu beginnen.
- Druckbild

t in ms	i(t) in A	u(t) in V

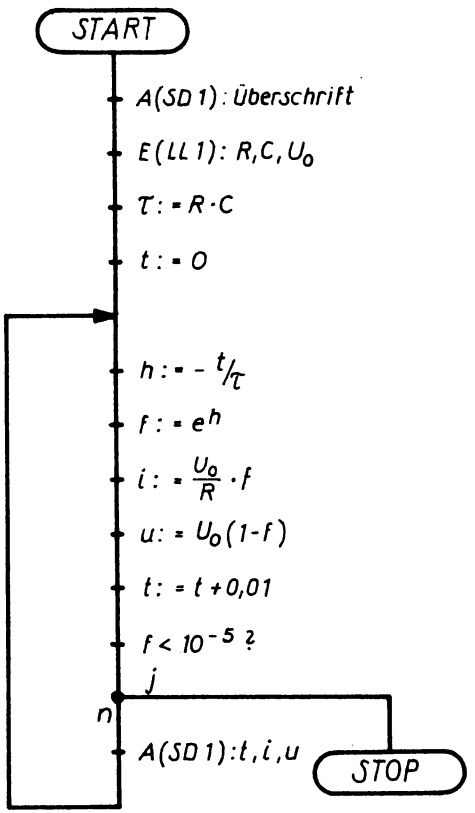
1. Mathematisches Modell:

$$\tau = R \cdot C \qquad [\tau] = \Omega \cdot F = \Omega \cdot \frac{s}{\Omega} = s$$

$$i = \frac{U_0}{R} e^{\frac{-t}{\tau}} \qquad [e^{\frac{-t}{\tau}}] = 1 \Rightarrow [i] = \frac{V}{\Omega} = A$$

$$u = U_0 (1 - e^{\frac{-t}{\tau}}) \qquad \Rightarrow [u] = V$$

2. PAP:


START
A(SD 1): Überschrift
E(LL 1): R,C,U0
τ: = R·C
t: = 0
h: = - t/τ
f: = e^h
i: = U0/R · f
u: = U0(1-f)
t: = t+0,01
f < 10^-5 ?
j
n
A(SD 1): t,i,u
STOP

7. Lösungen

Aufgabe 1.2.1. $s = 5, \quad x = 26; \quad a = 265{,}5$

Aufgabe 2.3.

Mathematisches Modell:

$O_H = 3\pi r^2 = 3 \cdot h$

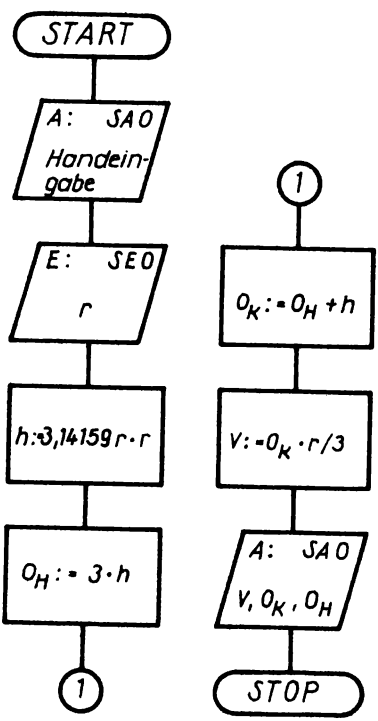
$O_K = 4\pi r^2 = O_H + \pi r^2 = O_H + h$

$V = \frac{4}{3}\pi r^3 = \frac{1}{3} \cdot O_K \cdot r$

$h = \pi r^2$

$[O_H] = cm^2; [O_K] = cm^2; [V] = cm^3$

PAP:



Aufgabe 2.6.

Mathematisches Modell:

$$c = \frac{b \cdot s^3 \cdot E}{6 \cdot l^3}$$

$$[c] = \frac{mm \cdot mm^3 \cdot N}{cm^3 \cdot m^2} = \frac{10^{-3}m \cdot (10^{-3}m)^3 \cdot N}{(10^{-2}m)^3 \quad m^2} = \frac{10^{-12}m^4N}{10^{-6}m^5} = \frac{N}{10^6 m}$$

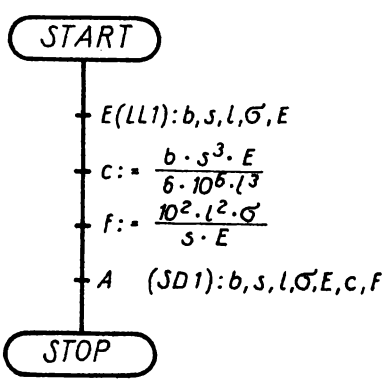
$$\Rightarrow c = \frac{b \cdot s^3 \cdot E}{10^6 \cdot 6 \cdot l^3}$$

$$f = \frac{l^2 \cdot \sigma_{zul}}{s \cdot E}$$

$$[f] = \frac{cm^2 \cdot N \cdot m^2}{mm \cdot N \cdot m^2} = 10^2 mm$$

$$\Rightarrow f = 10^2 \cdot \frac{l^2 \cdot \sigma_{zul}}{s \cdot E}$$

PAP:



Aufgabe 3.1.1.

Gewählte Eingangsdaten	Ergebnisse
L = 400; K = 0	L = 400; SV = 40, FZR = 0
L = 500; K = 1	L = 500; SV = 50; FZR = 0
L = 800; K = 0	L = 800; SV = 60; FZR = 0
L = 950; K = 1	L = 950; SV = 60; FZR = 35

Aufgabe 3.2.3.

Mathematisches Modell:

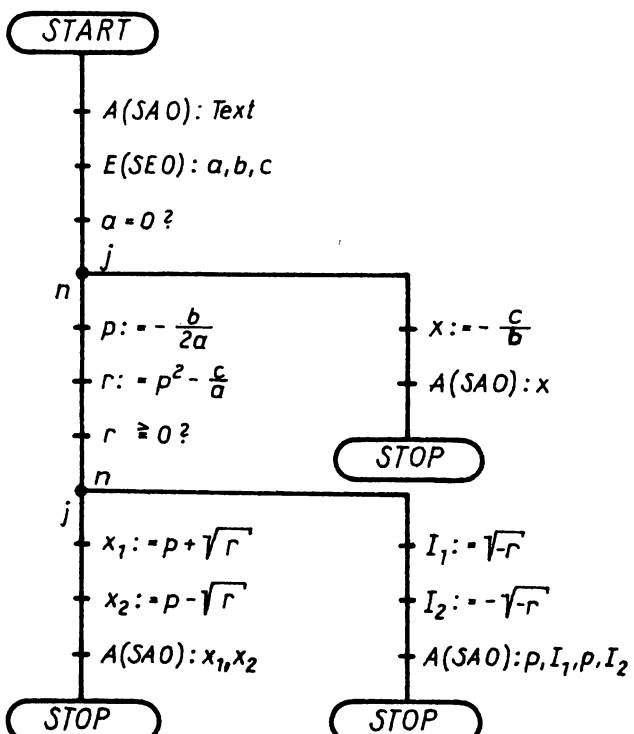
Wenn $a = 0$ gilt : $x = -\frac{c}{b}$

Wenn $a \neq 0$ gilt : $p = \frac{b}{a}$; $q = \frac{c}{a}$; $r = (\frac{p}{2})^2 - q$

$r \geq 0 \quad x_{1,2} = \frac{p}{2} \pm \sqrt{r}$

$r < 0 \quad x_{1,2} = \frac{p}{2} \pm i\sqrt{-r}$

PAP:



Aufgabe 3.4.

Mathematisches Modell:

$$x_1 = \cos\alpha = \frac{b^2 + c^2 - a^2}{2bc} \qquad x_2 = \cos\beta = \frac{c^2 + a^2 - b^2}{2ac}$$

$$x_3 = \cos\gamma = \frac{a^2 + b^2 - c^2}{2ab}$$

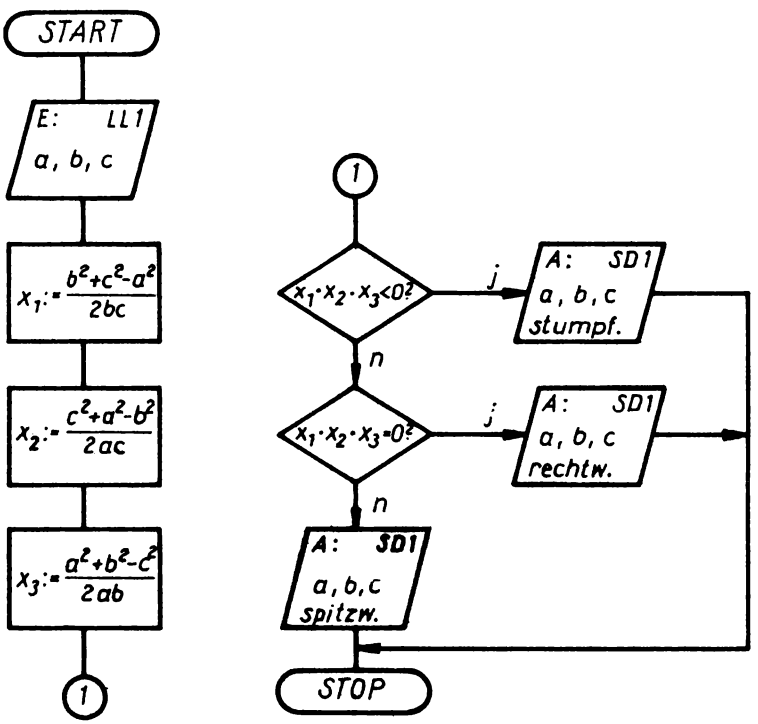
1. Möglichkeit:

Da $\cos 90° = 0$ gilt für ein rechtwinkliges Dreieck: $x_1 \cdot x_2 \cdot x_3 = 0$.

Da $\cos\delta < 0$ für $90° < \delta < 180°$ gilt für ein stumpfwinkliges Dreieck: $x_1 \cdot x_2 \cdot x_3 < 0$

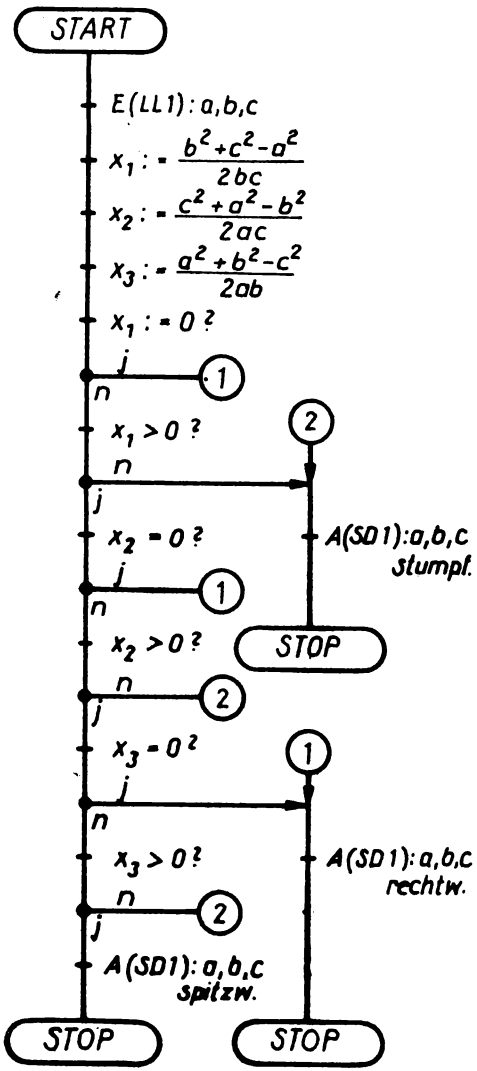
Für ein spitzwinkliges Dreieck gilt: $x_1 \cdot x_2 \cdot x_3 > 0$

PAP:



2. Möglichkeit

PAP:



Aufgabe 4.1.1.

Es wird s = 1,3698234 gedruckt.

Aufgabe 4.2.1.1.

Ergibtanweisung: i : = i + 2

Vergleich : i > 15 ?

oder i = 17 ?

Aufgabe 4.2.1.4.

Ergibtanweisung: i : = 1

Vergleich : i > 15 ?

oder i = 17 ?

Aufgabe 4.2.2.1

1. Ergibt anweisung: $x := x_a$
2. Ergibtanweisung: $x := x + x_s$

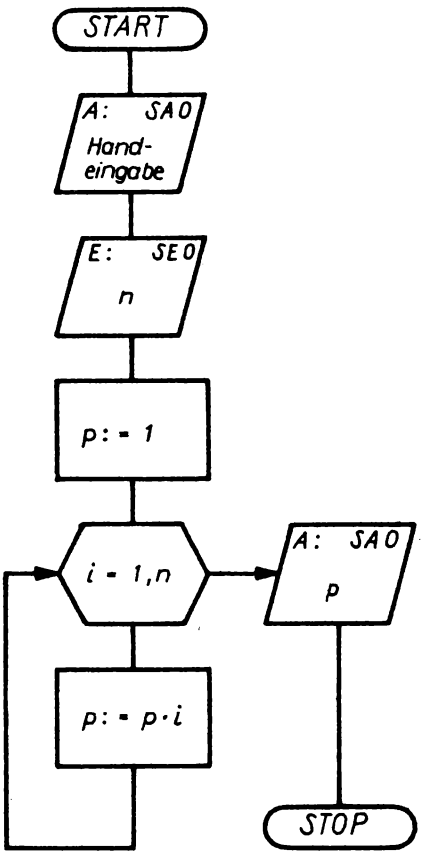
Vergleich : $x < x_e$?

Aufgabe 4.2.2.3.

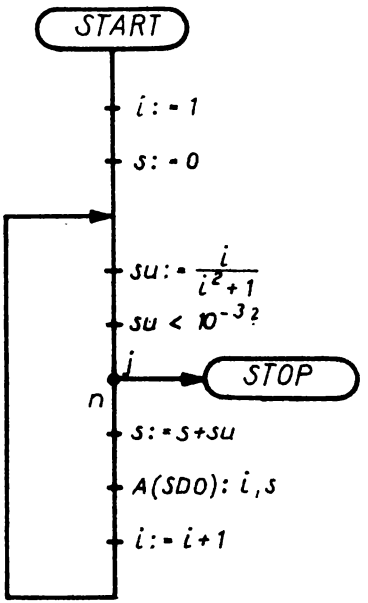
1. Ergibtanweisung: $x := x_a - x_s$
2. Ergibtanweisung: $x := x + x_s$

Vergleich : $x \geq x_e$?

Aufgabe 4.4.



Aufgabe 4.6.



Mathematisches Modell

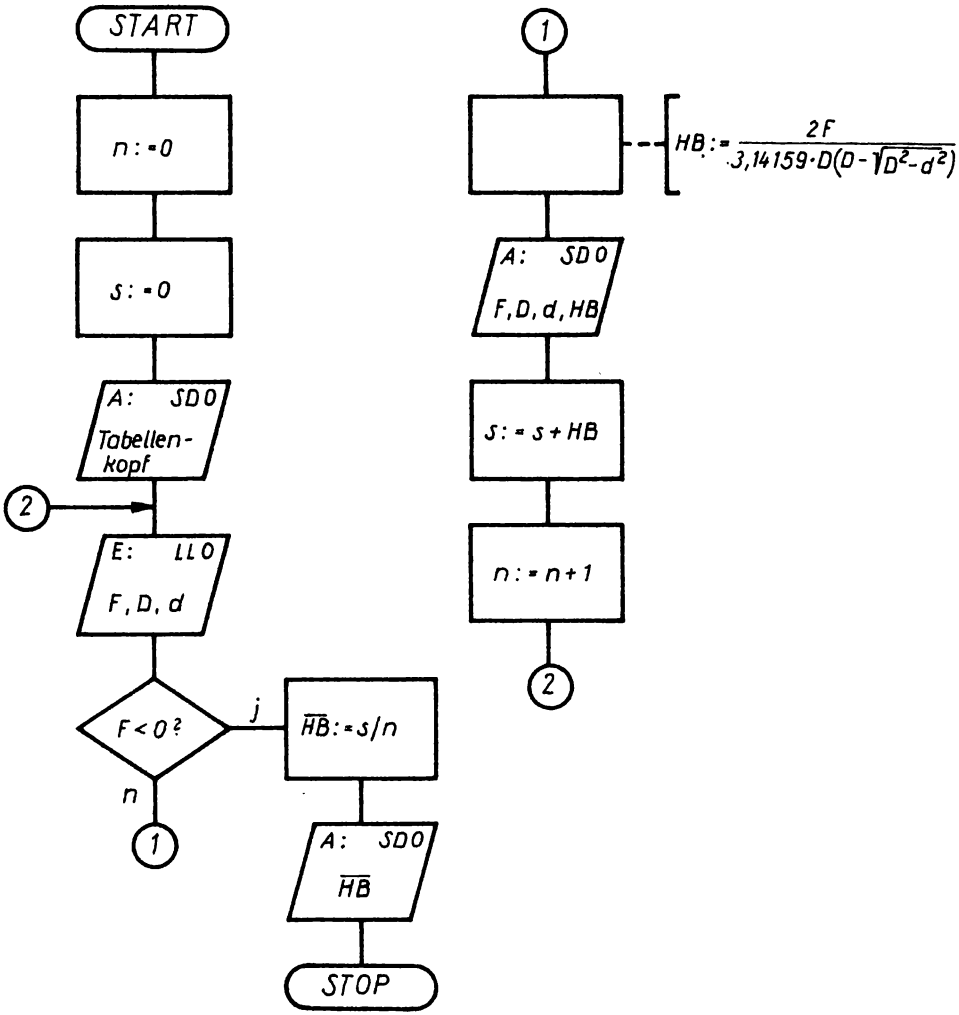
$$HB = \frac{2F}{\pi D(D - \sqrt{D^2 - d^2})} \qquad [F] = N \quad [D] = [d] = mm \quad [HB] = N/mm^2$$

$$\overline{HB} = \frac{1}{n} \sum_{i=1}^{n} HB_i$$

Anzahl n ist durch zählen der Durchläufe zu ermitteln.

