de Gruyter Lehrbuch

Bayer · Programmierübungen in ALGOL 60

Programmierübungen in ALGOL 60

von

Dr. Georg Bayer

Unter Mitarbeit

von *Dipl.-Ing. Lothar Potratz*
und *Dipl.-Math. Siegfried Weiß*

Walter de Gruyter & Co · Berlin · New York 1971

ISBN 3 11 003562 6

Inhaltsverzeichnis

1. Einleitung

Das Ziel der Übungen

ist es, den Lernenden durch zusätzliches Arbeitsmaterial für das Programmieren in Algol zu unterstützen; es kann neben einer oder im Anschluß an eine Einführung in Algol 60 verwendet werden. Durch die Übungen soll dem Leser der Umgang mit Algol 60 geläufig werden, und er soll die Formulierung von Algorithmen durch das Mitarbeiten an Beispielen erlernen. Letzteres Ziel steht dabei im Vordergrund.

Für den Anfänger erweist es sich immer wieder als schwierig, eine für den Rechenautomaten verarbeitungsgerechte Rechenvorschrift zur Lösung einer gegebenen Aufgabe zu finden. Die Beherrschung der Ausdrucksmöglichkeiten einer Programmiersprache, hier der „Algorithmic language Algol 60", ist einerseits die notwendige Voraussetzung zur Formulierung eines für den Rechenautomaten eingabefertigen Programms, andererseits aber enthält die Programmiersprache naturgemäß gerade diejenigen Ausdrucksformen (z. B. Variable, Felder, Prozeduren usw.), welche die Denk- und Arbeitsweise beim Programmieren bestimmen.

Wie findet man geeignete Rechenvorschriften? Ein Satz von arithmetischen Formeln zum Beispiel, welche die Existenz der Lösung eines numerischen Problems konstruktiv beweisen, ist vielleicht einfach in ein Algolprogramm umsetzbar, stellt jedoch selten einen den Fähigkeiten des Rechenautomaten gut angepaßten Algorithmus dar, ist vielleicht nicht einmal praktikabel. Ähnlich ist es mit vielen anderen Problemen, deren Lösungsverfahren aus der Theorie oder aus der Anschauung her zwar grundsätzlich bekannt sind, die jedoch erst auf eine solche Weise algorithmisch formuliert und damit programmiert werden müssen, daß sie von einem Rechenautomaten verarbeitet werden können.

Es gibt in einigen Teilgebieten (z. B. Listenverarbeitung, Numerische Verfahren) typische, modellhafte Algorithmen, die systematisch studiert, angewandt und auf andere Probleme übertragen werden können. Im Rahmen dieser Übungen kann es uns nicht darum gehen, solche Wege zu beschreiten, die einer tieferen theoretischen Fundierung bedürfen. Wir wollen vielmehr den Leser zum Nachdenken und Programmieren anregen, damit er auf diese Weise eigene Erfahrungen mit Algorithmen sammelt und anzuwenden lernt.

Aus diesem Grunde haben wir auch zu den Aufgaben immer eine solche Lösung angegeben, die zum Nachdenken anregt. Leider können solche Lösungen dem Leser als trickreich erscheinen; es war aber nicht unser Ziel, Programmiertricks vorzuführen (vgl. z. B. Lösung 2 in Aufgabe 3.4.9).

Bei der Auswahl des Übungsstoffes haben wir numerische Standardaufgaben (z. B. den Gaußalgorithmus) und Aufgaben der unmittelbaren Praxis (z. B. Be-

rechnung eines Kurbelgetriebes) nicht in Betracht gezogen: Erstere Aufgaben gehören in die Programmbibliotheken, letztere Aufgaben soll der Leser mit den erworbenen Kenntnissen angreifen können.

Anleitung zu den Übungen

Wir setzen nur voraus, daß der Leser die Programmiersprache Algol kennt. (Siehe dazu die Literaturhinweise.) Der Leser bedarf keiner darüber hinausgehenden, speziellen Vorkenntnisse, insbesondere kaum mathematischer. Es sollte möglich sein, daß er die erworbenen Erfahrungen auf seinem eigenen Anwendungsgebiet verwerten kann.

Der Übungsstoff ist so gegliedert, daß er ungefähr der Darstellung von Algol in Lehrbüchern entspricht. Die ersten Abschnitte eines jeden Kapitels dienen der Einübung der in den Überschriften genannten Begriffe; der jeweils letzte Abschnitt bringt ausgearbeitete vollständige Programmbeispiele, die den Stoff des betreffenden Kapitels und aller vorhergehenden umfassen.

