APL

Problemorientierte Einführung

Von
Dr. Michael A. Curth
und
Dr. Helmut Edelmann

2., erweiterte Auflage

R. Oldenbourg Verlag München Wien

Für Karin und Renate & Heinz

CIP-Titelaufnahme der Deutschen Bibliothek

Curth, Michael A.:
APL : problemorientierte Einf. / von Michael A. Curth u. Helmut Edelmann. – 2., erw. Aufl. – München ; Wien : Oldenbourg, 1988
ISBN 3-486-20814-4

NE: Edelmann, Helmut:

Gesamtherstellung: Huber KG, Dießen

ISBN 3-486-20814-4

Inhaltsverzeichnis

Vorwort zur 2. Auflage

Die Grundideen dieser problemorientierten Einführung in die Programmiersprache APL, nach denen bereits die erste Auflage konzipiert worden ist, haben sich bewährt, so daß dieselben auch für diese zweite Auflage beibehalten werden konnten.

Wir nutzen im Rahmen dieser zweiten Auflage die Gelegenheit, die wohl nie ganz "ausrottbaren" Fehler zu korrigieren, die sich trotz intensiver Bemühungen in die Erstauflage eingeschlichen haben. Zudem werden einige im Laufe der Zeit nötig gewordenen Aktualisierungen vorgenommen. Dies geschieht auch im Interesse eines größeren englischen Verlagshauses, das sich dazu entschlossen hat, eine Übersetzung des Werkes für den angelsächsischen Sprachraum herauszugeben. Eine wesentliche Erweiterung der zweiten Auflage bildet das Kapitel zur Sprachversion APL2, die seit Erscheinen des Buches im Jahre 1986 stark an Bedeutung zugenommen hat.

Allen, die uns Anregungen und Hinweise für die Neuauflage gegeben haben, möchten wir an dieser Stelle herzlich danken.

Michael A. Curth
Helmut Edelmann

Vorwort zur 1. Auflage

APL ist eine Programmier- bzw. Planungssprache, die in der betrieblichen und universitären Datenverarbeitung zunehmend an Bedeutung gewinnt. Das vorliegende Lehr- und Nachschlagewerk ist als Grundlage zur umfassenden Einarbeitung in die interaktive Programmiersprache APL für Studenten der Wirtschaftswissenschaften und für Endbenutzer in der betrieblichen Datenverarbeitung konzipiert. Der anwendungsbezogene Aufbau dieses Buches basiert zum einen auf Vorlesungen und Übungen für APL, die die Verfasser für Studenten der Wirtschaftswissenschaften an der Universität Essen GHS seit mehreren Semestern halten. Zum anderen konnten zahlreiche Erfahrungen aus dem praktischen Einsatz von APL in verschiedenen Unternehmen verwertet werden.

Der erste Teil des Buches behandelt die historische Entwicklung, besondere Eigenschaften und die Anwendungsmöglichkeiten von APL. Teil 2 beinhaltet die Grundlagen der Sprache APL sowie eine systematische Darstellung aller sprachеigenen Funktionen und Operatoren. Der einführende Block in die Sprachbestandteile von APL wird mit Teil 3 durch die Beschreibung der Erstellung, Verwaltung und Validierung von APL-Programmen abgeschlossen.
Den zweiten großen Block des Buches bildet der vierte Teil, in dem die Lösung ausgewählter Problembeispiele verschiedener wirtschaftswissenschaftlicher Anwendungsbereiche mit APL gezeigt wird. Dabei werden fortgeschrittene Programmiertechniken und die Verwendung von APL-Standardsoftware demonstriert.

Aufgaben, die den Inhalt einzelner Kapitel bzw. Abschnitte beinhalten, erleichtern das schrittweise Durcharbeiten des Buches. Im Anhang befinden sich neben den Aufgabenlösungen Übersichten zu den wesentlichen APL-Funktionen/ -Operatoren sowie zu Systemvariablen, -funktionen und Steuerbefehlen. Bei der Benutzung dieses Buches als Nachschlagewerk sind besonders gekennzeichnete Hinweise im Text und ein ausführliches Stichwortverzeichnis hilfreich.