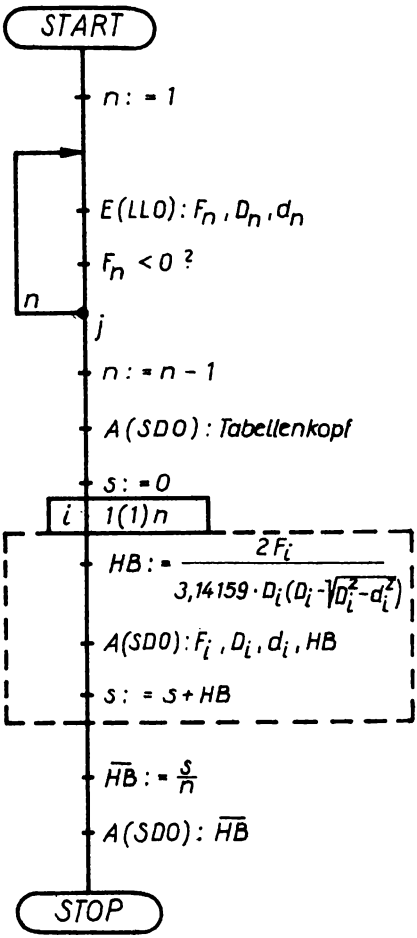
PAP:

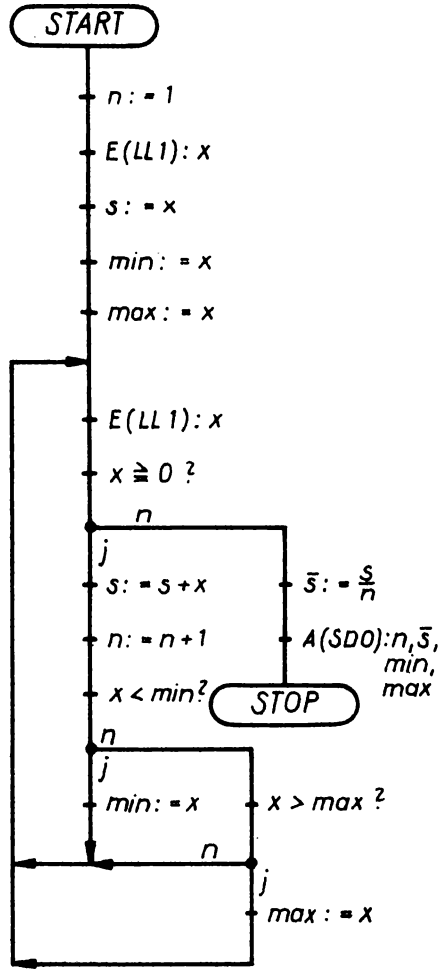
1. Möglichkeit (ohne Verwendung eines Feldes)



2. Möglichkeit (Verwendung von Feldern):

Aufgabe 5.7.





FACHSCHUL-
STUDIUM

FSD Dipl.-Phys. Werner Seifert

GRUNDLAGEN DER INFORMATIONSVERARBEITUNG

7

BASIC
für
Kleincomputer

Herausgeber:
Institut für Fachschulwesen der
Deutschen Demokratischen Republik
Karl-Marx-Stadt

03 0160 07 0

Dieser Lehrbrief wurde
verfaßt und bearbeitet von:

FSD Dipl.-Phys. Werner Seifert
Institut für Fachschulwesen der DDR, Karl-Marx-Stadt

lektoriert von:

SD Dipl.-Ing. Werner Sauerzapf
Ingenieurschule für Bauwesen, Leipzig

Redaktionsschluß: 30. 1. 86

Als Manuskript gedruckt · Alle Rechte vorbehalten
Veröffentlicht:
INSTITUT FÜR FACHSCHULWESEN DER
DEUTSCHEN DEMOKRATISCHEN REPUBLIK
Karl-Marx-Stadt

Druck und buchbinderische Verarbeitung:

ZENTRALSTELLE FÜR LEHR- UND ORGANISATIONSMITTEL DES
MINISTERIUMS FÜR HOCH- UND FACHSCHULWESEN, ZWICKAU
Ag 613/466/86/25.000 1. Ausgabe 1. Auflage
Vorzugsschutzgebühr: 2,00 M

Inhaltsverzeichnis

1. Einführung

1.1. Vorbemerkungen

Der vorliegende Lehrbrief hat das Ziel, Sie mit der Sprache BASIC für Kleincomputer vertraut zu machen. Nach dem Studium des Lehrbriefes sollen Sie eigene Programme entwickeln und mit Hilfe der Kleincomputer abarbeiten können.

Bei den Kleincomputern, die mit dem hier beschriebenen BASIC arbeiten, handelt es sich um zwei unterschiedliche Geräte. Die Unterschiede sind einesteils äußerlich in der Tastatur erkennbar und anderenteils durch bestimmte Möglichkeiten gekennzeichnet, die die Geräte besitzen. Besonders ausgeprägt sind diese bei grafischen Darstellungen auf dem Bildschirm.
Die Übereinstimmung bei der Nutzung des Kleincomputer-BASIC ist trotzdem sehr weitgehend. Auf Besonderheiten wird im Lehrbrief an entsprechender Stelle hingewiesen.

Von den beiden Kleincomputern – KC 85/1 und KC 85/2 – wird der erstere häufig auch (u. a. im folgenden Text) als Z 9001 bezeichnet.

BASIC ist eine relativ einfache Programmiersprache. Die Worte dieser Sprache stammen aus dem Englischen. Das Kleincomputer-BASIC ist einer von vielen Dialekten und hat einen relativ begrenzten Umfang. Es ist nicht identisch mit dem Bürocomputer-BASIC oder dem MC 80-BASIC. Für diese Dialekte gibt es eigenes Lehrmaterial.

Bevor Sie beginnen, mit der Sprache BASIC zu arbeiten, d. h. versuchen, Programme zu schreiben, müssen Vorüberlegungen und Vorarbeiten erfolgen.
Zuerst ist das zu bearbeitende Problem exakt zu formulieren. Danach wird ein Entwurf für die Struktur des Programms gemacht und der Algorithmus der Problemlösung erarbeitet. Dazu gehören die Aufstellung eines Ablaufplanes und die mathematische Formulierung des Problems. Nun kann die Beschreibung in der Programmiersprache erfolgen.

Durch BASIC wird eine Dialogarbeitsweise möglich. Sie können das zu erstellende Programm direkt am Rechner erarbeiten. Während der Erarbeitung und Änderung des Programms steuern Sie den Dialog mit Aktionen, die Sie überwiegend durch Kommandos und Eingaben auslösen (nutzergesteuerter Dialog). Deshalb ist es unbedingt notwendig, daß Sie vorher einen Ablaufplan für das Programm erarbeitet haben.
Während der Programmabarbeitung steuert der Rechner den Dialog, indem er von Ihnen Eingaben fordert (programmgesteuerter Dialog).

1.2. Sprachaufbau und Zeichenvorrat

Die Verständigung mit dem Computer erfolgt durch eine Sprache. In unserem Fall mit der Sprache BASIC. Die Sprache dient dazu, Kommandos einzugeben – diese werden sofort ausgeführt – oder Programme zu formulieren, die nach Bereitstellung der benötigten Daten auf dem Computer abgearbeitet werden können. Anders ausgedrückt, die Lösung des in BASIC formulierten Problems kann durch den Computer erarbeitet werden.

Die Buchstaben A, B, C. . . . X, Y, Z (beim Z 9001 auch Kleinbuchstaben);
die Ziffern Ø, 1, 2, . . . 9;
die arithmetischen Operatoren +, –, *, / und ^ (Potenzzeichen);
die Vergleichsoperatoren <, =, >;
die logischen Operatoren AND, OR, NOT und
die Sonderzeichen (z. B. Komma, Hochkomma, Semikolon, Klammern)
bilden die in BASIC zulässigen **Zeichen** oder den Zeichenvorrat, aus dem Worte und Ausdrücke gebildet werden können.

Wie jede Sprache, enthält auch BASIC Worte, die mit weiteren Ausdrücken zu Sätzen oder Anweisungen zusammengesetzt werden können. Ein Programm besteht aus mehreren Anweisungen. Die Worte werden aus den Buchstaben des Alphabets gebildet. Worte, die in BASIC eine Bedeutung haben, sind z. B. INPUT, PRINT u. a. Sie werden nur durch große Buchstaben dargestellt und als **Schlüsselworte** bezeichnet.

Die Regeln, nach denen die Ausdrücke und Anweisungen gebildet werden, nennt man die Syntax der Sprache.

2. Ablauf einer Terminalsitzung

Nach dem Einschalten des Fernsehgerätes und des Kleincomputers meldet sich das Betriebssystem des Computers.
Da wir die problemorientierte Sprache BASIC anwenden wollen, ist es notwendig, den Interpreter dieser Sprache in den Speicher zu bringen[1] oder – wie man auch sagt – zu laden, Dazu müssen wir den Cursor auf LOAD stellen bzw. beim Z 9001 das Wort BASIC eingeben und das Kassettengerät vorbereiten, indem wir die Kassette mit dem BASIC-Interpreter einlegen. Nun startet man den Kassettenrecorder mit der Wiedergabetaste.
War alles richtig vorbereitet, ertönt bald ein Pfeifton, während dessen die ENTER-Taste des Computers betätigt wird. Danach liest der Computer den Inhalt des Interpreters ein. Auf dem Bildschirm äußert sich das durch eine Zahlenfolge bzw. durch das Vorrücken des Cursors. Dadurch werden die einzelnen Teile (Records) gekennzeichnet und dokumentiert, daß diese Teile richtig eingelesen wurden.

Nach Beendigung des gesamten Ladevorgangs meldet sich der Computer selbständig mit:

HC-BASIC MEMORY END? : bzw. MEMORY SIZE? : (beim Z 9001)

Betätigen wir nochmals die ENTER-Taste, wird die Zahl der freien Speicherplätze angezeigt, und durch OK meldet der Computer, daß er bereit ist, Eingaben entgegenzunehmen. Er kann nun mit der Sprache BASIC im Direkt- oder Programmodus arbeiten.

So wie der Interpreter eingelesen wurde, kann man auch Programme von Kasette einlesen und diese wieder zurück auf Magnetband speichern.

Um ein Programm einzulesen, wird

CLOAD "name"

eingegeben, das Kassettengerät auf Wiedergabe gestellt und beim Pfeifton die ENTER-Taste betätigt. Die Kassette sollte an der Stelle stehen, bei der das Abspeichern des gewünschten Programms begann.

Beim Auslagern wird

CSAVE "name"

eingegeben, das Kassettengerät auf Aufnahme gestellt und die ENTER-Taste betätigt.
Für name wird eine frei wählbare Zeichenfolge von 1 bis 8 Zeichen eingegeben.

---

1) Das Einlesen ist nur erforderlich, wenn kein Steckmodul (ROM) mit dem Interpreter zur Verfügung steht.

3. Der Kleincomputer im Direktmodus als Tischrechner

3.1. Arithmetische Ausdrücke

Sie können den Kleincomputer wie einen Tischrechner benutzen, denn es ist nach dem Einspeichern des BASIC-Interpreters sofort möglich, arithmetische Ausdrücke zu berechnen.
Nach Eingabe des Fragezeichens kann der zu berechnende arithmetische Ausdruck eingegeben werden.

Beispiele:

```
? 3 * 15 + 6
? 3 * (15 + 6)
? 1Ø3.7 – 1.27 * 65.3
? .65/1ØØ
? 1.6/.82 * 4/15.1
```

Hat man sich bei der Eingabe verschrieben, so kann man den Cursor auf das falsche Zeichen setzen und es überschreiben. Das Ergebnis der Ausdrücke wird berechnet und angezeigt, wenn man die ENTER-Taste betätigt. Es erscheint in der folgenden Zeile und der Computer meldet mit OK, daß er zu neuen Eingaben bereit ist.

Der arithmetische Ausdruck kann auch nach PRINT stehen, wie es in Programmen üblich ist.

Beispiele:

```
PRINT 7/24 + 5/24
PRINT (.125 + 81) * (8 ∧ 1.7)
PRINT 4.33E6 + 1ØØØ
PRINT .6E – 2/5
```

Wie Sie an der Schreibweise der Beispiele erkannt haben, sind einige Besonderheiten zu beachten, die im folgenden zusammengestellt sind.

Zusammenfassung:

- Das sonst übliche Dezimalkomma ist ein Punkt.
  3.213
- Die Null wird im Unterschied zum Buchstaben O durchgestrichen.
  1ØØ7.Ø2
- Dezimalzahlen < 1 können ohne Vornullen geschrieben werden.
  .47
- Die Multiplikation wird durch * ausgedrückt.
  2.7 * (8.9 – 11.6)
- Die Division wird durch / ausgedrückt, damit alles auf eine Zeile geschrieben werden kann.
  5.5/1.83
- Das Potenzzeichen ist ∧ . Der Exponent folgt danach auf gleicher Zeile und kann beliebig sein.
  .638 ∧ –2.9
- Zehnerpotenzen werden ohne angehobenen Exponenten geschrieben. Er folgt nach einem E und muß ganzzahlig sein.
  6E3, 1.2E – 5
- Zu multiplizierende Ausdrücke sind in Klammern zu setzen.
  (3.1 + 15.7) * (7.Ø2 – 4.9Ø)
- Ist der Nenner eines Bruches selbst ein Ausdruck, so ist er in Klammern zu setzen.
  4.61/(2 – 7 * 3.12)

– BASIC verarbeitet reelle Zahlen im Bereich von $9.5E-39 < |Z| < 1.7E38$ und Null.

Das positive Vorzeichen kann entfallen. Das negative Vorzeichen muß vor der Zahl stehen.
Zur Darstellung einer Zahl werden maximal 6 Ziffern verwendet. Weitergehende Eingaben werden intern gerundet.

Übung:

Ü 1. Formen Sie folgende arithmetische Ausdrücke in die BASIC-Schreibweise um, und kontrollieren Sie, ob der von Ihnen eingegebene Ausdruck das richtige Ergebnis bringt.

$$\frac{345{,}7 - 6{,}225 \cdot 0{,}73}{7{,}285} = 46{,}8299$$

$$\frac{1}{\frac{1}{5} - \frac{1}{0{,}2} + 3{,}1} = -0{,}588235$$

$$\left(\frac{17{,}35}{37} + \frac{18.53}{43}\right) \cdot 35 = 31{,}4947$$

$$\frac{90807{,}2}{312{,}4 \cdot (-17{,}6)} = -16{,}5157$$

3.2. Mathematische Funktionen in arithmetischen Ausdrücken

Im Kleincomputer-BASIC gibt es die folgenden mathematischen Funktionen, die der Computer unmittelbar berechnen kann.

ABS(X)	Betrag von x
ATN(X)	arctan x, Ergebnis im Bogenmaß
COS(X)	cos x, x im Bogenmaß
EXP(X)	e^x, $x < 87{,}3$
INT(X)	größte ganze Zahl $\leq x$
LN(X)	natürlicher Logarithmus von x, $x > 0$
RND(X)	erzeugt Zufallszahl zwischen 0 und 1
SGN(X)	Signumfunktion SGN(X) = –1 für $x < 0$ SGN(X) = 0 für $x = 0$ SGN(X) = 1 für $x = 0$
SIN(X)	sin x, x im Bogenmaß
SQR(X)	$\sqrt{x}$, $x \geq 0$
TAN(X)	tan x, x im Bogenmaß
und die Konstante	
PI	$\pi = 3{,}14159$

Damit können weitere arithmetische Ausdrücke gebildet werden.

Beispiele:

```
3 * SIN (PI * 3Ø/18Ø)
1.2 * (.65 + EXP (-.5))
.33 * LN (1 + 2.Ø3)
PRINT 1 + ATN (.5)
SQR (2)/2
2 * SQR (5)
SQR (ABS (1Ø - 23))
PRINT 2 + TAN (15/2)
```

Lassen Sie sich die Ergebnisse durch PRINT, wie in der letzten Zeile angegeben, auf dem Bildschirm ausgeben.

Übungen:

Ü 2. Berechnen Sie mit dem Computer

$\sin\alpha$, $\cos\alpha$, $\tan\alpha$, mit $\alpha = 60°$

Entwickeln Sie dazu einen Ausdruck, der den Winkel in das Bogenmaß umwandelt, und schreiben Sie diesen Ausdruck als Argument hinter die Winkelfunktionen.
Kontrollieren Sie das Ergebnis mit einem Tafelwerk.

Ü 3. Formulieren Sie folgende Ausdrücke in BASIC unter Verwendung der Funktionen und lassen sich das Ergebnis ausgeben.

$\sqrt{1-\sin^2(\frac{\pi}{4})}$ $\qquad$ $\frac{4}{3}\pi 2{,}7^3$

$|0{,}13^2-4{,}81|$ $\qquad$ $1-e^{-0{,}52}$ $\qquad$ $\arctan 1$

Die Darstellung spezieller Ausdrucke und Funktionen, die in BASIC nicht definiert sind, geschieht durch mathematische Umformungen, die zum Teil auch im Handbuch zu den Kleincomputern aufgeführt sind.

$\sqrt[3]{x-y}$	(X - Y) Λ (1/3) =
	EXP (1/3*LN (X - Y))
lg x	LN (X) / LN (1Ø)

u. ä.

Zusammenfassung:

Ein arithmetischer Ausdruck setzt sich aus Operanden (Zahlen, Variablen, Funktionen und Klammern) zusammen, die durch Operatoren (+, -, *, / und Λ)verbunden werden. Die Abarbeitung geschieht nach den Regeln der Mathematik.

4. Zuweisungen

4.1. Variable

In allen arithmetischen Ausdrücken können auch allgemeine Zahlensymbole verwendet werden, die in BASIC durch Großbuchstaben dargestellt und wie in der Mathematik als Variable bezeichnet werden.

