de Gruyter Lehrbuch
Mägerle · Programmieren in BASIC

Einführung in das

Programmieren in BASIC

von

Erich W. Mägerle

Walter de Gruyter · Berlin · New York 1974

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

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 *einzigen* 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

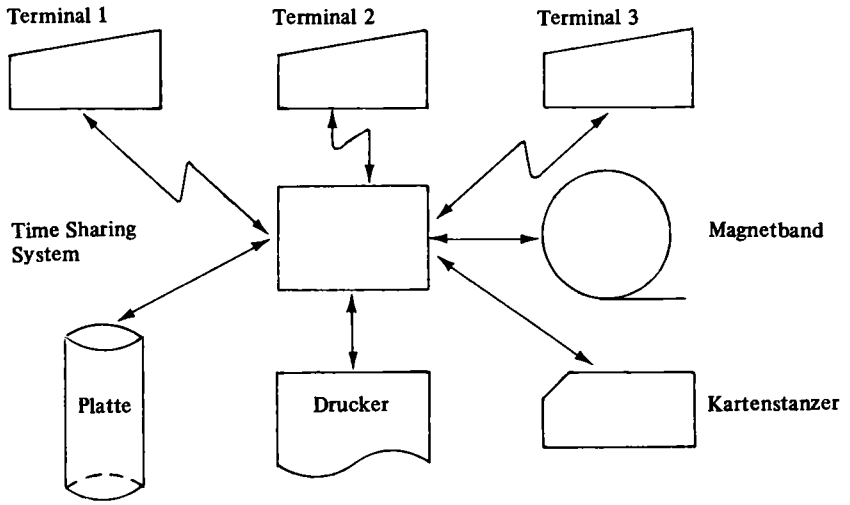
Allerdings sind solche Programme, bei denen die Rechenoperationen maschinenintern geschrieben werden, sehr aufwendig und setzen detaillierte Kenntnisse des technischen Aufbaus der Maschine voraus. Man hat deshalb Sprachen entwickelt, die sich weniger an der Maschine als vielmehr am zu lösenden Problem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

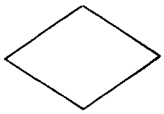
Nachstehend wird gezeigt, wie man z. B. ein mathematisches Problem anpacken muß, damit ein BASIC-Programm aufgebaut werden kann (kein allgemeingültiges „Rezept"):

1. *Problem definieren = Analyse*
   Festlegen der bekannten und unbekannten 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 ein 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 e 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

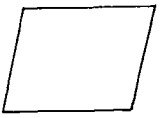
Symbole für die Darstellung von Programmablaufplänen

Verarbeitung

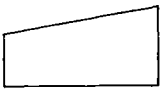


Verzweigen, Entscheid

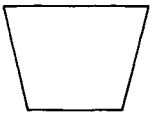
Verwendung von Subroutinen



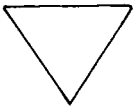
Eingabe/Ausgabe



Manuelle Eingabe im Zeitpunkt der Verarbeitung

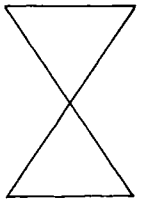


Eingreifen von Hand

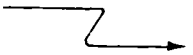


Mischen

Extrahieren, Selektieren



Abgleichen mehrerer Daten zu einem einzigen Teil (Collate)



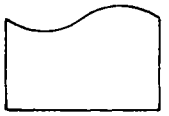
Datenfernverarbeitung

Anschluß-Stelle, Verbindungspunkt

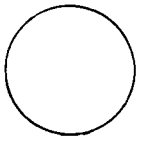
Anfang, Ende, Stop

Lochkarte

Lochstreifen

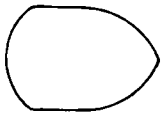


Liste

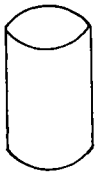


Magnetband

Magnettrommel



Display
(Bildschirm)



Magnetplatte

Flußlinie

Mathematischen Symbole

+	Addition, positives Vorzeichen	<	kleiner
–	Subraktion, negatives Vorzeichen	⩽	kleiner oder gleich
		=	gleich
*	Multiplikation	⩾	größer oder gleich
/	Division	>	größer
↑ oder **	Potenzieren	≠	ungleich
Σ	Summe	→	verschieben nach