Allen, die uns bei der Erstellung des Buches unterstützt haben, möchten wir an dieser Stelle herzlich danken. Ganz besonders danken wir für die schreibtechnische Umsetzung der verschiedenen Entwicklungsstufen des Manuskriptes Frau Erika Vorholt, Frau Regine Weiß und Frau Karin Edelmann, die außerdem als Ehefrau einer besonderen Belastung ausgesetzt war. Nicht zuletzt gilt unser Dank Herrn Dipl.-Volksw. Martin Weigert für die bereitwillige Betreuung in allen redaktionellen Fragen.

Michael A. Curth
Helmut Edelmann

Teil 1
APL - Eine interaktive Programmiersprache

1. Historische Entwicklung

Die vierziger Jahre werden heute als begründende Dekade der Computertechnologie angesehen, innerhalb derer vier Computergenerationen anhand ihrer Schaltelemente unterschieden werden (vgl. Abb. 1). Strittig ist dabei, ob der Beginn der 1. Computergeneration mit der Fertigstellung der Z3 von Konrad Zuse im Jahre 1941 zu sehen ist, die auf Relaisschaltungen beruhte, oder mit der 1946 gebauten ENIAC, die auf der Basis von Elektronenröhren funktionierte.

Entwicklungsstufe	Schaltelemente	Beschreibung
1. Computer-generation 1941 - 1954	Relais	1941: Z3 - 1. funktionsfähiger Computer mit Programmsteuerung durch gelochten Normalfilm
	Elektronenröhren	1946: ENIAC - Programmsteuerung durch fest verdrahtete Schalttafeln
2. Computer-generation 1955 - 1967	Transistoren	Elektronenröhren durch Transistoren ersetzt, dadurch • geringere Störanfälligkeit • kompaktere Form • höhere Geschwindigkeit
3. Computer-generation 1968 - 1977	Integrierte Schaltkreise	Integration von Bauelementen (z.B. Widerstände und Dioden) auf einem Siliziumplättchen (Chip) mit 64 Schaltkreisen auf 9 mm^2
4. Computer-generation ab 1978	Hochintegrierte Schaltkreise	64 000 Schaltkreise auf Chips mit einer Fläche von 30 mm^2

Abb. 1: Entwicklungsstufen der Computertechnologie

Für den Anwender einer EDV-Anlage ist in erster Linie die Form relevant, in der er ihr Arbeitsvorschriften mitteilen kann. Eine logisch zusammenhängende Reihe von Arbeitsvorschriften wird Programm genannt. Zum Abfassen von Programmen wurden Sprachen geschaffen, die man als Programmiersprachen bezeichnet (vgl. SCHMITZ/SEIBT 1985, S. 131). Auch im Rahmen der Programmiersprachen lassen sich historisch verschiedene Entwicklungen abgrenzen:

- Programmierung in Maschinensprache
- Programmierung in maschinenorientierten Sprachen
- Programmierung in problemorientierten Sprachen.

Die Programmierung in Maschinensprache war sehr zeitaufwendig und fehlerintensiv, weil als Arbeitsvorschriften nur Ziffern verwendet wurden, die Daten und Speicheradressen beinhalteten. Aus den Nachteilen der Anwendungen von Maschinensprachen abgeleitet, wurden die maschinenorientierten Sprachen geschaffen, die jeden Maschinenbefehl durch ein Buchstabenkürzel und die zugehörige Speicheradresse darstellten. Aus der Entwicklung problemorientierter Programmiersprachen in den 60er Jahren resultierend, ergab sich dann für den Anwender die Möglichkeit, seine Probleme in verständlicher Form zu beschreiben und zu lösen.

Neben problemorientierten Programmiersprachen wie COBOL, FORTRAN, ALGOL und PL/1 entstand auch APL, das sich ganz erheblich von den anderen Sprachen unterscheidet (vgl. Kapitel 2, Charakteristika von APL). Als geistiger Vater der Sprache APL muß Kenneth E. Iverson angesehen werden, der sie 1962 in einem Buch mit dem Titel "A Programming Language" (IVERSON, 1962) beschreibt.