Damit die Anzahl der möglichen Variablen nicht durch die Zahl der Buchstaben begrenzt wird, können 2 Buchstaben oder ein Buchstabe und eine Ziffer zur Bezeichnung benutzt werden. Die Variablenbezeichnung beginnt immer mit einem Buchstaben. Somit sind folgende Bezeichnungen oder Namen möglich:

A, B, C, . . . Z
AA, AB, AC, AZ
BA, BB, BC, BZ
.
.
.
ZA, ZB, ZC, ZZ
A0, A1, A2, A9
B0, B1, B2, B9
.
.
.
Z0, Z1, Z2, Z9

Es können auch Worte als Variablenbezeichnung verwendet werden, wie WURZEL, MITTELWERT usf. Der Computer nutzt aber nur die beiden ersten Buchstaben. Er kann also z. B. MITTELWERT nicht von MINIMUM unterscheiden.

Variablennamen, die in BASIC als Worte anderweitig schon belegt sind, können nicht verwendet werden. Für Variablennamen aus 2 Buchstaben sind die folgenden Kombinationen unzulässig:

AT, FN, IF, LN, ON, OR, PI und TO

Beispiele:

Arithmetische Ausdrücke mit Variablen

mathematische Schreibweise	BASIC-Schreibweise
$a - b$	A − B
$2c$	2 * C
$5cd$	5 * C * D
b^2	B * B
$(n^2 + m)^5$	(N * N + M) ∧ 5
$1 + n^{1,2}$	1 + N ∧ 1.2
$k + lm$	K + L * M
$(x + y)z$	(X + Y) * Z
$\frac{x}{yz}$	X/(Y * Z) = X/Y/Z
$\frac{x + y}{xy}$	(X + Y) / (X * Y)

$$\frac{a+b}{(a+c)(a-d)}$$ (A + B) / ((A+C) * (A–D))

Übung:

Ü 4. Formulieren Sie folgende Ausdrücke in die BASIC-Schreibweise um:

$k^2 + \frac{m}{n}$

$\frac{2a+b-c}{c-d}$

$\frac{a+b}{a-b}$

$\frac{6(x-5)}{(x+5)(x-3)}$

$(x+1)^2$

$\frac{xy-y^2}{z-1}$

$\frac{4}{3}\pi r^3$

$a \sin(wt+\alpha)$

$U_0(1-e^{-t})$

$|a^2-b|$

$\ln x + 1$

$\sqrt{\frac{p^2}{4}-q}$

$\sqrt{1-\sin^2 y}$

4.2. Ergibtanweisungen

Jeder Variablen muß vor ihrer Benutzung in einem Programm ein konkreter Zahlenwert zugewiesen werden. Dies geschieht durch die Ergibtanweisung:

LET variable = ausdruck

Das LET kann entfallen. Wir handhaben dies künftig so!

Beispiele:

```
LET A = 3
B = 12.67
B3 = – 3.2E2*7.1
BC = .32/4.3
```

Die Ergibtanweisungen sind wie folgt zu lesen:

A ergibt sich zu 3
B ergibt sich zu 12.67
B3 ergibt sich aus . . .

Durch die Ergibtanweisung wird der links stehenden Variablen das aus dem rechts stehenden Ausdruck ermittelte Ergebnis zugewiesen. Unter dem Namen der Variablen wird das errechnete Ergebnis gleichzeitig gespeichert und steht uns ständig zur Verfügung. Der Rechner legt automatisch eine Variablen-

liste an, und wir können die aktuelle Belegung abfragen; z. B. den Wert von A durch PRINTA.
Im rechts vom Gleichheitszeichen stehenden Ausdruck können weitere Variable enthalten sein, die schon eine Zahl zugewiesen bekamen.

Beispiele:

```
G = (H  + .87) * SQR(B)          N = SIN(3)
I = K + 1                        B = PI
P = ((3+A) ^ 2) / (1-B) * SQR(C) L = (5 + B) / C
S = X *X + Y * Y - 2 * X * Y     WU = EXP (1/N * LN(A))
```

Da es Programmteile gibt, die mehrfach durchlaufen werden, sind folgende Ergibtanweisungen möglich und sinnvoll:

```
B = A + B        K = K * K        M = M * N
I = I + 1        M = M/2
```

Es tritt dabei die gleiche Variable rechts und links des Gleichheitszeichens auf. Eine solche Formulierung widerspricht der Schreibweise einer mathematischen Gleichung. Mit der Ergibtanweisung wird eine Operation beschrieben, bei der der rechts stehende arithmetische Ausdruck – mit Hilfe der Werte der Variablen – berechnet und das Ergebnis der links stehenden Variablen zugewiesen wird. Dadurch wird der alte Wert der Variablen überschrieben und geht verloren.

Beispiel:

Die Anweisung I = I + 1 soll dreimal nacheinander abgearbeitet werden. Die Anfangsbelegung ist I = Ø

linke Seite	rechte Seite	
1	Ø + 1 = 1	1. Abarbeitung
2	1 + 1 = 2	2. Abarbeitung
	2 + 1 = 3	3. Abarbeitung

Die Variable I zählt damit die Anzahl der Abarbeitungen dieser Ergibtanweisung.

Übung:

Ü 5. Welcher Wert ergibt sich für K aus K = K * K, wenn anfangs K = 2 ist und die Ergibtanweisung dreimal abgearbeitet wird.

4.3. Zuweisung durch Tastatureingabe

Wie im vorangegangenen Abschnitt erläutert, können Zahlen durch Ergibtanweisungen Variablen zugewiesen werden. Aber auch durch Eingaben über die Tastatur ist eine Zuweisung an Variable möglich. Dies geschieht durch

```
! INPUT variable
```

Durch diese Anweisung wird die Abarbeitung eines Programms angehalten und der Rechner wartet auf die Eingabe einer Zahl von der Tastatur, indem er ein Fragezeichen ausgibt. Die Eingabe wird durch ENTER abgeschlossen. Die eingegebene und angezeigte Zahl wird der Variablen hinter INPUT zugewiesen.

Beispiel:

INPUT X Tastatureingabe 67.Ø3

Im Programm kann nun X = 67.Ø3 wiederholt verwendet werden. Durch spätere Wiederholung von INPUT X kann der Variablen X eine andere Zahl zugewiesen werden.

5. Zeichenketten und -variable

Unser Computer kann auch Zeichenketten verarbeiten. Eine Zeichenkette besteht aus beliebigen Textstücken (Großbuchstaben, Sonderzeichen und Ziffern; Kleinbuchstaben nur beim Z 9001), und ist also eine Zeichenfolge. Die Zeichenfolge kann bis zu 255 Zeichen lang sein. Diese Zeichenketten, auch **Strings** genannt, dienen im Programm zu Erläuterungen und Hinweisen. Sie können auch wie arithmetische Ausdrücke verarbeitet werden und heißen dann Zeichenkettenausdrücke. Um sie zu speichern, muß man auch Variable – Zeichenketten oder Stringvariable – bereitstellen, die Strings diesen zuweisen und unter diesem Namen speichern. Wenn man dem wie üblich zu bildenden Variablennamen ein Sonderzeichen $ anhängt, erhält man eine Zeichenkettenvariable.

Beispiele: C$, C1$, CC$, AX$

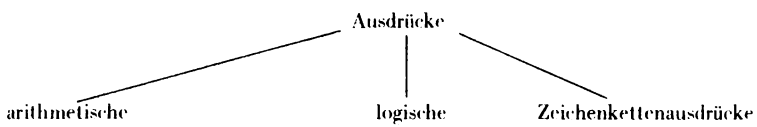
Die Zuordnung geschieht wieder mit der Ergibtanweisung

| stringvariable = "zeichenfolge"

Beispiele: C1$ = "DIES IST EINE ZEICHENKETTE"
A$ = "Begriff 1"
N5$ = "3. Wort"

Wie Sie aus den Beispielen erkennen, ist die **Zeichenkette immer in Anführungszeichen** zu setzen.
Die Belegung der Zeichenkettenvariablen kann wie bei numerischen Variablen abgerufen werden.
Durch beispielsweise ? A$ oder PRINT A$ wird die der Stringvariablen A$ zugewiesene Zeichenkette ausgegeben.
Der Arbeit mit Strings behalten wir einen gesonderten Abschnitt vor.
Damit haben wir 2 von den 3 Arten der Ausdrücke kennengelernt. Es fehlen noch die logischen Ausdrücke, die Sie später kennenlernen.

Demnach gibt es:



6. Programmerarbeitung

6.1. Programmaufbau

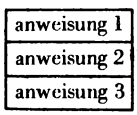
Ein Programm baut sich in BASIC aus aufeinanderfolgenden Anweisungen auf. Für übersichtliche Programme schreibt man auf jeder Zeile nur eine Anweisung. Die Zeilen werden in ihrer Reihenfolge numeriert. Üblicherweise geschieht dies in Zehnerschritten, um eventuell später weitere Anweisungen einfügen zu können.

Beispiel:

10 anweisung 1
20 anweisung 2
30 anweisung 3

Ein Ende braucht nicht angegeben zu werden.

Struktogramme verdeutlichen den Ablauf der Programme. Da die Anweisungen in ihrer Reihenfolge abgearbeitet werden, muß dies auch im Struktogramm zum Ausdruck kommen. Die Anweisungen werden von oben nach unten abgearbeitet, folgen also aufeinander.



Struktogramm einer Aufeinanderfolge (Sequenz) von Anweisungen

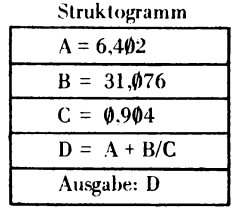
Die Bearbeitung der Anweisungen erfolgt immer in der Reihenfolge der Anweisungsnummern, auch wenn die Programmzeilen nicht geordnet sind.

6.2. Erste Programme

Nun können Sie schon beginnen, erste Programme zu entwickeln. Es soll ein arithmetischer Ausdruck berechnet und ausgegeben werden. Dazu werden 3 Variablen durch Ergibtanweisungen Zahlen zugewiesen. Danach wird der Ausdruck berechnet, das Ergebnis D zugewiesen und auf dem Bildschirm angezeigt.

```
10  A = 6.402
20  B = 31.076
30  C = .904
40  D = A + B/C
50  PRINT D
```

Struktogramm



Geben Sie das Programm ein!
Ist eine Zeile fertiggeschrieben, wird sie durch die ENTER-Taste abgeschlossen. Dadurch wird die Zeile in den Speicher übernommen. Wurde dies vergessen, kommt der folgende Text mit auf die nicht abgeschlossene Zeile.

> Werden Fehler vor dem Betätigen der ENTER-Taste bemerkt, kann man den Cursor auf die zu korrigierende Stelle setzen und diese überschreiben oder löschen.
> Werden Fehler nach dem Betätigen der ENTER-Taste bemerkt, muß die Zeilennummer und der richtige Text neu eingegeben werden. Dadurch wird die alte Zeile ungültig.

Vor Abarbeitung der Zeile erfolgt eine Kontrolle auf richtige Syntax (Syntaxkontrolle). Fehler werden durch ERROR und einen Fehlerhinweis gemeldet. Die Tafel der Fehlerhinweise finden Sie im Anhang. Die Fehlerhinweise helfen Ihnen, die Art des Fehlers zu erkennen.

Übung:

Ü 6. Starten Sie das Programm durch RUN und ENTER.

Wird ein Fehler angezeigt, so können Sie aus der Fehlermeldung

| Fehlerhinweis ERROR IN zeilennummer

die Zeile entnehmen, in der er sich befindet.
Durch LIST und ENTER oder LIST zeilennummer ENTER können Sie sich das Programm anzeigen lassen und den Fehler suchen.
Die Fehlerhinweise aus 2 Buchstaben weisen Sie daraufhin, welche Fehler gemacht wurden.

Soll die Berechnung des arithmetischen Ausdrucks in unserem ersten Programm auch noch für andere Zahlen erfolgen, muß das Programm geändert werden. Einfacher ist es, die Zahlen mit Hilfe einer INPUT-Anweisung einzugeben. Das Vorgehen bei Ein- und Ausgaben behandelt der nächste Abschnitt.

6.3. Ein- und Ausgaben

Eingaben werden durch INPUT-Anweisungen angefordert:

| INPUT eingabeliste

Ausgaben werden durch PRINT-Anweisungen realisiert:

| PRINT ausgabeliste

Die Ausgabeliste enthält die Ausdrücke, deren Werte ausgegeben werden, und die auszugebenden Strings. Die Trennung der Ausdrücke erfolgt durch Semikolon oder Komma.
Hinter PRINT dürfen Zahlen, Variable, Strings und aus diesen gebildete Ausdrücke stehen.

Beispiele zu Eingaben:

INPUT M	Eingabe einer Zahl. Sie wird durch Fragezeichen angefordert.
INPUT M, N	Eingabe von 2 Zahlen. Die 2. Zahl wird durch 2 Fragezeichen auf einer neuen Zeile angefordert, wenn zum Abschluß der 1. Zahl ENTER benutzt wurde.
INPUT "M="; M	Es erscheint der Text M= auf dem Bildschirm und danach ist die Zahl für M einzugeben.
INPUT "K:L:M:N:";K,L,M,N	Es wird der Text K:L:M:N: ausgegeben, der die Reihenfolge der Eingaben kennzeichnet.

Eine Eingabe wird durch ein ? angefordert. Sind mehrere Zahlen einzugeben, so kann man jede Zahl durch ENTER abschließen oder die Zahlen durch Komma trennen und nach der letzten mit ENTER beenden.
Werden die Zahlen durch Komma getrennt und zu wenige eingegeben, erfolgt eine weitere Anforderung durch ? ? .
Werden zu viele eingegeben erfolgt die Anzeige: EXTRA IGNORED!
Stimmen Datentyp und Variable nicht überein, meldet sich der Computer mit ? REDO FROM START. Die Eingabe ist zu wiederholen.

Beispiele zu Ausgaben:

PRINT C	Ausgabe des Wertes von C
PRINT C; X	Ausgabe der Werte von C und X auf eine Zeile hintereinander
PRINT C, X	Ausgabe der Werte von C und X auf eine Zeile in 2 Spalten (Tabellenform)
PRINT C PRINT X	Ausgabe der Werte von C und X auf 2 Zeilen.
PRINT	Ausgabe einer Leerzeile
PRINT "X="; X	Ausgabe des Textes X= und anschließend des Wertes von X.
PRINT A + B/C	Das jeweilige Ergebnis des Ausdruckes und vorhandene Zeichenketten werden ausgegeben.

```
PRINT SIN (ALFA)
PRINT SQR (R + S * S)
PRINT "RADIKAND=";X,"(WURZEL X)=";SQR(X)
```

Beispiel:

```
10 INPUT "X EINGEBEN:"; X
20 INPUT "Y EINGEBEN:"; Y
30 Z1 = X ^ Y
40 Z2 = X * Y
50 Z3 = X + Y
60 PRINT "POTENZ", "PRODUKT", "SUMME"
70 PRINT Z1, Z2, Z3
```

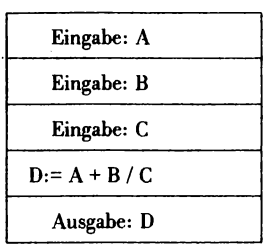
Geben Sie sich 2 Zahlen vor und überlegen, welche Ausgabe durch das Programm erfolgen wird.
Das Programm wird eingegeben und durch RUN gestartet. Hinter dem Text, der die Eingabe anfordert, schreiben Sie jeweils die angeforderte Zahl und beenden die Eingabe durch ENTER.

Übungen:

Ü 7. Ändern Sie das vorangegangene Programm, indem Sie in Zeile 60 und 70 statt Kommas Semikolons setzen.

Ü 8. Schreiben Sie ein zweizeiliges Programm. Durch die 1. Anweisung soll eine Zahl Z eingegeben, durch die 2. Anweisung soll dieselbe Zahl ausgegeben werden.
Kontrollieren Sie nach der Programmabarbeitung durch PRINT Z, daß die Zahl Z im Speicher erhalten blieb!

Ü 9. Arbeiten Sie das Programm von Abschnitt 6.2. so um, daß am Anfang 3 INPUT-Anweisungen stehen. Gestalten Sie die Eingaben so, daß es zu keinen Verwechslungen in der Reihenfolge der Variablen kommen kann.



Ü 10. Erarbeiten Sie ein Programm, das nur eine Eingabe- und eine Ausgabeanweisung enthält. Bei der Eingabe von A soll als erläuternder Text ausgedruckt werden: EINGABE VON A: Die Ausgabe soll folgendes Bild bringen:
WURZEL VON A IST GLEICH C
wobei für A und C konkrete Zahlen stehen sollen.

Zusammenfassung:

Eingabe von mehreren Variablen durch

```
INPUT X, Y, Z      Variable durch Komma getrennt.
```

Eine Zeichenkette (in Anführungszeichen) zur Kennzeichnung der Variablen auf dem Bildschirm wird ausgegeben.

```
INPUT "X=";X
```

Die allgemeinste Form der INPUT-Anweisung lautet:

```
INPUT „string";variable,variable ...
```

Ausgabe von mehreren Variablen durch

```
PRINT X, Y, Z    Ausgabe in 3 Spalten (Tabelle), wenn Variable durch Komma getrennt.
                 Ausgabe hintereinander, wenn durch Semikolon getrennt.
PRINT "AUSGABE: X=";X      Ausgabe von Text und Zahlenwert
```

6.4. Programmänderungen

Sie können Programme verändern, ergänzen und auch Streichungen vornehmen. Dazu müssen Sie

```
EDIT zeilennummer
```

eingeben. Dann können Sie in der aufgerufenen Zeile korrigieren. Die Tasten ← und → bewegen den Cursor in die entsprechende Richtung. Dort wo der Cursor steht, kann man das vorhandene Zeichen

- mit der Taste DEL löschen,
- mit der Taste INS Platz für weitere Zeichen schaffen.
- durch Eingabe eines neuen Zeichen, das alte überschreiben.

Durch ENTER wird die Korrektur übernommen und die nächste Zeile angezeigt.

Den EDIT-Modus beendet man mit der Taste BRK bzw. STOP (Z 9001). Nach der letzten Programmzeile wird er automatisch verlassen!

