

**de Gruyter Lehrbuch**  
**Mägerle · Programmieren in BASIC**



**Einführung in das**

# **Programmieren in BASIC**

von

**Erich W. Mägerle**



**Walter de Gruyter · Berlin · New York 1974**

©

Copyright 1974 by Walter de Gruyter & Co., vormals G. J. Göschen'sche Verlagshandlung, J. Guttentag, Verlagsbuchhandlung Georg Reimer, Karl J. Trübner, Veit & Comp., Berlin 30.

Alle Rechte, insbesondere das Recht der Vervielfältigung und Verbreitung sowie der Übersetzung, vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (durch Photokopie, Mikrofilm oder ein anderes Verfahren) ohne schriftliche Genehmigung des Verlages reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Printed in Germany.

Satz: Composer Walter de Gruyter & Co., Berlin. – Druck: Mercedes-Druck, Berlin

Bindearbeiten: Wübben & Co., Berlin

Library of congress catalog card number 74-76079.

ISBN 3 11 004801 9

# Inhaltsverzeichnis

0. Einleitung	7
1. BASIC-Ausdrücke	13
1.1 Konstanten	13
1.1.1 Arithmetische Konstanten	13
1.1.2 Zeichenkonstanten	14
1.1.3 Systemkonstanten	14
1.2 Variablen	14
1.2.1 Arithmetische Variable	14
1.2.2 Zeichenvariable	15
1.2.3 Arithmetische Bereichsvariablen	15
1.2.4 Zeichenbereichsvariablen	15
1.3 Arithmetische Formeln	16
2. Eingabe-Anweisungen	18
2.1 INPUT-Anweisung	18
2.2 MAT INPUT-Anweisung	19
2.3 DATA-Anweisung	21
2.4 READ-Anweisung	21
2.5 MAT READ-Anweisung	23
2.6 RESTORE-Anweisung	24
2.7 GET-Anweisung	25
2.8 MAT GET-Anweisung	27
2.9 RESET-Anweisung	28
2.10 CLOSE-Anweisung	28
3. Ausgabe-Anweisungen	30
3.1 PRINT-Anweisung	30
3.2 PRINT USING-Anweisung	34
3.3 MAT PRINT-Anweisung	39
3.4 MAT PRINT USING-Anweisung	40
3.5 PUT-Anweisung	41
3.6 MAT PUT-Anweisung	41
4. ERGIBT-Anweisungen	43
4.1 LET-Anweisung	43
4.2 Auflösung arithmetischer Formeln	44
4.3 DEF-Anweisung	45
4.4 Die Benutzung von Systemfunktionen	47
4.5 MAT ERGIBT-Anweisung	49
4.5.1 Matrix-Addition	50
4.5.2 Einsermatrix	50
4.5.3 Identitätsmatrix	51
4.5.4 Matrix-Inversion	51
4.5.5 Matrix-Multiplikation	52
4.5.6 Skalare Matrix-Multiplikation	53
4.5.7 Matrix-Subtraktion	53
4.5.8 Transponieren einer Matrix	54
4.5.9 Nullmatrix	54
5. Schleifen-Steuerung-Anweisung	
5.1 FOR- und NEXT-Anweisung	56
5.2 Schachtelung von FOR-Schleifen	57

6. Verzweigungs-Anweisungen . . . . .	59
6.1 GO TO-Anweisung . . . . .	59
6.2 Computed GO TO-Anweisung . . . . .	59
6.3 IF-Anweisung . . . . .	60
7. Das Definieren von Bereichen . . . . .	62
7.1 DIM-Anweisung . . . . .	62
8. Kommentierung eines Programmes . . . . .	63
8.1 REM-Anweisung . . . . .	63
9. Hinzufügen von Programmsegmenten . . . . .	65
9.1 GOSUB- und RETURN-Anweisung . . . . .	65
10. Stoppen der Programmausführung . . . . .	66
10.1 END-Anweisung . . . . .	66
10.2 STOP-Anweisung . . . . .	66
10.3 PAUSE-Anweisung . . . . .	67
11. Verbindung von Hauptprogrammen . . . . .	68
11.1 COM-Anweisung . . . . .	68
11.2 PICK-Anweisung . . . . .	70
12. Anhang . . . . .	71
12.1 Zusammenfassung der BASIC-Anweisungen . . . . .	71
12.2 Übersicht der Systemfunktionen . . . . .	77
12.3 Übersicht der BASIC-Ausdrücke . . . . .	81
12.4 System-Konstanten . . . . .	81
12.5 Arithmetische Operatoren . . . . .	82
12.6 Syntax-Symbole . . . . .	82
12.7 Ausgetestete Programmierbeispiele . . . . .	83
12.8 Aufgaben zum Selbstlösen . . . . .	95
Stichwortverzeichnis . . . . .	111

# Einleitung

BASIC entstand Anfang der siebziger Jahre und zählt zu den jüngsten problemorientierten Programmiersprache. Sie ist in eine Reihe etwa mit PL/1 und FORTRAN einzustufen, läßt sich jedoch bedeutend einfacher erlernen und ist außerdem wegen des geringeren Programmieraufwandes wirtschaftlicher als andere Programmiersprachen.