Seit der ersten Implementierung von APL für das IBM-System /360 im Jahre 1966 wurden eine Reihe von Sprachentwicklungen durch namhafte DV-Hersteller wie IBM, I.P. SHARP und SPERRY UNIVAC vorangetrieben. Besondere Bedeutung kommt dabei der im Jahre 1974 von IBM vorgestellten APLSV-Version zu, die durch "Gemeinsame Variable" (Shared Variables) die Möglichkeit schafft, innerhalb von APL-Programmen auch auf Daten und Geräte außerhalb des APL-Systems zuzugreifen. Ein weiterer wichtiger Meilenstein innerhalb der APL-Sprachentwicklung wird vermutlich die ebenfalls von IBM in den letzten Jahren entwickelte APL2-Version sein, deren Hauptfortschritte in einer weiteren Integration von APL in unterstützende Partnerprogramme (wie z.B. Datenbankzugriff) und einem mächtigeren Sprachumfang liegen (vgl. JANKO, 1985, S. 26f.).

2. Charakteristika von APL

Die wesentlichen Charakteristika der Programmiersprache APL lassen sich am besten durch Vergleiche von APL mit anderen problemorientierten Programmiersprachen zeigen.

Eine Grundphilosophie fast aller Sprachen ist die Verwendung von Schlüsselworten, die dem Programm einen selbstdokumentierenden Charakter verleihen. Hieraus resultiert aber die Gefahr, daß bei komplexen und verwickelten Zusammenhängen zuviel Text einen Blick auf das Wesentliche verwehrt. In solchen Fällen hilft sich der Mensch im allgemeinen durch kurze prägnante Symbole (Beispiele aus dem täglichen Leben sind Verkehrszeichen, Notenschrift und Graphiken). Dieser Tatsache versucht APL durch eine Reihe einfacher, nichtverbaler Symbole gerecht zu werden; Schlüsselworte fehlen hier ganz.

Beispiel:

Die Anweisung innerhalb eines PL/1-Programms zur Ausgabe der Zahl 14 ist etwa:

```
PUT LIST ('14');
```

In APL würde dieselbe Anweisung lauten:

```
□←14
```

Dabei wird das Zeichen " □ " als Fenster bezeichnet, das Zeichen " ← " als Zuweisungspfeil. Interpretiert man diese Symbolik, so wird also die Zahl 14 auf ein Fenster geschrieben und damit ausgedruckt, da alles, was auf einem Fenster steht, gelesen werden kann.

Insgesamt stehen in APL über 80 solcher Symbole zur Verfügung. Da APL sowohl zur Gruppe der numerisch-wissenschaftlichen als auch zur Gruppe der kaufmännischen Sprachen gehört, decken die APL-Symbole verschiedenste Funktionen ab. Mit Hilfe der Symbole erweitert APL lediglich die aus der Mathematik bekannten Zeichen, wie z.B. "+" für Addition, " > " für Vergleich auf "Größer" und "!" für Fakultätbildung. Dabei benutzt der Anwender nur diejenigen Symbole, die er zur Lösung seiner Probleme benötigt, und kann sich so seine eigene Untermenge der Sprache definieren.

Ein weiteres typisches Merkmal von APL ist die Rekombination von Zeichen, d.h., durch Zusammensetzung verschiedener einzelner Zeichen mit eigener Bedeutung entsteht ein neues Symbol mit neuer Semantik.

Beispiel:

SYNTAX	*SEMANTIK*	*BEISPIEL*
AUS o	*MIT* ⍤⍤ *MULTIPLIZIEREN*	o2 6.283185
UND *	*POTENZIEREN*	2*3 8
ENTSTEHT ●	*LOGARITHMIEREN ZUR BASIS E*	●2.71828182 1

Abb. 2: Beispiel für die Rekombination eines Symbols

Aufgrund der mächtigen, nichtverbalen Symbole und der Rekombinationsmöglichkeit von Zeichen sind APL-Programme um ein Vielfaches kürzer als entsprechende Programme anderer Programmiersprachen. Abb. 3 mag dies an einem Beispiel durch einen Vergleich zwischen APL und PL/1 verdeutlichen, in dem ein Zahlenvektor absteigend sortiert werden soll.