Übung:

Ü 11. Verändern Sie das vorangegangene Programm wie folgt:

```
10  INPUT "A=";A
20  INPUT "B=";B
30  INPUT "C=";C
40  D = A + B / C
41  M = (A + B + C) / 3
42  F= SQR (ABS (D))
50  PRINT "MITTEL="; M
51  PRINT "WURZEL D ABSOLUT ="; F
```

Die Zeile 5Ø kann im EDIT-Modus überschrieben oder gänzlich neu zusammen mit Zeilen 41, 42 und 51 eingegeben werden. Das Programm kann durch RENUMBER automatisch in Zehnerschritten neu numeriert und, nachdem sich der Computer mit OK gemeldet hat, wieder angezeigt werden. Geben Sie dazu RENUMBER und LIST ein.

Zusammenfassung:

- Ein Programm besteht aus Zeilen mit Zeilennummern.
- Eine Programmzeile wird durch Betätigen der ENTER-Taste gespeichert. Sie kann max. 72 Zeichen enthalten. 4Ø passen auf eine Bildschirmzeile.
- Die Programmzeilen werden in der Reihenfolge ihrer Zeilennummern aufgelistet und abgearbeitet.
- Programmzeilen kann man verändern oder neu eingeben.
- Programmzeilen kann man durch Eingabe der Zeilennummer und ENTER löschen.
- Für übersichtliche Programme schreibt man auf jede Zeile nur eine Anweisung.
- Ein Programm wird durch RUN gestartet.
- Der Programmtext kann durch LIST angezeigt werden.
- Zeichenketten, die in Anführungszeichen eingeschlossen sind, hinter der INPUT- oder der PRINT-Anweisung werden mit ausgegeben.

6.5. Zählschleife

Es soll ein Programm geschrieben werden, das eine Tabelle erarbeiten und anzeigen läßt. Von einer einzugebenden Zahl soll der Kehrwert, die Quadratwurzel und die Quadratzahl berechnet werden. Die Berechnung soll wiederholt vorgenommen werden.

```
1Ø INPUT X
2Ø E1 = 1/X
3Ø E2 = SQR(X)
4Ø E3 = X * X
5Ø PRINT E1, E2, E3
```

Starten Sie das Programm durch RUN, und geben Sie X = 1 ein. Wiederholen Sie den Start, und geben Sie X = 2 ein usf. Um z. B. eine Tabelle für X = 1 bis 1ØØ zu erzeugen, sind die ständigen Eingaben mühsam. Für solche wiederkehrenden Berechnungen und gleichzeitige Veränderung der Variablen gibt es die Schleifenanweisung.

```
FOR laufvariable = anfangswert TO endwert STEP schrittweite
  anweisung
  anweisung
  .
  .
  .
NEXT laufvariable
```

Durch die Zählschleife wird die Laufvariable, beginnend mit dem Anfangswert, in jedem Durchlauf um die Schrittweite erhöht, bis die Laufvariable den Endwert erreicht hat. Natürlich werden auch die Anweisungen zwischen FOR und NEXT bei jedem Durchlauf bearbeitet.

1. Durchlauf laufvariable = anfangswert
2. Durchlauf laufvariable = anfangswert + schrittweite
3. Durchlauf laufvariable = anfangswert + 2*schrittweite

Wenn bei $\begin{matrix}\text{positiver}\\ \text{negativer}\end{matrix}$ Schrittweite laufvariable $\gtrless$ endwert ist, endet die Schleife. Das Programm wird in der Zeile nach NEXT fortgesetzt.

Unser Programm kann durch die Zählschleife wie folgt geschrieben werden:

```
10 FOR X = 1 TO 10 STEP 1
20   PRINT 1/X, SQR(X), X*X
30 NEXT X
```

In der Zeile 10 wird die Laufvariable X von 1 beginnend bereitgestellt, in Zeile 20 benutzt und danach vor jedem neuen Durchlauf um $\Delta X = 1$ (Schrittweite 1) erhöht.
In Zeile 20 sind die Ergibtanweisungen entfallen. Die arithmetischen Ausdrücke wurden hinter die PRINT-Anweisung geschrieben. Die Ausdrücke werden dadurch berechnet und angezeigt aber nicht gespeichert. Den Abschluß muß immer die Anweisung NEXT X bilden. Dadurch wird das Ende eines Durchlaufes angegeben und die Wiederholung der Programmzeilen angeordnet. Auf diese Weise kommt ein Zyklus oder eine Schleife zustande. Starten Sie das Programm!

Durch Einfügen der Zeilen
15 PRINT "X=";X und
25 PRINT
kann das Druckbild verbessert werden.

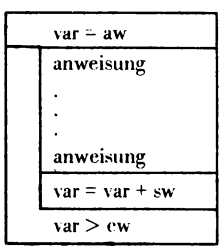
Nachfolgend einige Beispiele für die Werte, die die Laufvariable in den folgenden Zählschleifen annimmt.

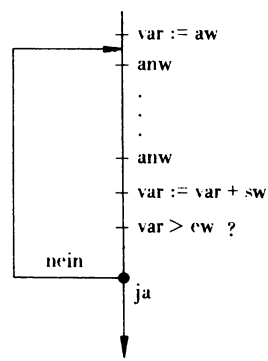
FOR A=5 TO 10 STEP 2	A = 5, 7, 9
FOR B=0 TO 12 STEP 3	B = 0, 3, 6, 9, 12
FOR C=−3.5 TO 0 STEP .5	C = −3.5, −3.0, −2.5, −2.0, −1.5, −1.0, −0.5, 0
FOR D = 12 TO −11 STEP −6	D = 12, 6, 0, −6

Übung:

Ü 12. Welche Werte durchläuft die Variable X in den folgenden Anweisungen?
FOR X = 0 TO PI STEP 0.5
FOR X = 5 TO −10 STEP −3

Zur Veranschaulichung der Schleifenentstehung wird die Darstellung des Ablaufs im Struktogramm und durch die Programmlinie für positive Schrittwerte angegeben.





aw	Anfangswert
ew	Endwert
sw	Schrittweite
var	Laufvariable

Mit dem Anfangswert für die Laufvariable beginnend wird in die Schleife gegangen. Nach Abarbeitung der Schleifenanweisungen wird die Laufvariable um die Schrittweite erhöht und danach geprüft, ob die Bedingung – Variable größer Endwert – erfüllt ist. Bei Erfüllung der Bedingung wird das Programm mit der nächsten Zeile nach NEXT fortgesetzt. Bei Nichterfüllung werden die Schleifenanweisungen, jetzt aber mit der um die Schrittweite erhöhten Variable, erneut durchlaufen.

Beispiele: (Programmausschnitte)

```
    5Ø  S = Ø
   ⎡6Ø    FOR I=1  TO 5 STEP 2
Schleife 7Ø    S = S + I
   ⎣8Ø    NEXT I
    9Ø  PRINT S
```

	I	S
	beliebig	Ø
1. Durchlauf	1	1
2. Durchlauf	3	4
3. Durchlauf	5	9

Nach dreimaligem Durchlaufen der Schleife wird diese verlassen und mit Zeile 9Ø das Programm fortgesetzt, also der letzte Wert von S ausgegeben. Er beträgt S = 9.

Beträgt die Schrittweite +1, so kann STEP 1 entfallen. Dies wird in den folgenden Beispielen genutzt.

```
1Ø  L = Ø
2Ø    FOR X = 5 TO 1
3Ø      L = L + X
4Ø    NEXT X
5Ø  PRINT L
```

Die Anweisung 3Ø wird trotz Anfangswert > Endwert einmal ausgeführt. Das Resultat ist L = 5

Übungen:

Berechnen Sie X, bevor Sie Ihre Ergebnisse mit dem Computer überprüfen!

Ü 13.
```
1ØØ  X = 1
11Ø    FOR N = 1 TO 5
12Ø      X = X * 2
13Ø    NEXT N
14Ø  PRINT X
```

Ü 14.
```
1ØØ  X1 = 5
11Ø  X2 = Ø
12Ø    FOR N = 1 TO 4
13Ø      X1 = X1 + N
14Ø      X2 = X2 + N ∧ 2
15Ø    NEXT N
16Ø  PRINT "X1=";X1, "X2=";X2
```

Schleifen dürfen ineinander verschachtelt werden. Dabei muß die innere Schleife vollständig von der äußeren umfaßt werden.

```
2ØØ  FOR K = 1 TO 1Ø STEP 3
21Ø    FOR P = 2 TO 8
22Ø      A = P * K
23Ø      PRINT A
24Ø    NEXT P
25Ø  NEXT K
```

innere Schleife

äußere Schleife

Die Zeilen 24Ø und 25Ø können zu NEXT P, K zusammengefaßt werden.

Übungen:

Ü 15. Lassen Sie durch eine Zählschleife dreimal hintereinander das Wort BASIC ausdrucken.

Ü 16. Überlegen Sie, welche Werte das Produkt P = K · I bei der Abarbeitung der folgenden verschachtelten Zählschleifen annimmt.

```
10   FOR I = 1 TO 3 STEP 1
20     FOR K = 2 TO 10 STEP 2
30       PRINT K * I;
40     NEXT K
50   NEXT I
```

Prüfen Sie Ihr Ergebnis am Rechner.

Zusammenfassung:

Die Anweisungen, die zusammen die Zählschleife realisieren, haben folgende Gestalt:

```
FOR variable = anfangswert TO endwert STEP schrittweite
  anweisungen
NEXT variable
```

Für Anfangs- und Endwert sowie Schrittweite können beliebige arithmetische Ausdrücke stehen.

```
FOR   N = Z + 2 TO P * R/3 STEP Z/2
  .
  .
  .
NEXT N
```

Z, P und R müssen vor ihrer Benutzung in der Schleife eine Wertzuweisung erfahren haben.
Wenn die Schrittweite +1 beträgt, kann die Angabe STEP 1 entfallen.

Die Variable ist nach FOR und NEXT die gleiche und heißt Laufvariable.
Die in den Anweisungen zwischen FOR und NEXT vorhandenen anderen Variablen werden in die Veränderung der Laufvariablen nicht einbezogen, z. B.

```
100  A = 10
110    FOR N = .1 TO 2 STEP .1
120      E = \ * EXP (N/2)
130      PRINT E
140    NEXT N
```

Achtung:

Die Anweisungen zwischen FOR und NEXT werden in jedem Fall einmal abgearbeitet! Außerdem können Schrittweite und Endwert im Verlaufe der Bearbeitung der Schleife nicht mehr verändert werden.

7. Einige Kommandos

Die nachfolgend zu besprechenden Kommandos werden im Direktmodus nach Betätigen der ENTER-Taste sofort ausgeführt. Sie dienen überwiegend dazu, Programme zu verändern.

LIST

Wie schon erwähnt, kann man durch LIST das abgespeicherte Programm auf dem Bildschirm anzeigen. Dabei werden 1Ø Zeilen des Programms ausgegeben. Nach ENTER werden weitere 1Ø Zeilen angezeigt. Folgt auf LIST eine Zeilennummer, so wird das Ausgeben bei dieser Zeilennummer begonnen.
Der LIST-Modus wird durch die Taste BRK (KC 85/2) bzw. STOP (Z 9001) oder automatisch am Programmende verlassen.

EDIT zeilennummer

Durch obiges Kommando kann man in der aufgerufenen und automatisch angezeigten Zeile Korrekturen vornehmen. Durch ENTER werden diese übernommen und die nächste Zeile bereitgestellt. Sind keine Änderungen weiter notwendig, beendet man den EDIT-Modus durch die Taste BRK (KC 85/2) bzw. STOP (Z 9001) oder automatisch am Programmende.

AUTO

Durch AUTO werden selbständige Zeilennummern von 1Ø beginnend in Zehnerschritten bereitgestellt. Dadurch können Sie sich die Numerierung ersparen, wenn Sie das Programm zeilenweise in richtiger Reihenfolge eingeben.
AUTO znr beginnt die Numerierung bei der angegebenen Zeilennummer. Das Ende der automatischen Zeilennummerierung wird durch die Taste BRK bzw. STOP herbeigeführt.

RENUMBER

Dieses Kommando stellt für alle Zeilen – beginnend bei der Zeilennummer der ersten Programmzeile – den Zehnerschrittabstand her. Es berücksichtigt auch alle Zeilennummern im Programm, die die Fortsetzung des Programms bei der angegebenen Zeile anzeigen.

Mit RENUMBER alte anfangszeilennummer,
alte endzeilennummer,
neue anfangszeilennummer,
Schrittweite
kann man Zeilennummern von Programmteilen verändern.

DELETE znr 1, znr 2

Damit kann man die Zeilen zwischen den angegebenen Zeilennummern einschließlich dieser löschen.

RUN

ist Ihnen schon bekannt und startet die Programme. **Gleichzeitig werden alle Variablen gelöscht** (Null gesetzt)!

BRK (KC 85/2) bzw. STOP (Z 9001)
hält ein laufendes Programm an. Mit

BREAK IN znr

wird die Unterbrechungsstelle angegeben. Die Abarbeitung des Programms kann durch

CONT

fortgesetzt werden.
Während der Programmunterbrechung können Sie andere Berechnungen durchführen!

Abschließend Kommandos, die Löschungen veranlassen.
Bei ihrer Anwendung ist Vorsicht geboten!

CLEAR löscht alle Variablen.
CLS löscht den Bildschirm.
NEW löscht alle im Speicher (RAM) befindlichen Programme und Variablen.

8. Einfache Programme

Beispiel:

Berechnen Sie die Dreiecksfläche $A = \frac{g \cdot h}{2}$ mit g = 2,7 cm und h = 3,2 cm. Gestalten Sie das Programm besonders bei den Ein- und Ausgaben übersichtlich.

```
10 REM BERECHNUNG DREIECKSFLAECHE
20 CLS
30 INPUT "G = "; G
40 INPUT "H = "; H
50 A = G * H/2
60 PRINT "A = "; A; "QUADRATZENTIMETER"
```

In Zeile 10 ist REM die Abkürzung von Bemerkung.
Eine so gekennzeichnete Zeile wird bei der Abarbeitung übergangen und nur nach LIST ausgedruckt. Die Angaben nach REM sind Erläuterungen für den Nutzer.

Beispiel:

Berechnen Sie mit Hilfe der FOR . . . NEXT-Anweisung die Summe der Zahlen von 1 bis 100.

```
10 REM SUMME DER NATUERLICHEN ZAHLEN BIS 100
20 S = 0
30   FOR N = 1 TO 100
40     S = S + N
50   NEXT N
60 PRINT "Summe ="; S
```

Übung:

Ü 17. Berechnen Sie die Summe der geraden/ungeraden Zahlen zwischen 0 und 100.

Beispiel:

Erarbeiten Sie ein Programm, welches eine Tabelle für den Winkel zwischen 0 und 90° in 10°-Schritten, in Grad- und Bogenmaß und den Wert der Sinusfunktion ausgibt.

```
10 PRINT "GRAD", "BOGEN", "SIN(X)"
20 PRINT
30   FOR I = 0 TO 90 STEP 10
40     X = I/180 * PI
50     PRINT I, X, SIN(X)
60   NEXT I
```

Übungen:

Ü 18. Erarbeiten Sie ein Programm, das eine Tabelle der Sinus- und Cosinusfunktion druckt. Das Argument X soll in Schritten von 0,1 bis π erhöht werden.

Ü 19. Eine Kugel rollt aus der Ruhelage eine schiefe Ebene herab. Die Geschwindigkeit am Ende der geneigten Ebene beträgt

$$v = \sqrt{\frac{10}{7} \cdot g \cdot s \cdot \sin\alpha}$$

Entwickeln Sie ein Programm zur Berechnung der Geschwindigkeit. Variieren Sie die Länge und den Winkel der Ebene durch Zählschleifen und lassen sich die Ergebnisse in Tabellenform ausdrucken.

ALFA/GRAD	5		10		25	
	S	V	S	V	S	V
	1		1			
	2		2			
	.		.			
	.		.			
	.		.			
	10		10			

g Erdbeschleunigung
s Länge der schiefen Ebene
α Neigungswinkel

9. Programmverzweigung

9.1. Vollständige Alternative

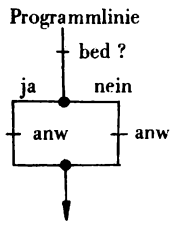
Mit der Alternative wird die Verzweigung eines Programms beschrieben.
Die Schreibweise für eine vollständige Alternative lautet:

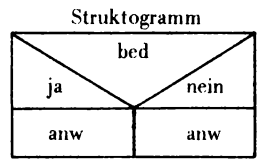
```
IF bedingung  THEN anw 11: anw 12: . . .
              :ELSE anw 21: anw 22: . . .
```

Sie wird gelesen:
Wenn Bedingung erfüllt, dann anw 11; anw 12: . . .
sonst anw 21: anw 22: . . .

Die Anweisungen 11, 12 usf. sind die des 1.. die Anweisungen 21, 22 usf. die des 2. Zweiges.
Nach Durchlaufen eines Zweiges wird das Programm mit der nächsten Zeile fortgesetzt. Welcher Zweig durchlaufen wird, hängt von der Erfüllung der Bedingung ab. Die gesamte Anweisung muß auf eine Zeile gehen; selbst bei Zuhilfenahme einer Hilfszeile können es maximal 72 Zeichen sein.

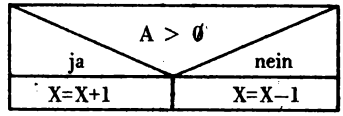
Ist die Bedingung erfüllt und damit wahr, so kann man mit ja antworten, und das Programm wird hinter THEN fortgesetzt. Ist die Bedingung nicht erfüllt, also falsch, so muß man mit nein antworten, und das Programm wird hinter ELSE fortgeführt. Demzufolge wird der Zweig hinter THEN auch als ja-Zweig und der hinter ELSE auch als nein-Zweig bezeichnet. Diese Bezeichnungen finden Sie im Struktogramm und bei der Programmliniendarstellung wieder.