BASIC eignet sich besonders für die Lösung technisch-wissenschaftlicher Probleme und wird deshalb vornehmlich von Ingenieuren, Konstrukteuren, Physikern, Chemikern, Mathematikern und von Statistikern und Planern angewandt. Jedoch lassen sich auch Finanz-, Planungs- und Budgetprobleme mit dieser Sprache formulieren. Nicht zuletzt ist sie nutzbar für vielfältigste Routine-Berechnungen, die häufig einen großen Zeitaufwand erfordern.

BASIC ist eine Dialogsprache. Spezielle Befehle ermöglichen Dateneingabe über Datenstationen während der Programmdurchführung. In der Matrizenrechnung ist BASIC besonders gut ausgebaut. Mit einem *einzigem* Befehl können folgende Matrix-Operationen ausgeführt werden:

- Addition
- Subtraktion
- Multiplikation
- Skalare Multiplikation
- Transponieren
- Inversion
- Identitätsmatrix
- Einsermatrix
- Matrixeingabe über Datenstation oder als Datei
- Matrixausgabe drucken oder abspeichern

BASIC kann auf verschiedenen Betriebssystemen verwendet werden. Erwähnt seien hier CALL/360 und S/3–6 IBM. In diesem Buch wird jedoch kein bestimmtes Betriebssystem behandelt, da dieses vom jeweils verwendeten Computer abhängt.

## Programmiersprachen

Grundsätzlich werden zwei Gruppen von Programmiersprachen unterschieden:

### a) *Maschinenorientierte Programmiersprachen*

Hier sind die Instruktionen in Ablehnung an die technische Struktur der Maschine aufgebaut. Sie bestehen aus hexadezimalen Codes, die folgende Aussagen liefern müssen:

WAS	WIEVIEL	WOHIN	WOHER
-----	---------	-------	-------

WAS	ist zu tun (auszuführende Operation)	①
WIEVIEL	Bytes sind zu verarbeiten (Feldlänge)	②
WOHIN	kommen die Daten (Adresse von Feld 1)	③
WOHER	kommen die Daten (Adresse von Feld 2)	④

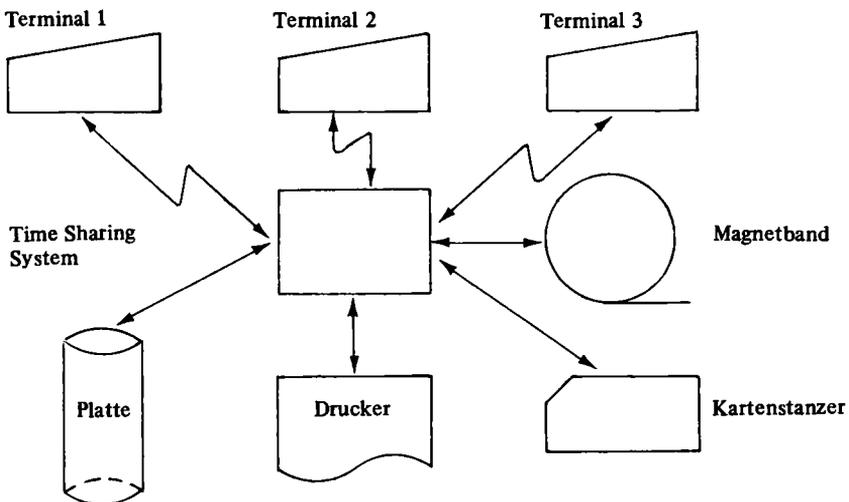
Allerdings sind solche Programme, bei denen die Rechenoperationen maschinenintern geschrieben werden, sehr aufwendig und setzen detaillierte Kenntnisse des technischen Aufbaus der Maschine voraus. Man hat deshalb Sprachen entwickelt, die sich weniger an der Maschine als vielmehr am zu lösenden Problem orientieren:

### b) *Problemorientierte Programmiersprachen*

Zu diesen Sprachen gehört auch BASIC. Sie haben den großen Vorteil, daß symbolische Formulierungen verwendet werden können, die der üblichen Sprachregelung ähnlich sind. Eine Multiplikation z.B. hat die gleiche Form wie in der Mathematik, etwa:  $X = A * F$ , usw.