PL/1 – PROBLEMLOESUNG

```
...
DO I = ANZ TO 2 BY -1;
  DO J = 1 TO I - 1;
    IF UMSATZ(I) > UMSATZ(J)
    THEN DO;
      HILF = UMSATZ(I);
      UMSATZ(I) = UMSATZ(J);
      UMSATZ(J) = HILF;
    END;
  END;
END;
...
```

APL – PROBLEMLOESUNG

```
...
UMSATZ←UMSATZ[⍒UMSATZ]
...
```

Abb. 3: Sortieren eines numerischen Vektors mit den Sprachen PL/1 und APL

In APL sind noch zwei weitere Konzepte realisiert: die Rechts-Links-Regel für die Reihenfolge der Auswertung von Ausdrücken und die dynamische Variablendefinition. Diese Konzepte werden detailliert in Teil 2 behandelt.

Sämtliche aufgezeigten Besonderheiten ermöglichen es dem Anfänger, nach kurzer Zeit der Beschäftigung mit APL zu programmieren, sowie dem fortgeschrittenen Anwender, seine Probleme ad hoc effizient zu lösen.

3. Anwendungsbereiche

APL ist ein Teilnehmersystem mit starker Dialogorientierung, d.h., der Dialog zwischen EDV und Teilnehmer gibt dem Anwender ständig die Möglichkeit, korrigierend in den Verarbeitungsablauf einzugreifen, sobald es erforderlich wird. Dadurch und durch seine besonderen Eigenschaften hat sich APL im wissenschaftlichen Bereich sowie im Rahmen der betrieblichen Datenverarbeitung eine feste Position gesichert. APL gehört heute zu den sechs am weitesten verbreiteten Programmiersprachen (vgl. JANKO, 1985, S. 17).

3.1 Einsatzmöglichkeiten im Hochschulbereich

Aufgrund der interaktiven Problemlösungsmöglichkeiten findet APL neben dem ingenieur-/naturwissenschaftlichen vor allem auch im wirtschaftswissenschaftlichen Bereich Verwendung. Auf der einen Seite wird durch die Verwendung von APL als Hilfsmittel für das Lösen betriebswirtschaftlicher Probleme ermöglicht, Fallstudien, Problembeispiele und eigene Modelle mit geringem Zeit- und Arbeitsaufwand zu erstellen (ausgewählte Beispiele dazu finden sich in Teil 4). Auf der anderen Seite ergibt sich in der Lehre ein Training im Problemlösungsverhalten, das von den Wirtschaftswissenschaften geprägt wird, und nicht von technischen und mathematischen Programmierungsdetails wie in der herkömmlichen Programmiersprachenausbildung, z.B. mit COBOL, ALGOL und PL/1. Somit sind Kenntnisse der Datenverarbeitung keine Voraussetzung für die Anwendung von APL. Es ergeben sich folgende Zielgruppen für den Einsatz von APL an der Hochschule:

- Angehörige der Hochschule, die ein geeignetes Werkzeug zur Lösung betriebswirtschaftlicher Probleme benötigen,

- alle Studenten, die Informatik bzw. Betriebs- oder Wirtschaftsinformatik als Haupt- oder Nebenfach belegt haben und in diesem Rahmen eine Programmiersprache erlernen und

- Angehörige einer Hochschule, die eine Kombination o.g. Punkte anstreben.

3.2 Einsatzmöglichkeiten in der betrieblichen Datenverarbeitung

APL findet innerhalb der betrieblichen Datenverarbeitung sowohl in der DV-Abteilung als auch in den Fachabteilungen starke Verbreitung. Der in den letzten Jahren immer größer gewordene Anwendungsstau durch die Überlastung der zentralen DV-Abteilung führte zum Konzept der Individuellen Datenverarbeitung. Hiernach werden diejenigen Aufgaben, die der Endbenutzer selbst lösen kann, von der zentralen DV-Abteilung auf die entsprechende Fachabteilung übertragen. Die zentrale Datenverarbeitung soll dann neben Steuerungs- und Koordinationsmaßnahmen nur noch die erforderlichen Werkzeuge bereitstellen sowie Schulungs- und Beratungsfunktionen wahrnehmen. So muß sie z.B. bei der Bereitstellung von APL darauf hinweisen, daß bei der Programmierung mit APL eine detaillierte Dokumentation unerläßlich ist, weil sonst durch die mögliche Anhäufung von verschiedenen Zeichen in einer Programmzeile das Programm für den Programmersteller nach kurzer Zeit nicht mehr lesbar ist.