Beispiel:

IF A > Ø THEN X = X + 1 : ELSE X = X – 1



Beispiel:

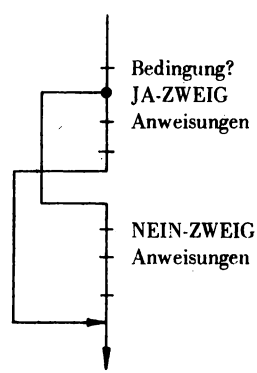
Das Programm soll die Quadratwurzel aus einer Zahl R berechnen. Wenn der Radikand R negativ ist, soll eine entsprechende Ausgabe erfolgen.

```
1Ø REM WURZEL AUS R
2Ø INPUT "R = "; R
3Ø IF R > = Ø THEN A = SQR(R) : PRINT "WURZEL R = "; A
        :ELSE PRINT "R NEGATIV"
4Ø PRINT "FORTSETZUNG DES PROGRAMMS"
```

In diesem Programm steht $A = \sqrt{R}$ nur dann ab Zeile 4Ø zur Verfügung, wenn R > = Ø ist. Ist R < Ø, wird der Hinweis ausgegeben, daß die Wurzel nicht ausgeführt werden kann. In Zeile 4Ø werden beide Zweige des Programms wieder vereint.

Ein Programm wird in der Reihenfolge der Zeilennummern abgearbeitet. Diese Reihenfolge kann nur durch die Anweisungen IF . . ., FOR . . . und GOTO durchbrochen werden.
Da oft die notwendigen Anweisungen nicht auf die Zeile passen, muß man die Zeige nacheinander schreiben.
Man erhält dann folgende Schreibeweise beim Programmaufbau.

```
1ØØ REM ALTERNATIVE VOLLSTAENDIG
11Ø IF bedingung THEN 12Ø : ELSE 16Ø
12Ø REM JA-ZWEIG
13Ø anweisungen
 .
 .
 .
15Ø GOTO 2ØØ
16Ø REM NEIN-ZWEIG
17Ø anweisungen
 .
 .
 .
2ØØ REM FORTSETZUNG DES PROGRAMMS
```



Die Zeilennummern sind willkürlich. Hinter THEN und ELSE stehen Zeilennummern, die den Beginn der Zweige angeben. Nach der Bedingung folgt immer der JA-Zweig!

Die Zeile 15Ø enthält die Anweisung GOTO 2ØØ. Durch sie wird das Programm bei Zeile 2ØØ fortgesetzt. Sie dient also zum Übergeben des NEIN-Zweiges und ist nur im Zusammenhang mit den angegebenen Strukturen zu benutzen!

Beispiel zu dieser Schreibweise:
Wir wählen die Lösung $x_{1/2} = -p/2 \pm \sqrt{\frac{p^2}{4} - q}$ der quadratischen Gleichung $x^2 + px + q = Ø$. Es müssen 2 Fälle unterschieden werden.

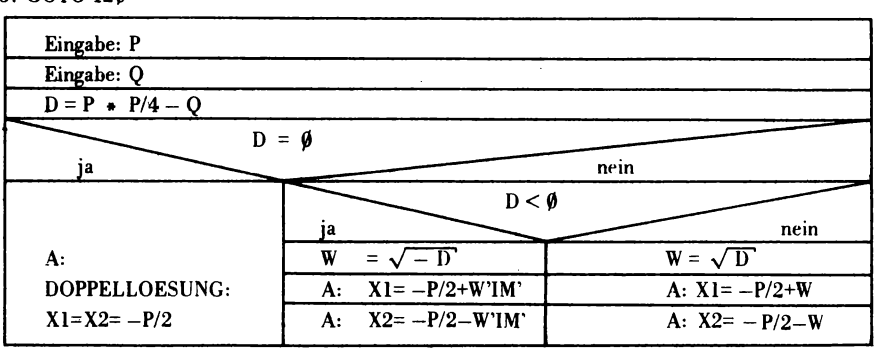
Die Diskriminante $D = p^2/4 - q$ ist kleiner oder größer gleich Null. Je nachdem, wie die Entscheidung ausfällt, sind unterschiedliche Anweisungen im Programm aufzuschreiben.

```
          10  INPUT "P = "; P
          20  INPUT "Q = "; Q
          30  D = P * P/4 - Q
          40  IF D < 0 THEN 50 : ELSE 90
JA-       50  W = SQR(-D)
ZWEIG     60  PRINT "X1="; -P/2; "+"; W; "IM"
          70  PRINT "X2="; -P/2; "-"; W; "IM"
          80  GOTO 120
NEIN-     90  W = SQR(D)
ZWEIG    100  PRINT "X1 = "; -P/2 + W
         110  PRINT "X2 = "; -P/2 - W
         120  REM "FORTSETZUNG DES PROGRAMMS"
```

Die in Zeile 80 formulierte Anweisung GOTO 120 dient wie gesagt dazu, den NEIN-Zweig zu übergeben und an die Stelle zu kommen, wo das Programm fortgesetzt wird, ganz gleich, welcher Zweig durchlaufen wurde.

Will man bei der Lösung der quadratischen Gleichung den Fall, daß die Dikriminante Null ist, gesondert ausweisen, so muß man eine weitere Alternative benutzen, da jetzt 3 Fälle zu unterscheiden sind. Wir ändern unser Programm, indem wir die Zeilen 35 bis 37 einfügen.

```
35 IF D = 0 THEN 36 : ELSE 40
36 PRINT "DOPPELLOESUNG: X1 = X2 ="; -P/2
37 GOTO 120
```



Übung:

Ü 20. Geben Sie das Programm ein und testen Sie es mit Beispielen aus Ihrem Mathematikbuch.

9.2. Mögliche Operatoren in den Bedingungen

9.2.1. Vergleichsoperatoren

In einem Vergleich kann man prüfen, ob eine Variable größer oder gleich einer vorgegebenen Zahl oder dem Wert eines arithmetischen Ausdruckes ist.
Wie schreibt man eine Bedingung, die dies berücksichtigt?

Beispiele:

2ø IF R > = ø THEN . . . 1øø IF A = M/3 + 1 THEN . . .

Folgende Vergleichsoperatoren stehen für die Formulierung von Bedingungen zur Verfügung.

Vergleichsoperatoren

= gleich
> größer
< kleiner
> = größer oder gleich
< = kleiner oder gleich
< > ungleich

9.2.2. Logische Operatoren

Einfache Bedingungen mit Vergleichsoperatoren können darüber hinaus durch die logischen Operatoren AND(UND), OR(ODER) und NOT(NICHT) verknüpft werden.

Beispiel:

Hat man in einem Koordinatensystem 2 variable Punkte und will prüfen, ob diese zusammenfallen, so müssen die x- und y-Koordinaten untereinander gleich sein ($x_1 = x_2$, $y_1 = y_2$).

Eine solche Bedingung formuliert man durch:

X1 = X2 AND Y1 = Y2

Die Bedingung ist nur dann erfüllt, wenn gleichzeitig die Vergleiche links und rechts von AND erfüllt sind.

Man kann also formulieren:

| Ausdrücke, die die Operatoren AND, OR und NOT enthalten, sind logische Ausdrücke.

Die Aussage des logischen Ausdrucks

bedingung 1 AND bedingung 2

ist nur wahr, wenn beide Bedingungen wahr sind.
Die Aussage des logischen Ausdrucks

bedingung 1 OR bedingung 2

ist wahr, wenn eine oder beide Bedingungen wahr sind. Sie ist falsch, wenn beide Bedingungen nicht wahr sind.
Durch NOT wird die Verneinung der nachfolgenden Bedingung ausgedrückt.

Beispiele:

- Will man eine Zahl X eingrenzen, so kann diese zwischen unterer und oberer Grenze liegen. Man schreibt für eine solche Bedingung:

  X > UG AND X < = OG UG untere, OG obere Grenze

- Soll eine Zahl außerhalb eines Intervalls liegen, nutzt man OR.
  Ein Programm soll verzweigt werden in Abhängigkeit davon, ob A außerhalb des Intervalls ($1 \leqq X \leqq 3$) liegt.

  Programmausschnitt:

```
11Ø INPUT A
12Ø IF A < 1 OR A > 3 THEN PRINT "A LIEGT AUSSERHALB"
                      : ELSE PRINT "A IM INTERVALL"
```

  Testen Sie den Programmausschnitt mit Zahlen zwischen Ø und 4!

- NOT kann man in einem Programm bei der Prüfung einer Variablen auf Null nutzen, weil dann mit der Variablen nicht dividiert werden darf.

  Programmzeile:

  22Ø IF NOT A=Ø THEN Q=S/A : ELSE PRINT "DIVISION MIT Ø VERBOTEN"
  NOT A=Ø kann auch als A <> Ø geschrieben werden.

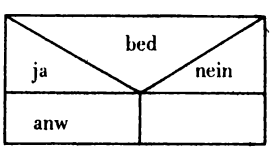
9.3. Unvollständige Alternative

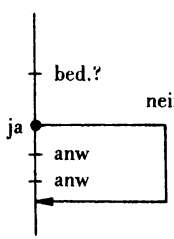
Die unvollständige Alternative besteht, wie die vollständige Alternative, aus 2 Zweigen. Nur enthält hier der NEIN-Zweig keine Anweisung und dient zum Überspringen des JA-Zweiges.

```
IF bed THEN anw 11:anw 12 . . .   (:ELSE 11Ø)
```

ELSE zeilennummer kann entfallen, weil bei nichterfüllter Bedingung das Programm automatisch mit der nächsten Zeile fortgesetzt wird.
Gehen die Anweisungen nicht auf eine Zeile, muß folgendermaßen geschrieben werden:

```
3ØØ IF bed THEN 31Ø : ELSE 35Ø
31Ø anw
  .  anw
  .
  .
35Ø
```

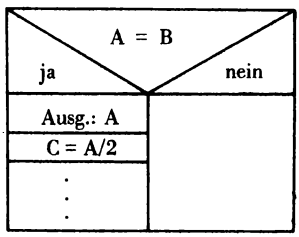


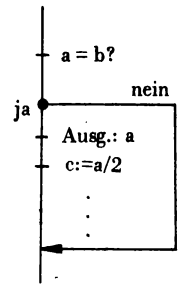


Ist die Bedingung nicht erfüllt, werden die Anweisungen übergangen. Im Struktogramm bzw. bei der Programmlinie äußert sich das durch einen leeren Zweig.

Beispiel:

```
100 IF A = B THEN 110 : ELSE 180
110 PRINT A
120 C = A/2
130 ...
 .
 .
 .
180 ...
```





Übung:

Ü 21. Eine Zahl L soll eingegeben und danach geprüft werden, ob sie negativ ist.
Bei JA soll L IST NEGATIV ausgegeben,
bei NEIN soll diese Anweisung übergangen werden.
Schreiben Sie für diesen Programmausschnitt die notwendigen Anweisungen auf, und zeichnen Sie das Struktogramm!

9.4. Fallauswahl

Ein Programm kann auch mehrfach verzweigt werden. Das tritt besonders im Zusammenhang mit Menüs auf.
Ein solches Menü könnte lauten:

Die Berechnung soll für ein Hohlprofil durchgeführt werden, dessen Querschnitt
rechteckig /1
quadratisch /2
kreisrund /3 ist.
Die Berechnung soll nicht durchgeführt werden. /4

Entsprechend Ihrer Entscheidung müssen Sie die Zahl 1 . . . 4 eingeben. Danach wird der gewünschte Programmzweig aktiviert. Die dazugehörige Anweisung lautet:

```
ON N GOTO znr 1, znr 2, znr 3, znr 4
```

Die Anweisung wird als Verteileranweisung bezeichnet.
Ist N = 1, wird zur 1. Zeilennummer (znr 1) hinter GOTO gegangen, ist N = 2, wird bei der 2. Zeilennummer fortgesetzt usw.

Das Struktogramm sieht folgendermaßen aus:

N = 1	N = 2	N = 3	N = 4
(1. Zweig) anweisung	(2. Zweig) anweisung	(3. Zweig) anweisung	(4. Zweig) anweisung

N muß natürlich vor der Abfrage eingegeben (INPUT N) oder auch berechnet werden.
Ist N < 1 oder N größer als die Anzahl der Zeilennummern, wird die Verteileranweisung übergangen.
Eine ähnliche Fallauswahl (Verteiler) kann auch mit Unterprogrammen (siehe 16.) aufgebaut werden. Dabei ist GOTO durch GOSUB zu ersetzen.

Übung:

Ü 21. Entwickeln Sie ein Programm mit 3 Zweigen zur Berechnung der Masse von Stäben (Vollmaterial) aus Stahl unter Benutzung der Fallauswahl. Zuerst soll Text (Menü) wie im oben dargelegten Beispiel ausgegeben werden. Danach erfolgt in den 3 Zweigen die Berechnung von $m = \rho \cdot A \cdot l$ für die verschiedenen Querschnitte. Dazu sind die Maße für die Querschnittsberechnung und die Länge l einzugeben.

10. Schleifen

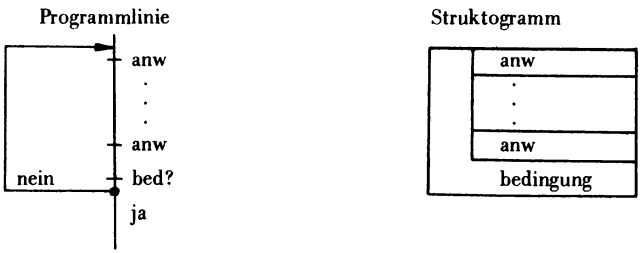
10.1. Schleifenaufbau

Um Schleifen aufzubauen, wird die IF . . . THEN . . . ELSE- oder die schon bekannte FOR . . . NEXT-Anweisung benötigt. Diese Anweisung und die GOTO-Anweisung sind elementare Mittel, um Schleifen zu formulieren. Mit ihnen können alle Schleifen beschrieben werden.
Es besteht aber die Gefahr, unübersichtliche Konstruktionen aufzubauen. Deshalb wird die Forderung erhoben, nur mit den im folgenden beschriebenen 2 Schleifen zu arbeiten und die Anweisung GOTO nur in diesem Zusammenhang zu benutzen.

10.2. Schleifenarten

10.2.1. Die nichtabweisende Schleife

Diese Schleife hat folgende Struktur:



Die Abfrage erfolgt nach dem Abarbeiten der Anweisungen innerhalb der Schleife. Ist die Bedingung nicht erfüllt, geht es zurück an den Schleifenanfang.

Die erfüllte Bedingung führt zum Schleifenabbruch und Fortsetzung mit der nächsten Anweisung im Programm. Deshalb wird die Bedingung auch **Abbruchbedingung** genannt.

Die notwendige Anweisungsfolge hat die Gestalt:

```
znr 1 anweisung
.
.
.
znr   IF bedingung THEN znr 2 : ELSE znr 1
znr 2 anweisung
```

Beispiel:

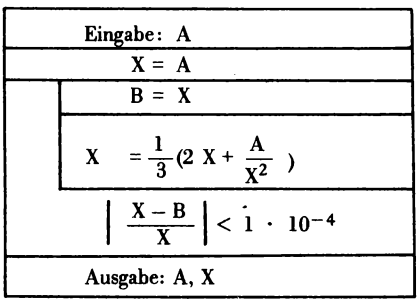
Als Beispiel für eine nichtabweisende Schleife soll die Berechnung einer 3. Wurzel mit der Rekursionsformel

$$x_{n+1} = \frac{1}{3}\left(2\,x_n + \frac{a}{x_n{}^2}\right)$$

vorgestellt werden.

```
 5  REM 3. WURZEL AUS A
10  INPUT "A = "; A                    Eingabe des Radikanden
20  X = A                              Berechnungsbeginn mit X = A
30    B = X                            Speichern des vorhandenen X-Wertes
40    X =(2*X+A/(X*X))/3               Berechnen des neuen X-Wertes
50    IF ABS ((X - B)/X) < 1 E - 4 THEN 60 : ELSE 30
                                       Berechnen der relativen Differenz zwischen neuem und
                                       altem Ergebnis und Abbruch, sofern
                                       Δx/x < 1 · 10^-4
60  PRINT "A = "; A, "(WURZEL A) = "; X    Ergebnisausgabe
```

Die Schleife erstreckt sich auf die Zeilen 30 bis 50.
Für A = 0 kann das Programm nicht verwendet werden.
Im nachfolgenden Struktogramm ist die Programmstruktur deutlich zu erkennen.



Übung:

Ü 22. Testen Sie das Programm und erweitern Sie es dahingehend, daß nach der Eingabe geprüft wird, ob A = 0 ist und diese Eingabe zurückgewiesen wird.

Als weiteres Beispiel wird die Berechnung der Punkte einer Wurfparabel vorgestellt (ohne Luftreibung), wenn die x-Koordinate wiederholend um Δx (im Programm DX) vergrößert wird. Die Gleichung lautet:

$$y = x \cdot \tan\alpha - \frac{x^2 g}{2 v^2 \cos^2\alpha}$$

α ist der Abwurfwinkel im Bogenmaß, v die Anfangsgeschwindigkeit in m/s und g die Erdbeschleunigung in m/s^2, x und y die Koordinaten in m.

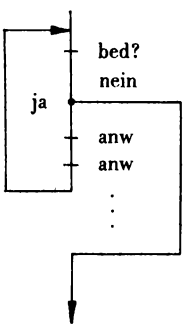
```
10  DX = 2.5
20  ALFA = 1
30  V = 30
40  G = 9.81
50  X = 0
60    X = X + DX
70    Y = X *TAN(ALFA) - X * X * G/(2 * (V* COS(ALFA)) ^ 2)
80    PRINT X, Y
90    IF Y < = 0 THEN 100 : ELSE 60
100 END
```

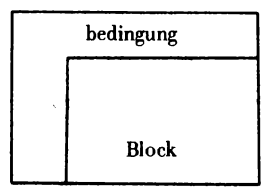
Die Anweisung END bezeichnet das Programmende, also diejenige Zeile, die erreicht wird, wenn die Bedingung erfüllt ist. Eine Notwendigkeit das Programm mit END abzuschließen besteht nicht, solange sich nur ein Programm im Arbeitsspeicher befindet.

Später werden wir die Y-Werte (Höhen) sogar auf dem Bildschirm zeichnen können.
Den Winkel ALFA und die Anfangsgeschwindigkeit V können Sie bei der nächsten Programmabarbeitung verändern. Abgebrochen wird der Schleifendurchlauf, wenn Y < = 0 wird, d. h. der Wurfkörper aufschlägt.