### Dialog mit dem Computer

BASIC ermöglicht weiterhin als sog. Dialogsprache ein Frag-Antwort-Verfahren mit dem Computer. Voraussetzung dafür ist die Verwendung von Betriebssystemen, die nach dem Time Sharing-Prinzip gestaltet sind. Es sind dies Teilnehmersysteme, in denen der Computer gleichzeitig für mehrere Benutzer arbeitet. Möglich wird das durch räumliche Aufteilung der Arbeitsspeicher und Zerlegung jeder Arbeit in kleinste Abschnitte. So können die verschiedenen Arbeiten in kurzen Zeiträumen abwechselnd abgewickelt werden. Durch die hohe Arbeitsgeschwindigkeit und die langsame Datenübertragung der Telefonleitung entsteht der Eindruck der Gleichzeitigkeit, die eine gleichzeitige Benutzung des Computers über verschiedene Datenstationen (Terminal) gestattet:



### **Problemstellung und Lösung**

Nachstehend wird gezeigt, wie man z. B. ein mathematisches Problem anpacken muß, damit ein BASIC-Programm aufgebaut werden kann (kein allgemeingültiges „Rezept“):

1. *Problem definieren = Analyse*

Festlegen der bekannten und unbekanntenen Größen. Randbedingungen formulieren.

2. *Mathematische Formulierung*

Zusammenstellen der Berechnungsformeln. Dazu gehört auch die Formulierung von Algorithmen bzw. die entsprechenden Iterationen.

(Beispiel: Mit Rechenanlagen läßt sich eine Integration nicht durchführen. Es muß daher ein Näherungsverfahren gesucht werden z. B. Trapezregel.)

3. *Blockdiagramm erstellen*

Rechenanlagen können jeden *logischen* Schritt ausführen. Diese Logik muß in einem Blockdiagramm dargestellt werden. Gerade in diesem Schritt lohnt es sich, sehr genau zu arbeiten.

4. *Programm codieren*

Mit Hilfe des erstellten Blockdiagramms wird das Programm erstellt (vercoden). Dieses BASIC-Programm nennt man Quellen- oder Sourceprogramm.

5. *Eingabe an der Datenstation*

Das codierte BASIC-Programm kann direkt an der Datenstation eingetippt werden. Dies ist ein weiterer Vorteil von BASIC und dem Time Sharing-Verfahren.

6. *Maschinenprogramm erstellen*

Das erstellte Programm ist jetzt in einer persönlichen Bibliothek im Rechenzentrum abgespeichert und zu jedem Zeitpunkt sofort aufrufbar. Dieses Quellenprogramm muß in eine für die Rechenanlage gerechte Form umgewandelt werden mit Hilfe des Compilers. Es entsteht damit das Maschinenprogramm, auch Objectprogramm genannt. Sind im BASIC-Programm formale Fehler (Syntax) vorhanden, so werden diese auf der Datenstation gedruckt. Die Erklärung dazu ermöglicht die Korrektur der Fehler. Dieser Schritt muß wiederholt werden, bis keine Fehlermeldungen mehr entstehen.

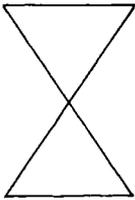
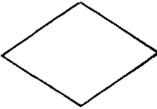
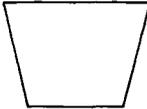
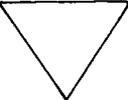
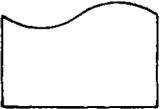
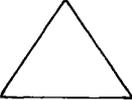
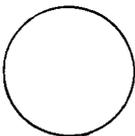
7. *Testen auf logische Fehler*

Ein formal fehlerfreies Programm gibt uns noch keine Gewißheit, daß es auch in der Logik „stubenrein“ ist. Man gibt dem Programm nun Eingabedaten, damit eine Berechnung durchgeführt werden kann. Das gleiche Beispiel muß auch „von Hand“ berechnet werden. Stimmen die Resultate überein, so sind keine logischen Fehler vorhanden.

8. *Durchlauf aktueller Berechnungen = Produktion*

In diesem Schritt hilft die Rechenanlage immer wiederkehrende Arbeiten sicher und genau durchzuführen. Der Zeitgewinn hier kann die vorangegangenen Arbeiten kompensieren, da ein Programm immer wieder verwendbar ist.

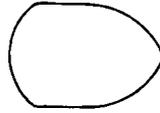
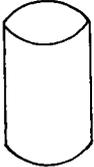
**Verwendete Symbole***Symbole für die Darstellung von Programmablaufplänen*

	Verarbeitung		Ableichen mehrerer Daten zu einem einzigen Teil (Collate)
	Verzweigen, Entscheid		Datenfernver- arbeitung
	Verwendung von Subroutinen		Anschluß-Stelle, Verbindungspunkt
	Eingabe/Ausgabe		Anfang, Ende, Stop
	Manuelle Eingabe im Zeitpunkt der Verarbeitung		Lochkarte
	Eingreifen von Hand		Lochstreifen
	Mischen		Liste
	Extrahieren, Selektieren		Magnetband

Einleitung



Magnettrommel

Display  
(Bildschirm)

Magnetplatte



Flußlinie

### Mathematischen Symbole

+	Addition, positives Vorzeichen	<	kleiner
-	Subtraktion, negatives Vorzeichen	≤	kleiner oder gleich
*	Multiplikation	=	gleich
/	Division	≥	größer oder gleich
↑ oder **	Potenzieren	>	größer
Σ	Summe	≠	ungleich
		→	verschieben nach