Gerade die grundlegenden Prinzipien und Konzepte, die APL von den anderen Sprachen unterscheidet, machen es zu einem Instrument, mit dem der Endbenutzer ad hoc die Behandlung fachspezifischer Aufgaben in betrieblichen Funktionsbereichen und in der Gesamtunternehmung maschinell unterstützen kann, ohne damit die DV-Abteilung zu beauftragen. Besonders gut zeigen sich die Vorteile von APL für den Endbenutzer (s. Abb. 4) bei nichtroutinemäßigen Aufgabenstellungen.

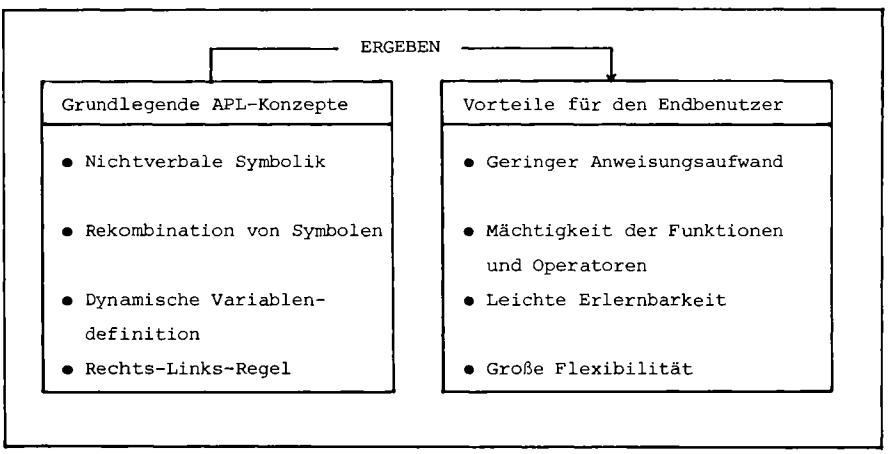
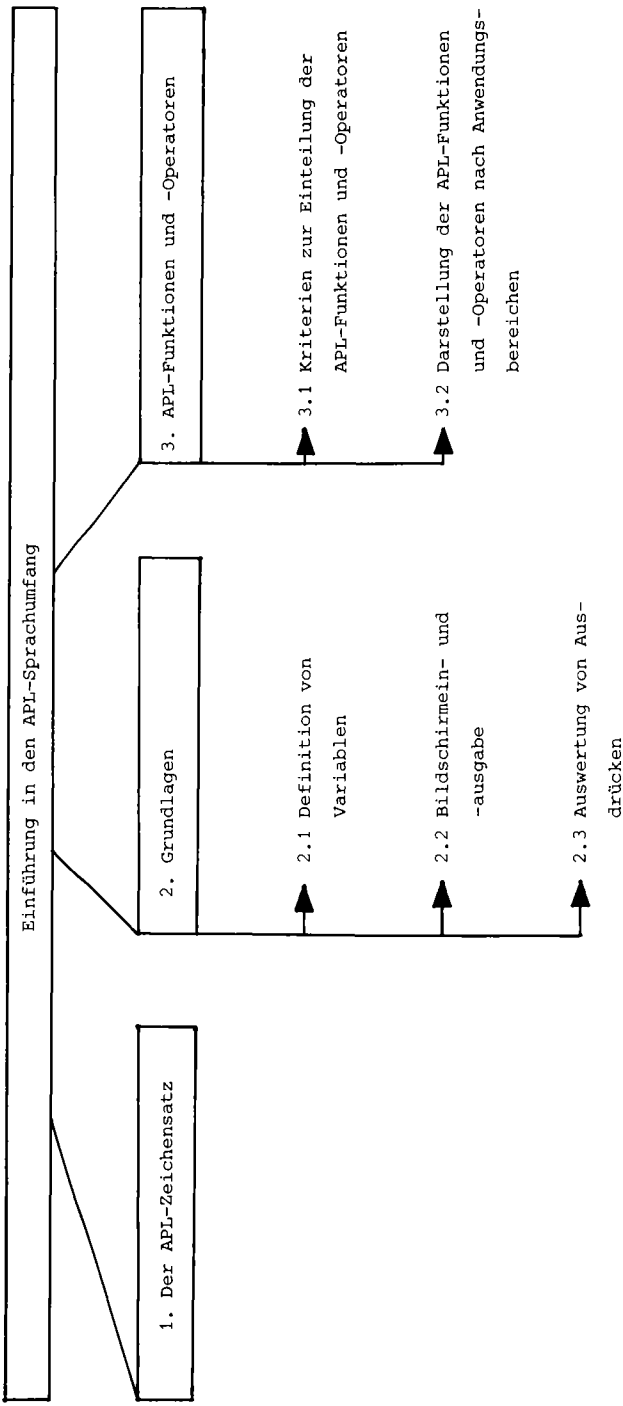