10.2.2. Die abweisende Schleife

Die abweisende Schleife hat folgenden Aufbau:





Ist die Bedingung nicht erfüllt, so wird die Schleife nicht durchlaufen und das Programm mit der nächsten Anweisung nach der Schleife fortgesetzt. Deshalb wird die Bedingung auch **Wiederholbedingung** genannt und ist auch so zu formulieren:

Wiederhole solange die Bedingung erfüllt ist!

Die Anweisungen haben folgende Anordnung:

```
znr 1 IF bedingung THEN znr 2 : ELSE znr 4
znr 2 anweisung
  .     .
  .     .
  .     .
znr 3 GOTO znr 1
znr 4 . . .
```

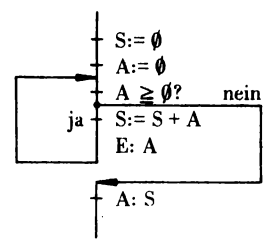
Beispiel:

Die Summierung beliebig vieler Zahlen > Ø ist durchzuführen und solange zu wiederholen, wie positive Zahlen eingegeben werden. Das Ende der Zahlenfolge kann demnach durch eine negative Zahl markiert werden.
Wir entwickeln das Programm:

```
1Ø  S = Ø                    Anfangswert der Summe = Ø
2Ø  A = Ø                    Anfangswert von A
3Ø  vorerst freilassen für die Abfrage
4Ø  S = S + A                Summe ergibt sich aus der schon vorhandenen
                             Summe + A
5Ø  INPUT A                  Eingabe von A
6Ø  GOTO 3Ø                  Rückkehr zum Anfang der Schleife, znr 3Ø
```

Wir haben eine Schleife gebildet, in die eingegeben und summiert wird. Dies wiederholt sich nun endlos, deshalb muß eine Abbruchbedingung eingefügt werden. Es ist die Anweisung IF bed THEN . . . : ELSE . . . , die in die freigelassene Zeile geschrieben wird.

```
1Ø  S = Ø
2Ø  A = Ø
3Ø    IF A > = Ø THEN 4Ø : ELSE 7Ø
4Ø    S = S + A
5Ø    INPUT A
6Ø    GOTO 3Ø
7Ø  PRINT "SUMME = "; S
```



Nach dem Start werden die Anweisungen in ihrer Reihenfolge bis Zeile 3Ø durchlaufen. Hier wird festgestellt, daß A = Ø ist und mit Zeile 4Ø fortgesetzt. S = Ø ist das Ergebnis von Zeile 4Ø. Danach wird die erste Eingabe vollzogen und die Schleife bei Zeile 3Ø neu begonnen.
Ist bei der Bedingungsprüfung A < Ø, wird der Schleifendurchlauf nicht ausgeführt und das Programm in Zeile 7Ø mit der Ausgabe der Summe beendet.

Die Anweisungen in der Schleife werden **wiederholt** bis zur Nicht-Erfüllung der Bedingung abgearbeitet. Deshalb wird dieser Vorgang Iteration (Wiederholung) und die Bedinung Wiederholbedingung genannt. Da die Bedingung am Anfang der Schleife steht, heißt die Schleife abweisend.

Übung:

Ü 23. Sie sollen das Programm erweitern, indem Sie die Anzahl der summierten Zahlen durch Zählen der Schleifendurchläufe mit der Anweisung I = I + 1 bestimmen und abschließend den Mittelwert der summierten Zahlen S/(I − 1) berechnen.

Beim Entwurf von Schleifen ist sorgfältig vorzugehen. Alle Schleifen sollten vor ihrem Einbau in ein Programm getestet werden. Dazu bringt man in die Schleife die Ergibtanweisung N=N + 1 ein, die die Durchläufe zählt und mit PRINT ausdruckt. Außerdem kann in diese PRINT-Anweisung auch die

Variable einbezogen werden, die in der Schleife berechnet wird. Dadurch kann deren Entwicklung verfolgt werden.

In den Programmausschnitt

```
240 A = 5
250 X = 0
260 Y = A
270  IF Y > .01 THEN 280 : ELSE PRINT X, Y : END
280  Y = A * EXP (-X)
290  X = X + .1
300  GOTO 270
```

der ab Zeile 270 wiederholt wird, solange Y > .01 ist, kann man

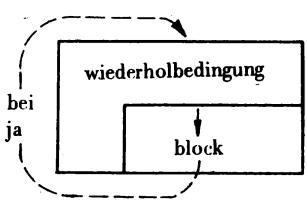
```
285 N = N + 1
286 PRINT N,X, Y
```

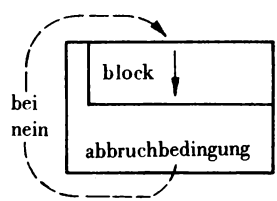
einfügen, um die Schleifenentwicklung zu verfolgen.
Außerdem sollte 265 N=0 vorangestellt werden.

Zusammenfassung:

Es gibt 2 Arten von Schleifen, die abweisende Schleife mit der Bedingung am Anfang und die nichtabweisende Schleife mit der Bedingung am Ende.
Die nichtabweisende Schleife wird mindestens einmal durchlaufen. Alle anderen konstruierbaren Schleifen sind auf diese beiden zurückführbar. Deshalb ist es sinnvoll, von Anfang an nur diese beiden Arten zu benutzen.

Zur Veranschaulichung dient die Darstellung als Struktogramm





Die Ihnen schon bekannte FOR . . . NEXT-Zählschleife ist im Kleincomputer-BASIC eine nichtabweisende Schleife, bei der die Anzahl der Durchläufe von Anfang an festliegt. Sie wird auch dann einmal durchlaufen, wenn der Anfangswert größer als der Endwert bei positiver Zählrichtung ist (Bedingung nicht erfüllt).
Vergleiche dazu Abschnitt 6.5.

Übung:

Ü 24. Berechnen Sie durch eine abweisende Schleife

$$y = a\ (1 - e^{\frac{-x}{k}})$$

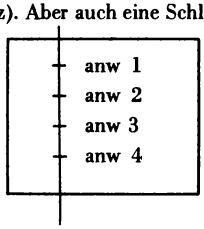
in Abhängigkeit von x für a = 5 und k = 0,1. Das Endergebnis (y, x, n) soll ausgegeben werden.

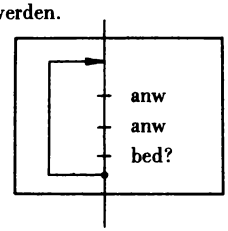
Die Berechnung soll solange wiederholt werden, wie $e^{\frac{-x}{k}} > 10^{-3}$ ist (Wiederholbedingung).
Die Anzahl der Wiederholungen soll gezählt und mit ausgegeben (n) werden.

11. Zum Strukturkonzept

Das Strukturkonzept beinhaltet Aussagen zu einem ingenieurgemäßen Vorgehen bei der Programmierung. Dies ist um so notwendiger, da bei der heutigen Kompliziertheit und dem Umfang der Software nur die Verwendung von einheitlichen Strukturkomponenten möglich ist, die das Zusammenspiel von Softwareprodukten erlauben. Wenn Sie auch zumeist nur einfache Programme entwickeln, ist die Kenntnis und Anwendung der „sinnvollen" Strukturen von Anfang an notwendig.

Hier kann verständlicherweise nur das Prinzip dieses Vorgehens bei der Programmentwicklung dargelegt werden und auch kein Anspruch auf Vollständigkeit erhoben werden.
Schon jedes einfache Programm sollte übersichtlich sein. Dies erreicht man durch eine klare Gliederung in bestimmte Abschnitte, die man Blöcke nennt. Ein Block ist ein Programmteil, der nur einen Eingang und nur einen Ausgang besitzt. Demnach ist der einfache Block eine Folge von Anweisungen (Sequenz). Aber auch eine Schleife kann als Block angesehen werden.





Nun zu den einzelnen Strukturelementen.
Ein Programm besitzt eine Programmhauptlinie. Soll diese verlassen werden, geschieht das durch eine Verzweigung. Hat sie 2 Zweige, so spricht man von Alternative, weil einer der Zweige durchlaufen wird. Die zugehörig Anweisung ist die

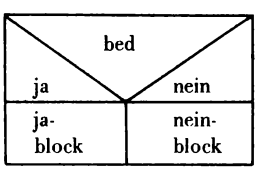
Alternative IF bed THEN JA-Zweig ELSE NEIN-Zweig

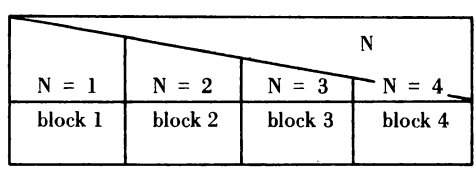
Hat die Verzweigung mehrere Zweige, so spricht man von

Fallauswahl ON N GOTO znr 1, znr 2, znr 3, . . .

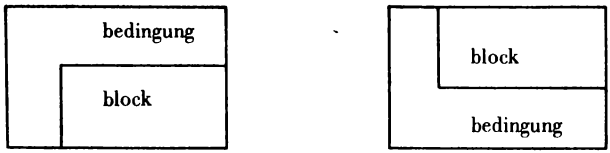
Je nachdem, ob N = 1, 2, 3, 4, . . . wird das Programm bei der Zeilennummer znr 1, znr 2, znr 3 usw. fortgesetzt.

N muß vorher eingegeben oder berechnet werden und eine natürliche Zahl sein.





Die möglichen Schleifen kennen Sie schon.
Es sind die abweisende und die nichtabweisende Schleife.



abweisende Schleife nichtabweisende

Der Eingang liegt im Struktogramm oben, der Ausgang unten.

Die Anweisung GOTO znr darf nur im Zusammenhang mit den dargelegten Strukturen – z. B. bei der abweisenden Schleife zum Rücksprung an den Schleifenanfang – verwendet werden.
Eine beliebige Verwendung von GOTO ist deshalb unerwünscht, weil dadurch unbeabsichtigte Schleifen entstehen können, die unbedingt zu vermeiden sind.

Dies sind die Elemente des Strukturkonzepts. Aus diesen Elementen sollen und können Sie Programme aufbauen. Die Blöcke werden dabei so verbunden, daß das Programm wieder einen Block bildet.
Mehrere Programme können nun nach bestimmten Ordnungsprinzipien (Hierarchie) zu einem Softwareprodukt zusammengefaßt werden. Wie dies geschieht, würde im Rahmen dieses Lehrbriefes zu weit führen. Sie müssen aber dafür sorgen, daß durch Anwendung der Strukturelemente der vorgezeichnete Weg zur Softwareprodukterarbeitung gegangen werden kann.
Dadurch wird es künftig möglich sein, den Computer umfassend zu nutzen und eine hohe Effektivität bei allen computernutzenden Prozessen zu erreichen.

Demnach verwenden wir:
- die Sequenz (Aufeinanderfolge)
- die Selektion
  - mit der Fallauswahl,
  - der vollständigen Alternative und
  - der unvollständigen Alternative
- die Iteration
  - mit der Zählschleife,
  - der nichtabweisenden Schleife und
  - der abweisenden Schleife.

12. Bereitstellung vieler Daten

Für das Bereitstellen größerer Mengen konstanter Daten in einem Programm stehen zwei besondere Anweisungen zur Verfügung. Sie lauten:

```
DATA datenliste
READ variablenliste
```

Daten können Zahlen oder Zeichenketten sein. Daten und Variable der Listen werden durch Komma getrennt.
Die Daten werden in das Programm übernommen und mit ihm ein- und ausgelesen. Dadurch spart man die wiederholte Eingabe.
Durch READ werden die hinter DATA stehenden Zahlen oder Zeichenketten den Variablen zugewiesen. Die Zuweisung geschieht in fortlaufender Reihenfolge. Man spricht auch vom Setzen eines Datenzeigers, durch den man die Zuordnung veranschaulicht.

Beispiel:

```
10 DATA   1,  2,  3,  4,  5,  6,  7,  8
          ....  ⇕  ....>
                                        Datenzeiger
20 READ   A,  B,  C,  D,  E,  F,  G,  H
30 PRINT  F, B, H
```

Als Ergebnis erhält z. B. die Variable B die Zahl 2 und F die Zahl 6 zugewiesen. Alle Variablen können Sie nun im Programm beliebig verwenden.
Ist die Zahl der einzulesenden Daten kleiner als durch die Anzahl der Variablen gefordert, so erfolgt eine Fehlermeldung.

Beispiel:

```
100 DATA 5, 10, 20
150 READ A, B, X, Y
 OD ERROR IN 150
```

Ist die Zahl der einzulesenden Daten größer als benötigt, so werden nur so viele gelesen wie angefordert.
Somit besteht auch die Möglichkeit, durch mehrere READ-Anweisungen etappenweise einzulesen.

Beispiel:

```
                1. Etappe    | 2. Etappe    | 3. Etappe
500 DATA   C1, C2, C3, E1, E2, E3, 666
510 READ   A$, B$, C$
.
.
.
630 READ   X1$, X2$, X3$
640 READ   A
```

Die Variablen hinter READ (normale Variable oder Zeichenkettenvariable) müssen mit der Art der Daten (Zahlen oder Zeichenketten) übereinstimmen.

Anführungszeichen bei der Eingabe von Zeichenketten können entfallen, wenn keine Kommas oder keine vorangestellte bzw. nachfolgende Leerzeichen vorhanden sind.

Gehen alle Daten nicht auf eine Zeile, können mehrere DATA-Anweisungen geschrieben werden, von denen die Daten in ihrer Reihenfolge gelesen werden. Die DATA-Anweisungen können an beliebiger Stelle im Programm stehen.

Beispiel:

```
260 DATA 3.5, 4.0, . . .
270 DATA 7.2, .1, 13.52, . . .
.
.
.
460 READ AB, AC, AD, . . .
```

Durch die Anweisungen

```
RESTORE zeilennummer
READ variablenliste
```

können die Daten nochmals an beliebiger Stelle im Programm auch anderen Variablen zugewiesen werden.

Es ist dadurch die Möglichkeit gegeben, bei einer beliebigen Zeilennummer der DATA-Anweisung zu beginnen. Dabei wird nur der Teil der Daten eingelesen, der ab dieser Zeile steht. Man nennt dies auch Rücksetzen des Datenzeigers.

Beispiel:

```
10 DATA 11
20 DATA A1, A2, A3
30 READ A
40 READ X1$, X2$, X3$
100 RESTORE 20
110 READ K$, L$, M$
```

Lassen Sie sich die Belegung der Variablen ausgeben!

Beispiel:

Auswahl einer Zahl aus einer Zahlenfolge (Tabelle).
Die gesuchte Zahl ist einzugeben. Die Tabellenwerte folgen nach DATA! Die gesuchte Zahl und ihre Stellung (N) in der Tabelle sind auszugeben.

```
10 INPUT "GESUCHTE ZAHL"; M
20 DATA 2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
30 FOR N=1 TO 15
40   READ X
50   IF X=M THEN PRINT N, X : I=I+1
60 NEXT N
70 IF I=0 THEN PRINT "WERT NICHT VORHANDEN"
```

Mit der Zählschleife werden die Zahlen der Reihe nach – im 1. Durchlauf die erste Zahl, im 2. Durchlauf die zweite Zahl usw. – der Variablen X zugewiesen, die in jedem Durchlauf neu belegt wird. Stimmen die Zahlen überein, so wird die Zahl und ihre Stellung ausgegeben. Gleichzeitig wird der Zähler I um 1 erhöht (Zeile 50). Dadurch kann in Zeile 70 geprüft werden, ob die Zahl nicht vorhanden ist. In Zeile 50 und 70 wurde der ELSE-Zweig weggelassen (unvollständige Alternative).

Übung:

Ü 25. Bestimmen Sie mit Hilfe des logischen Operators AND in der Zeile 50 die Zahlen, die in einem Intervall liegen. Die Intervallgrenzen sind vorher (Zeile 10) einzugeben.

13. FELDER

Wie in der Mathematik können Variable mit Indizes benutzt werden. Dabei interessiert hier die Möglichkeit, mehrere Indizes zu benutzen und diese selbst variabel zu gestalten. Die Schreibweise in BASIC unterscheidet sich von der mathematischen.

mathematisch	BASIC
x_1, x_2, a_3, c_0	X(1), X(2), A(3), C(0)
$x_{1,3}; z_{7,2}$	X(1,3), Z(7,2)
y_{ik}, a_{mn}	Y(I,K), A(M,N)

Eine Menge von Variablen, die sich durch ihre Indizes unterscheiden, wird Feld genannt. Das Feld heißt ein-, zwei- oder mehrdimensional nach der Anzahl der Indizes. Die bekanntesten Felder sind wohl die zweidimensionalen, die in der Mathematik als Matrizen bezeichnet werden.
Alle Felder müssen vor ihrer Benutzung in einem Programm vereinbart werden. Die Vereinbarung lautet:

```
DIM variable (natürliche zahl, natürliche zahl . . .)
```

Für die Indizes können auch Variable und arithmetische Ausdrücke verwendet werden.

Beispiele:

DIM A(12) – DIM A1(1Ø) – DIM X(15,3Ø) – DIM F(M+2,N/2–1)

Die Dimensionierung ist nur erforderlich, wenn die Indizes > 1Ø sind. Durch die DIM-Anweisung werden alle Elemente des Feldes gleich Null gesetzt.

Die begrenzte Speicherkapazität des Rechners erfordert die Überprüfung, ob der freie Speicherplatz ausreicht. Die Anzahl der freien Speicherplätze wird mit

```
PRINT FRE(variable)
```

ausgegeben.

Durch DIM C(5,9) wird folgendes Variablenfeld mit 6Ø Elementen bereitgestellt.

C(Ø,Ø), C(Ø,1), C(Ø,2) . . . C(Ø,9)
C(1,Ø), C(1,1), C(1,2) . . . C(1,9)
.
.
.
C(5,Ø), C(5,1), C(5,2) . . . C(5,9)

Sie erkennen, der kleinste Index ist Null.

Dieses vereinbarte Feld kann im Programm mit C(I,K) benutzt werden und hat dadurch variable Indizes.