Abb. 4: Grundlegende APL-Prinzipien und Vorteile für den Endbenutzer

Hierbei handelt es sich häufig um Probleme, die im Rahmen von Planungs- und Berichtsprozessen auftreten, z.B. bei der Erstellung von OR-Modellen, statistischen Analysen und Aufbereitung von Ergebnissen für alle betrieblichen Planungsprozesse. Aufgrund dieser Ausrichtung wird APL auch als Planungssprache bezeichnet (zur Abgrenzung von Planungssprachen vgl. SCHNEIDER/SCHWAB/RENNINGER, 1983, S. 1 ff.).


Einführung in den APL-Sprachumfang
1. Der APL-Zeichensatz
2. Grundlagen
2.1 Definition von Variablen
2.2 Bildschirmein- und -ausgabe
2.3 Auswertung von Ausdrücken
3. APL-Funktionen und -Operatoren
3.1 Kriterien zur Einteilung der APL-Funktionen und -Operatoren
3.2 Darstellung der APL-Funktionen und -Operatoren nach Anwendungsbereichen

Teil 2
Einführung in den APL-Sprachumfang

1. Der APL-Zeichensatz

In Teil 1 wurde schon darauf hingewiesen, daß sich die Bestandteile der Sprache APL in zweifacher Hinsicht von denen anderer problemorientierter Programmiersprachen unterscheiden:

- einerseits existieren hier ausschließlich nonverbale Symbole, d.h. es fehlen Schlüsselworte wie z.B. GET LIST, DO, DECLARE, GO TO und END,
- andererseits lassen sich diese Symbole z.T. zu neuen Zeichen mit anderer Bedeutung zusammensetzen.

Diese Aspekte muß die Tastatur eines APL-Terminals berücksichtigen und weicht deshalb von der gewohnten Tastatur deutlich ab (vgl. Abb. 5).

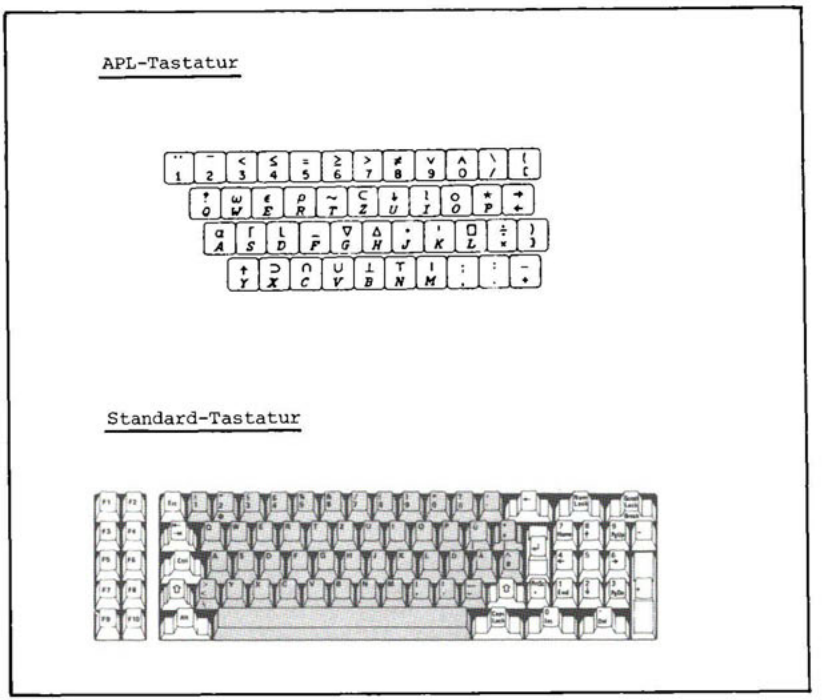


Abb. 5: Vergleich von APL-Tastatur und Standardtastatur