Bei einem zweidimensionalen Feld (Matrix) kann man die Anschauung erhöhen, indem man – wie dargestellt – den 1. Index als Zeilenzahl und den 2. als Spaltenzahl interpretiert:

C-(Zeilenzahl, Spaltenzahl) = C(Z,S).

Stellen Sie sich ein Hochlager mit 1Ø x 6 = 6Ø Ablagen vor. Das Lager soll also in 1Ø Zeilen und 6 Spalten aufgegliedert sein. Jede Ablage kann man mit 2 Zahlen (Zeile und Spalte) kennzeichnen. Will man ein solches Feld in BASIC benutzen, muß man es mit DIM A(9,5) vereinbaren, da die Zählung mit Null beginnt.
In der Praxis beginnt man die Zählung üblicherweise mit 1, dann muß man DIM A(1Ø,6) vereinbaren und benutzt Zeile und Spalte mit dem Index Null nicht. In diesem Sinne bedeutet A(9,4) die Ablage in der neunten Zeile von oben und der vierten Spalte von links.
Die Stückzahl der Gegenstände in dieser Ablage weisen wir nun der Variablen A(9,4) zu, d. h., wenn sich 32 Stück in dieser Ablage befinden, so schreiben wir A(9,4) = 32.

In den folgenden Beispielen werden die Variablen mit dem Index Null nicht benutzt!

Beispiel:

Wir weisen Feldvariablen beliebige positive Zahlen zu, indem wir ein kleines Eingabeprogramm mit Zählschleife benutzen.

```
1Ø DIM A(5,3)
2Ø   FOR S = 1 TO 3
3Ø     INPUT A(1,S)
4Ø   NEXT S
```

Mit diesem Programm werden den Elementen der 1. Zeile die Stückzahlen zugewiesen. Danach muß die Zeilenzahl um 1 erhöht werden. Dies kann durch eine 2. Zählschleife geschehen, die die erste vollständig umfaßt.

```
10 DIM A(5,3)
20 FOR Z = 1 TO 5
30   FOR S = 1 TO 3
40    INPUT A(Z,S)
50   NEXT S
60 NEXT Z
```

Die Vereinbarung des Feldes A ist nicht notwendig, da beide Indizes kleiner gleich 10 sind. Der Rechner reserviert automatisch Speicherplatz für ein Feld A(10,10) bei der ersten Benutzung eines Feldelementes (Zeile 40).

Beispiel:

Wir wollen nun das Maximum der Stückzahl bestimmen, indem wir das Feld zeilenweise durchmustern. In der 1. Zeile (Z = 1) suchen wir das Maximum, wobei – ausgehend von MAX = 0 – Zahl für Zahl geprüft wird, ob das Maximum kleiner als die zu vergleichende Stückzahl ist. Wenn das der Fall ist, wird diese Zahl das Maximum. Das geschieht durch IF MAX < A(Z,S) THEN MAX = A(Z,S)

Das elementweise Fortschreiten beim Vergleich erfolgt wieder durch 2 verschachtelte Zählschleifen wie bei der Eingabe. Der Anfangswert von MAX ist 0.
Das Programm wird ab Zeile 100 nach dem Eingabeprogramm geschrieben, so daß beide unmittelbar nacheinander ablaufen.

```
100 REM MAXIMUM EINES FELDES
110 MAX = 0
120 FOR Z = 1 TO 5
130   FOR S = 1 TO 3
140     IF MAX < A(Z,S) THEN MAX = A(Z,S) : PRINT Z, S, MAX
150   NEXT S
160 NEXT Z
170 PRINT "MAX="; MAX
```

Wenn wir noch wissen möchten, welches Element des Feldes das größte ist, müssen wir Z und S ausdrucken lassen! Dies wird in Zeile 140 realisiert.
Der letzte Tauschvorgang geschah bei der größten Zahl des Feldes. In der Tabelle können wir Z und S ablesen.

Übungen:

Ü 26. Schreiben Sie das Programm für folgenden Fall um:
Sie suchen das Minimum der Stückzahl und prüfen, ob es kleiner 10 ist, weil dann der Bestand im Lager aufzufüllen ist. Das ist solange zu wiederholen, bis alle Lagerplätze mit einer Stückzahl unter 10 gefunden sind.

Ü 27. Ändern Sie das Programm so um, daß nur der Lagerplatz mit der absolut größten Stückzahl ausgegeben wird.

Ein Feld – also die Belegung der Feldvariablen – kann unabhängig vom Programm gespeichert werden. Dazu ist

```
CSAVE* "name";variable
```

einzugeben und das Kassettengerät in den Aufnahmezustand zu bringen sowie die ENTER-Taste zu betätigen. Mit

```
CLOAD* "name";variable
```

können Sie das abgespeicherte Feld wieder einlesen.

Beispiel:

Wurde ein Feld mit DIM X (12,2Ø) vereinbart, so kann es durch

CSAVE* "DATEN";X

auf Magnetband gespeichert werden. Wobei Sie name selbst festlegen (1 . . .8 Zeichen). Im Beispiel wurde als name DATEN gewählt.

14. Zeichenketten

14.1. Aufbau von Zeichenketten

Zeichenketten, auch Strings genannt, sind Ihnen schon bekannt. Sie werden in Anführungszeichen eingeschlossen und können aus allen Zeichen des Zeichenvorrates bestehen. Wir nutzten sie bisher zu Erläuterungen bei der Ein- und Ausgabe hinter PRINT und INPUT, z. B. PRINT "SUMME ="; S.
Eine Zeichenkette kann 255 Zeichen lang sein. Es können Zeichenkettenausdrücke gebildet werden.
Mit Zeichenketten können auch Felder aufgebaut werden.

Sie werden durch

| DIM zeichenkettenvariable (natürliche zahl, natürliche zahl . . .)

vereinbart.

Z. B. DIM B$ (2Ø,15) – DIM C2$ (I,K)
Den so vereinbarten Zeichenkettenvariablen wird die leere Zeichenkette (" ") zugewiesen.

14.2. Wertzuweisung von Zeichenketten, Ein- und Ausgabe

Eine Zeichenkette kann einer Stringvariablen mit einer Ergibtanweisung zugewiesen werden. Die Stringvariable ist eine Variable, der noch zusätzlich $ angehängt wurde.
Stringvariable sind: A$, A1$, AØ$, CN$, Z$, A$(3)

Die Zuweisung geschieht durch:

```
A$   = "SO"
AØ$  = "WERDEN"
A1$  = "ZEICHENKETTEN"
A2$  = "STRINGVARIABLEN"
A3$  = "ZUGEWIESEN"
```

Die Zuweisung kann auch durch die Anweisung INPUT und Eingabe über die Tastatur geschehen.

Übung:

Ü 28. Geben Sie Ihren Namen auf die Anforderung ein, und lassen Sie sich den Namen wieder ausgeben.
Bei der Eingabe über die Tastatur können die " " entfallen.

```
1Ø INPUT "VORNAME ="; V$;
2Ø INPUT "FAMILIENNAME ="; F$
3Ø PRINT V$; " "; F$ oder
3Ø PRINT F$; ","; V$  oder
3Ø PRINT V$, F$
```

Eine Zeichenkette kann auch leer sein. Sie wird durch " " angegeben. Natürlich kann sie auch Zwischenräume (Leerzeichen) enthalten. Die Summe aller Zeichen, die den Zeichenkettenvariablen zugewiesen werden kann, muß < 257 sein. Der Bereich ist erweiterbar.

14.3. Relationen zwischen Zeichenketten

Strings kann man miteinander vergleichen und auch alphabetisch ordnen. Dazu dienen die Vergleichsoperatoren.

Z1$ = Z2$

2 Zeichenketten sind nur gleich, wenn sie vollkommen übereinstimmen.

Z3$ < Z4$

Eine Zeichenkette (Z3$) ist kleiner als eine andere (Z4$), wenn Z3$ bei alphabetischer Ordnung der Ausdrücke vor Z4$ steht. Dies wird realisiert, indem jedem Zeichen eine Codezahl zugeordnet wird, durch die der Vergleich realisiert wird.
So ist es möglich, Namen oder ein Wörterbuch in alphabetischer Reihenfolge aufzubauen.

Beispiele:

"Meier" < "MEYER"
"Wolf" < "Wolff"
"FUSZ" < "FUSZBALL"
"Reise" < "Riese"

14.4. Zeichenkettenfunktionen

Die Zeichenkettenfunktionen gestatten weitere Operationen mit Zeichenketten.

- Länge einer Zeichenkette:
  Durch LEN ("string") oder LEN (stringvariable) wird die Länge (Anzahl der Zeichen) der Zeichenkette bestimmt.

  Beispiel:
  A$ = "WERKZEUGNR. 3Ø5"
  Bestimmen Sie die Länge der Zeichenkette A$ mit
  PRINT LEN (A$)

- Umwandlung einer Zahl in eine Zeichenkette
  Durch STR$ (zahl) wird die Zahl zu einer Ziffernfolge, die man z. B. an einen Artikelnamen anhängen und insgesamt als Zeichenkette weiterverarbeiten kann.

- Zahlenwert einer Zeichenkette aus Ziffern
  Durch VAL ("zeichenkette") wird der Zahlenwert einer Ziffernfolge ermittelt. D. h., die Ziffernfolge wird in eine Zahl umgewandelt. Beginnt die Zeichenkette nicht mit einer Ziffer, einem Dezimalpunkt oder einem Vorzeichen, so wird der Wert Null angenommen.

  Eine vorgegebene Zeichenkette kann man auch zerlegen,

durch LEFT$ (zk, X)	in X Zeichen von links,
durch RIGHT$ (zk, X)	in X Zeichen von rechts,
durch MID$ (zk, X)	in alle Zeichen ab der Stelle X,
durch MID$ (zk, X, Y)	in Y Zeichen ab der Stelle X.

  Die Zeichenkette (zk) in der Klammer muß in Anführungszeichen stehen, sie kann auch durch eine Zeichenkettenvariable ersetzt werden.

Beispiel:

Von der Zeichenkette C$ = "LEHRBRIEF" läßt sich durch PRINT MID$ (C$,5) die Zeichenkette BRIEF abtrennen.
Dasselbe kann durch RIGHT$ ("LEHRBRIEF", 5) erreicht werden.

Durch die Zeichenkettenverarbeitung und durch die Benutzung von Feldern besteht die Möglichkeit, einfache Dateien aufzubauen und zu bearbeiten.

Beispiel:

Für den nachfolgend beschriebenen Lagerbestand wird eine kleine Datei aufgebaut.

Zeile/Spalte		Ø	1	2	3	4	5
Ø	Bezeichnung	Bolzen	Splint	Schraube	Scheibe	Mutter	Schraube
1	Teilnummer	3Ø1	3Ø5	Ø51	2Ø4	621	Ø61
2	Material	Stahl2	Stahl	Stahl1	Stahl	Stahl1	Stahl2
3	Anzahl	N1	N2	N3	N4	N5	N6

Ein Artikel oder Teil kann bei einer Bestellung durch die Teilnummer gekennzeichnet werden. Von dieser ausgehend kann der Artikel gesucht und seine gesamte Bezeichnung und der Bestand ausgegeben werden.
Dazu ist es günstig, die Tabelle in ein Zeichenkettenfeld zu überführen und damit zu arbeiten.
Zuerst muß das Zeichenkettenfeld[1] definiert werden mit DIM A$(3,5). Die Eingabe der einzelnen Zeichenketten (Feldelemente) kann wieder mit dem Eingabeprogramm aus dem Abschnitt Felder, aber mit Stringvariablen geschehen.

Beziehen wir uns wieder auf die Bestellung. Wenn die Teilnummer bekannt ist, soll die dazugehörige Spalte ausgegeben werden. Wir müssen nun die bestellte Teilnummer in der Zeile 1 suchen. Die Elemente dieser Zeile des Feldes werden mit A$(1,K) bezeichnet, wobei die Spaltenvariable K von Ø bis 5 läuft.

```
  5 REM EINGABE DER FELDELEMENTE
 1Ø DIM A$(3,5)
 2Ø FOR  Z = Ø TO 3                    Eingabeteil
 3Ø   FOR S = Ø TO 5                   mit Ausgabe der
 4Ø     PRINT Z,S                      Feldelementindizes
 5Ø     INPUT A$(Z,S)
 6Ø NEXT S, Z
 7Ø PRINT
 8Ø INPUT "TEILNR."; B$                Eingabe der Teilnummer
 9Ø FOR K = Ø TO 5                     Suchschleife für die Spaltennr. K
1ØØ   IF A$(1,K)=B$ THEN 11Ø:ELSE NEXT K
11Ø REM AUSGABE SPALTE K
12Ø FOR I = Ø TO 3                     Ausgabe aller Elemente
13Ø   PRINT A$(I,K); ".";              der Spalte K (Zeilen Ø . . . 3)
14Ø NEXT I
15Ø GOTO 7Ø
```

1) Auch Zeichenkettenfelder mit Indizes ≤ 10 brauchen nicht vereinbart werden.

Das Ausgabeprogramm wird mit GOTO 7Ø sofort zur nächsten Eingabe geführt, um weitere Aktionen auslösen zu können. Das Programm sollte nicht wieder mit RUN gestartet werden, weil dadurch die Belegungen der Feldvariablen gelöscht werden.

Übung:

Ü 29. Überlegen Sie, wie weitere Teile in zusätzliche Spalten aufgenommen werden können und wie die Feldindizes dadurch zu verändern sind.

14.5. Verknüpfung von Zeichenketten

Mit dem Operationszeichen + lassen sich Zeichenketten aneinanderreihen und neue bilden.
Mit den im Abschnitt 14.2. angegebenen Belegungen der Stringvariablen kann man durch

```
B$=A$+"  "+AØ$+"  "+A1$+"  "+A2$+"  "+A3$
```

einen Satz bilden und diesen durch

```
PRINT B$
```

ausgeben lassen.
Durch

```
| STRING$ (N, "zeichenkette")
```

können Zeichenketten oder einzelne Zeichen mehrfach (N-fach) wiederholt werden. Eine über die ganze Zeile gehende Unterstreichung lautet mit dieser Anweisung:

```
PRINT STRING$ (40, "–")
```

Übung:

Ü 30. Weisen Sie die obigen 5 Worte den Feldelementen A$(Ø) . . . A$(4) zu. Lassen Sie die 5 Strings nacheinander auf eine Zeile durch eine Zählschleife schreiben.

Zusammenfassung:

Stringausdrücke bestehen aus Zeichenketten, d. h. einer Folge von Zeichen und String-Variablen sowie String-Funktionen. Zeichenkettenausdrücke können durch Zeichenkettenfunktionen und den Operator + zu neuen Ausdrücken verarbeitet werden. Mit Hilfe der Vergleichsoperatoren kann man Strings vergleichen.

15. Zeichnen auf dem Bildschirm (Nicht für Z 9001)

Der Bildschirm besitzt ein Koordinatensystem. Der Nullpunkt liegt in der unteren linken Ecke. Nach rechts läuft die x-Koordinate von Ø bis 319, nach oben die y-Koordinate von Ø bis 255. Wir befinden uns also im 1. Quadranten dieses Koordinatensystems mit nur positiven ganzzahligen Koordinaten.

Alle x- und y-Werte aus Programmen werden nur ganzzahlig benutzt.
Wie Ihnen aus der Mathematik bekannt, kann der zum Wertepaar (x, y) gehörende Punkt auf dem Bildschirm dargestellt werden. Dazu ist nur eine Anweisung, die die Koordinaten enthält, nötig. Sie lautet

```
PSET X, Y, F
```

Die Zahl F in der Anweisung ist die Codezahl für die Vordergrundfarbe. Die Farbcodezahlen finden Sie im Anhang.

Wir zeichnen als erstes einen Punkt. Geben Sie dazu ein

```
10 PSET 100, 50, 7
```

und starten Sie dieses Miniprogramm durch RUN.

Nun wollen wir eine waagerechte Strecke zeichnen. Die x-Koordinate soll von 0 beginnend bis 300 laufen; y soll 100 sein. Das Programm lautet unter Verwendung der Zählschleife

```
10 FOR X = 0 TO 300
20  PSET X, 100, 7
30 NEXT X
```

Gezeichnete Kurven können durch

```
PRESET X, Y, F.
```

wieder gelöscht werden.

Übungen:

Ü 31. Verändern Sie in einem nächsten Schritt die Höhe (y-Koordinate) der Waagerechten.

Ü 32. Entwerfen Sie ein Programm zum Zeichnen einer Senkrechten. Beachten Sie die Bildschirmgrenzen!

Ü 33. Abschließend sollen Sie Bekanntes aus der Mathematik wieder aufgreifen, indem Sie die Zweipunkteform der Geradengleichung benutzen. Lösen Sie die Zweipunkteform nach y auf und wandeln die Gleichung in die BASIC-Schreibweise um, mit $x_1 \rightarrow$ X1 usw. Danach wählen Sie 2 Punkte im Koordinatensystem P_1 (x_1, y_1) und P_2 (x_2, y_2) und setzen die gewählten Koordinaten in die Gleichung ein. Abschließend schreiben Sie die Gleichung in die Zeile 30 des folgenden Programms.
Durch dieses Programm wird die x-Koordinate von 10 bis 300 mit einer Zählschleife variiert, die y-Koordinate berechnet und der dazugehörige Punkt gezeichnet.

```
10 REM ZEICHNEN EINER GERADEN
20 FOR X = 10 TO 300
30  Y =
40  PSET X, Y, 7
50 NEXT
```

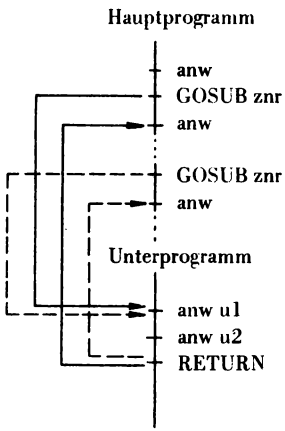
Ü 34. Geben Sie das Programm von Abschnitt 10.2.1. zur Wurfparabel ein und ersetzen Zeile 80 durch PSET 3 * X, 3 * Y,15. Der Faktor 3 ist wegen der in den Zeilen 10 . . . 30 festgelegten Parameter sinnvoll. Nach dem Programmstart wird die Wurfparabel im unteren Bildschirmteil punktweise gezeichnet.
In den Zeilen 20 und 30 können Sie durch eine INPUT-Anweisung die Möglichkeit schaffen, durch Eingaben die Wurfparameter zu ändern.
Natürlich sind die in den Übungen berechneten Koordinaten nicht ganzzahlig. Sie können durch X=INT(X) ganzzahlig gemacht werden, was nicht notwendig ist, weil der Rechner dies automatisch vornimmt.

16. Unterprogramme

In manchen Programmen treten häufig gleiche Programmteile auf. Diese Programmteile braucht man nicht wiederholt aufzuschreiben. Sie werden als Unterprogramm (UP) nur einmal formuliert und können durch eine Anweisung wiederholt abgearbeitet werden. Nach dem Durchlaufen des UP wird der Ablauf des Hauptprogramms dort fortgesetzt, wo die Unterbrechung erfolgte.
Die Anweisungen, mit denen dies realisiert wird, lauten:

| GOSUB zeilennummer und RETURN

Dadurch wird folgender Ablauf realisiert:



Hinweis:

Ein UP darf nicht durch GOTO angesprungen werden, da der Rechner die Fortsetzungsstelle nicht kennt, sobald er RETURN erreicht.

Beispiel:

Wir wollen dieses Vorgehen bei der Berechnung von Nullstellen von mathematischen Funktionen demonstrieren. Das Hauptprogramm enthält das mathematische Verfahren, das Unterprogramm dient zur Berechnung der notwendigen Funktionswerte der zu untersuchenden Funktion. Diese Funktion muß noch in das UP geschrieben werden.

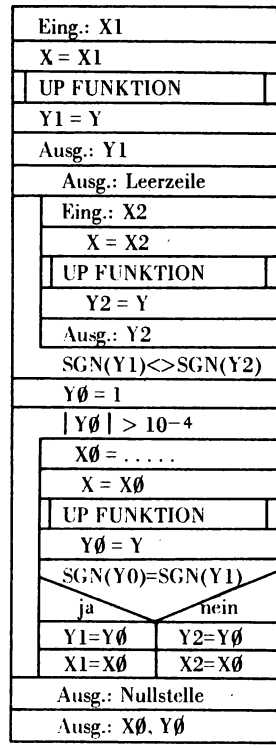
Wir wählen $y = e^{-x} + \frac{x}{5} - 1$

und beginnen das UP bei Zeile 5000. In die folgende Zeile kommt RETURN.

Hauptprogramm

```
 1Ø INPUT "X1 ="; X1
 2Ø X = X1
 3Ø GOSUB 5ØØ
 4Ø Y1 = Y
 5Ø PRINT "Y1 ="; Y1
 55 PRINT
 6Ø INPUT "X2 ="; X2
 7Ø X = X2
 8Ø GOSUB 5ØØ
 9Ø Y2 = Y
1ØØ PRINT "Y2 ="; Y2
11Ø IF SGN(Y2) <> SGN(Y1) THEN 12Ø
    : ELSE PRINT "NEUEINGABE FUER X2" : GOTO 6Ø
12Ø YØ = 1
13Ø IF ABS(YØ) > 1E–4 THEN 140 : ELSE 2ØØ
14Ø XØ = X1 – Y1 * (X2 – X1) / (Y2 – Y1)
15Ø X = XØ
16Ø GOSUB 5ØØ
17Ø YØ = Y
18Ø IF SGN (YØ) = SGN (Y1)  THEN  Y1  =  YØ: X1  =  XØ
                            : ELSE  Y2  =  YØ: X2  =  XØ
19Ø GOTO 13Ø
2ØØ PRINT "NULLSTELLE : XØ ="; XØ;
    "YØ ="; YØ : END
    Unterprogramm
5ØØ Y = EXP (–X) + X/5 – 1
51Ø RETURN
```

Struktogramm



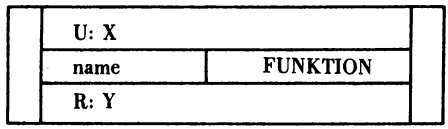
Die Programmnutzung gestaltet sich wie folgt:
Es müssen 2 X-Werte eingegeben werden, X1 und X2. Die dazu erfolgende Berechnung der Y-Werte im UP muß für Y1 und Y2 unterschiedliche Vorzeichen ergeben. Solange dies nicht der Fall ist, muß X2 neu eingegeben werden. Sind die unterschiedlichen Vorzeichen bei Y1 und Y2 vorhanden, berechnet das Programm die Nullstelle und druckt XØ und YØ aus. Je nach Lage der vorgegebenen Punkte kann die Berechnung kurze Zeit dauern. Die geforderte Genauigkeit liegt für YØ unter $1 \cdot 10^{-4}$. Das Hauptprogramm muß mit END abgeschlossen werden, wenn das Unterprogramm hinter dem Hauptprogramm steht.

Übung:

Ü 35. Suchen Sie mit dem Programm und der vorgegebenen Funktion eine Nullstelle.

Im Zusammenwirken von Haupt- und Unterprogramm spielt die Parameterübergabe eine wichtige Rolle. Wenn im UP die Funktionswerte Y berechnet werden sollen, muß das HP die dazu benötigten X-Werte bereitstellen, die das UP zur Berechnung der Y-Werte nutzt und sie an das HP übergibt.

Im Struktogramm kann man das wie folgt berücksichtigen.

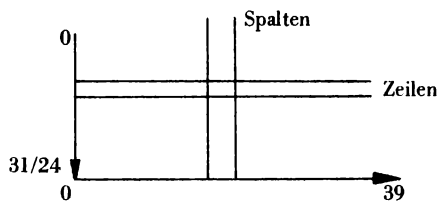


Zur Übergabe kommt X, zur Rückgabe kommt Y.
Das UP hat die Funktion, den Y- oder Funktionswert zu berechnen. Als Name erhält es FUNKTION.

17. Ausgabegestaltung

17.1. Einrichten eines Fensters

Die Gestaltung des Ausgabebildes und die Anordnung der auszugebenden Zahlen, Texte und Zeichnungen sind für den Nutzer von Programmen von großer Bedeutung.
Es ist möglich erläuternden Text und Berechnungen oder grafische Darstellungen vollkommen zu trennen, indem Fenster auf dem Bildschirm eingerichtet werden. Auf die Fläche eines Fensters kann z. B. der Text beschränkt werden.
Um die Fenster einfach einrichten zu können, ist der Bildschirm in Zeilen und Spalten aufgeteilt.



Die Kleincomputer haben unterschiedliche Zeilenzahlen:
KC 85/1 (Z 9001) 0 . . . 23 und KC 85/2 0 . . . 31 Zeilen.
Die Spaltenzahl geht bei beiden von Ø . . . 39.
Die Fenstereinteilung wird durch

```
| WINDOW anfangszeile, endzeile, anfangsspalte, endspalte
```

vorgenommen. Danach werden alle Ausgaben nur in das eröffnete Fenster geschrieben. Der restliche Bildschirm bleibt unverändert.

Beispiele (bezogen auf 32 Zeilen):

WINDOW Ø,15,Ø,39	Das Fenster nimmt die obere Bildhälfte ein.
WINDOW Ø,31,Ø,19	Das Fenster nimmt die linke Bildhälfte ein.
WINDOW 16,31,20,39	Das Fenster nimmt das untere rechte Viertel ein.

Durch WINDOW (ohne Zusatz) wird die normale Bildschirmfläche freigegeben (Grundeinstellung).
CLS nach WINDOW löscht nur die freigegebene Fläche des Fensters.

17.2. Textpositionierung

An beliebige Stelle des Bildschirms außerhalb eines Fensters (und auch in ihm) können Zahlen und Texte mit Hilfe von

```
PRINT AT (zeile, spalte); ausgabeliste
```

geschrieben werden. Danach wird im Fenster an der letzten aktutellen Position weitergeschrieben.

Beispiel:

An die x-Achse eines Koordinatensystems, erzeugt beim

KC 85/1

durch

```
10  FOR I=0 TO 35
20    PRINT AT (21,I);"-"
30  NEXT I
```

KC 85/2

```
10  FOR I=0 TO 300
20    PSET I, 17,7
30  NEXT I
```

kann mit

```
40  PRINT AT (22,30); "EINHEIT"
```

```
40  PRINT AT (30,30); "EINHEIT"
```

die physikalische Einheit geschrieben werden.

Aber auch im Fenster kann der Cursor an bestimmte Stellen gebracht und dort mit PRINT weitergeschrieben werden. Die Anweisung lautet:

```
LOCATE zeile, spalte
```

Die Zählung der Zeilen und Spalten für die Anweisung LOCATE beginnt im Fenster wieder mit Null. Beim Erzeugen von Tabellen kann in bestimmte Spalten ausgegeben werden. Mit der Anweisung

```
PRINT TAB (spalte);ausdruck
```

kann der Ausdruck in die gewünschte Spalte vom linken Bildschirmrand geschrieben werden. Das folgende Beispiel veranschaulicht die Ausgabespalten.

Beispiel:

```
10 FOR I=1 TO 28
20   PRINT TAB (I);"*"
30 NEXT I
```

Die Zunahme der Variablen I bewirkt auf jeder neuen Zeile die Vergrößerung der Spaltenzahl

Um einen gewünschten Abstand von auf der Zeile schon vorhandenen Zeichen zu erreichen verwendet man:

```
PRINT SPC (N);ausdruck
```

Wobei N die Anzahl der Leerzeichen (Spalten) zwischen der letzten Ausgabe und dem folgenden Ausdruck festlegt.

Beispiel:

```
10 FOR I=26 TO 1 STEP -1
20   PRINT"XXX";SPC(I);"000"
30 NEXT I
```

17.3. Farbgestaltung

17.3.1. Farben auf dem gesamten Bildschirm

Bei entsprechender Ausrüstung kann der Bildschirm auch farblich gestaltet werden. Es kann durch die Farbanweisungen

```
INK farbcode
```

die Vordergrundfarbe (Buchstabenfarbe) und durch

```
PAPER farbcode
```

die Hintergrundfarbe eingestellt werden. Diese Anweisungen dürfen zusammen oder nacheinander in beliebiger Reihenfolge benutzt werden. Die Farbeinstellung wird bis zu ihrer Änderung für den ganzen Bildschirm beibehalten. CLS nach einer Farbanweisung bewirkt, daß der ganze Bildschirm von oben beginnend eingefärbt wird.

Zusätzlich bietet der

KC 85/1	KC 85/2
durch	
BORDER farbcode	COLOR v, h
die Einstellung einer Rand-	die gleichzeitige Einstellung der
farbe des Bildschirms	Vorder- und Hintergrundfarbe.

Der Farbcode ist für beide Kleincomputer unterschiedlich und dem Anhang zu entnehmen.

Beispiel:

KC 85/1

```
10 PAPER 1:CLS
```

Der gesamte Bildschirm wird eingefärbt.

```
20 INK 2
30 PRINT "ROT AUF SCHWARZ"
```

KC 85/2

```
10 PAPER 1:CLS
```

Der gesamte Bildschirm wird eingefärbt.

```
20 INK 7
30 PRINT "WEISZ AUF BLAU"
```

Alle folgenden Texte werden in den angegebenen Farben ausgegeben.

17.3.2. Lokale Farben

Einzelne farbliche Flächen oder Texte, die von der für den gesamten Bildschirm gültigen Farbe abweichen, können mit Farbanweisungen hinter PRINT erzielt werden.

```
PRINT farbanweisungen;ausgabeliste
PRINT farbanweisungen;AT (zeile, spalte);ausgabeliste
```

Nach Ausführung der PRINT-Anweisung gelten die vorher angewiesenen Farben!

Beispiel für KC 85/2:

```
10 PRINT INK 4, PAPER 7; AT (15,7); "GRUEN AUF GRAU BEIM KC 85/2"
20 PRINT INK 13; "NEUE VORDERGRUNDFARBE";
30 PRINT "URSPRUENGLICHE FARBE"
```

Literaturverzeichnis

Strukturierte Programmierung. – Ottomar Herrlich; Ulrich Lindner: Schriftenreihe Informationsverarbeitung. – Teubner-Verlagsgesellschaft, Leipzig 1981

Software-Entwurf. – Jörg Schumann; Manfred Gerisch: Verlag Technik, Berlin 1984

Einführung in BASIC. – Egbert Voigt: radio fernsehen elektronik, 33 (1984) 479, H. 8

Programmierung mit BASIC. – Siegmar Müller: Reihe Automatisierungstechnik Band 216. – Verlag Technik, Berlin 1985

Die Programmiersprache BASIC. – Helmut Adler; Hans-Ulrich Karl: Lehrbrief für das Hochschulfernstudium. – Herausgeber: Zentralstelle für das Hochschulfernstudium, Dresden 1985

ANHANG:

Anlage 1

Zusammenstellung aller verwendeten Schlüsselworte und Kommandos

Die Aufteilung in Kommandos und Schlüsselworte ist nicht starr. Es können einige Kommandos auch in Programmen verwendet werden. Ebenso können verschiedene Schlüsselworte im Direktmodus verwendet werden.

Kommandos	Bemerkung
AUTO	Selbständige Zeilennumerierung;
CLEAR	Löscht den Variablenspeicher;
CLOAD	Lädt ein Programm von der Kassette
CLS	Löscht den Bildschirm;
CONT	Setzt ein unterbrochenes Programm fort;
CSAVE	Abspeichern eines Programmes auf Kassette;
DELETE	Zeilen löschen;
EDIT	Programmkorrektur;
LIST	Listet das Programm auf;
RENUMBER	Neumuerierung der Programmzeilen;
RUN	Startet das Progamm und führt es aus;
Schlüsselworte	
DATA	Folge der durch READ zu lesenden Werte;
DIM	Dimensioniert Variablenfelder;
END	Beendet das Programm;
FOR TO STEP	Festlegung einer Programmschleife;
GOSUB	Unterprogrammaufruf;
GOTO	Unbedingte Sprunganweisung;
IF	Bedingte Sprung- oder Handlungsanweisung;
INPUT	Gibt ein "? " aus und wartet auf eine Eingabe;
LET	Wertzuweisung;
LOCATE	Positioniert den Cursor im Fenster;
NEW	Löscht den Programm- und Variablenspeicher;
NEXT	Ende der Programmschleife;
ON GOTO	Mehrfache Programmverzweigung;
ON GOSUB	Mehrfache Programmverzweigung;
PRINT	Ausgabe auf Bildschirm
READ	Ordnet den hinter READ stehenden Variablen die hinter DATA stehenden Werte zu;
REM	Kommentarkennzeichnung;
RESTORE	Setzt den DATA-Zeiger auf den Anfang der Liste;
RETURN	Ende des Unterprogrammes;
STOP	Stoppt ein Programm;

Kommandos	Bemerkung
Mathematische Funktionen	
ABS(X)	Betrag von x;
ATN(X)	arctan x , Resultat im Bogenmaß;
COS(X)	x im Bogenmaß;
EXP(X)	e^x, x < = 87.3365;
INT(X)	größte ganze Zahl < = x
LN(X)	Logarithmus von x;
PI	PI = 3.14159;
SGN(X)	Signumfunktion;
SIN(X)	x im Bogenmaß;
SQR(X)	$\sqrt{x}$;
TAN(X)	x im Bogenmaß;
String-Funktionen	
LEFT$(A$, X)	Liefert die ersten X Zeichen von A$;
LEN (A$)	Zeichenlänge des Strings A$;
MID$(A$, X,Y)	Y Zeichen von A$, beginnend mit dem X-ten;
RIGHT$(A$, X)	Liefert die letzten X Zeichen von A$;
STR$(X)	Formt den Wert X in einen String um;
VAL (A$)	Numerischer Wert von A$;
STRING$(N, A$)	Vervielfacht A$
Sonstige Funktionen	
AT (Z,S)	Positionierungsfunktion
FRE(X)	Gibt die Größe des noch freien RAM-Speicherplatzes an;
SPC (I)	Formatierungsfunktion;
TAB (I)	Formatierungsfunktion;

Anlage 2

Farbcode für KC 85/1 und KC 85/2

KC 85/1 Code	KC 85/2	Vorder- und Hintergrundfarbe	Nur KC 85/2 Code	Vordergrundfarbe
1	Ø	Schwarz	8	Schwarz
5	1	Blau	9	Violett
2	2	Rot	10	Orange
6	3	Purpur	11	Purpurrot
3	4	Gruen	12	Gruenblau
7	5	Tuerkis	13	Blaugruen
4	6	Gelb	14	Gelbgruen
8	7	Weiss	15	Weiss
			Farbcode + 16	Vordergrundfarbe blinkend

Die Vordergrundfarben sind deutlich heller als die Hintergrundfarben.

Lister der Fehlerhinweise

Fehlermeldungen während der Programmabarbeitung geschehen durch

XX ERROR IN zeilennummer
└ Fehlerhinweis

Der Rechner geht in den Direktmodus über.

Fehlerhinweise:

BS	Feldelement außerhalb des dimensionierten Bereiches
CN	Programm kann nicht mit CONT fortgesetzt werden
DD	Feld mehrfach dimensioniert
FC	Unzulässiger Funktionsaufruf
ID	Fehlerhafte Eingabe im Direktbetrieb
LS	Zeichenkette länger als 255 Zeichen
MO	Anweisung unvollständig
NF	Variablen hinter FOR und NEXT passen nicht zusammen
OD	In der DATA-Anweisung wurden zu wenig Daten spezifiziert
OM	Speicherplatz im RAM reicht nicht aus
OS	Speicherplatz für Zeichenkette reicht nicht aus
OV	Zahlenbereich wird überschritten
RG	RETURN tritt vor GOSUB auf
SN	Syntaktischer Fehler
ST	Zeichenkette zu lang oder zu komplex
TM	Variablentyp und Datentyp passen nicht zusammen
UF	Funktion nicht definiert
UL	Es wurde eine nicht existente Zeilennummer angegeben
/Ø	Division durch Null