Wo es notwendig erscheint, wird der Lösungsweg einer Aufgabe schrittweise dargestellt und kann vom Leser ebenfalls schrittweise „aufgedeckt" werden. Wir empfehlen dem Leser, sich unserer Lösungen erst dann zu bedienen, wenn ihm eine eigene Lösung versagt bleibt. Seine Hauptaufgabe sollte darin bestehen, seine Lösung, die nicht falsch oder schlechter als die unsere sein muß, mit der unseren schließlich kritisch zu vergleichen. Programme und Programmabschnitte sollte er mit einem kleinen Zahlenbeispiel testen, das er selbst anstelle der Rechenanlage durchführt; dabei lernt er den Ablauf des Algorithmus kennen und findet eventuell Verbesserungsmöglichkeiten. Darüber hinaus empfehlen wir selbstverständlich auch die Benutzung eines Rechenautomaten; sichtbare Erfolge sollen den Leser stärken!

Bei schwierigen syntaktischen Problemen geben wir Erklärungen und verweisen zusätzlich auf den 'Algol Report' [AR] *(Revised report on the algorithmic language Algol 60).*

Prozeduren zur Ein/Ausgabe

sind im folgenden erklärt. Wir verwenden nur wenige Prozeduren. Daß Prozeduren bei verschiedenen Compilern verschieden erklärt sein mögen, kann den Übenden nicht stören. In der Ausgabe sind die ausgearbeiteten Programmbeispiele so angelegt, daß Druckerzeilen von höchstens 80 Anschlägen beansprucht werden.

read

Funktionsprozedur vom Typ **real.**
Wirkung eines Aufrufs: Es wird ein Zahlenwert vom Eingabemedium gelesen und die Funktion erhält diesen Wert. Anschließend steht auf dem Eingabegerät die nächste Zahl zum Lesen bereit. (Die Zahlen werden in der in Algol

üblichen Form dargestellt; voneinander getrennt werden sie durch Komma oder durch zwei (und mehr) Zwischenräume oder durch Übergang zu einer neuen Zeile.)

newline(n)

Anweisungsprozedur.

Wirkung eines Aufrufs: Auf dem Ausgabemedium werden soviel Übergänge auf den Beginn einer neuen Zeile gemacht, wie der Wert des Arguments angibt. Als Spezifikation von n denke man sich

value n; **integer** n;

print(x, m, n)

Anweisungsprozedur.

Wirkung eines Aufrufs: Der Wert von x wird auf dem Ausgabemedium ausgedruckt, anschließend zwei Zwischenräume.

Falls $m = 0$, $n \neq 0$: Ausgabe in Gleitpunktform mit n-stelliger Mantisse
nach dem Schema $\pm .xxxxxx_{10} \pm xx \sqcup\sqcup$ ($n + 8$ Anschläge)

Falls $m \neq 0$, $n \neq 0$: Ausgabe mit Dezimalpunkt, und zwar m Stellen vor, n Stellen nach dem Dezimalpunkt
nach dem Schema ± xxxx.xxxxxx ⌴⌴ ($m + n + 4$ Anschläge).

Falls $m \neq 0$, $n = 0$: Ausgabe als ganze Zahl mit m Stellen
nach dem Schema ± xxxxxxxx ⌴⌴ ($m + 3$ Anschläge)

Als Spezifikation für x, m, n denke man sich

value x, m, n; **integer** m, n; **real** x;

writetext(s)

Anweisungsprozedur.

Wirkung eines Aufrufs: Die dem formellen Parameter s entsprechende Zeichenkette wird (ohne die Kettenzeichen) ausgedruckt. Als Spezifikation von s denke man sich

string s;

(Die Kettenzeichen (<*string quotes*>) sind in diesem Buch durch '(' und ')' dargestellt.)

2. Einfache Elemente von Algol

2.1. Einfache arithmetische und Boolesche Ausdrücke

Aufgabe 2.1.1.

Man ersetze folgende Ausdrücke, falls sie definiert sind, durch Zahlen:

1	3/2 × 5.0	2	2 ↑ 2 ↑ 2
3	. 5 × $_{10}$3	4	0.5 × 10 ↑ 3
5	sqrt ($_{10}$- 2)	6	15.0 - sqrt (16 ↑ 2)
7	sqrt (- 4.0)	8	ln (- 1)
9	exp (ln (1))	10	ln (exp (- 0.2))
11	abs (-$_{10}$2 × 0.0002)	12	sign (b ↑ 2 + abs(a) + 14)
13	entier (- 0.5)	14	arctan (1.0)

Lösungsweg:

Setzt man für diese leicht überblickbaren Ausdrücke die entsprechenden Zahlen ein, so spart man Rechenzeit.

5,6 Die Maschine führt eine Funktionsberechnung aus.

7,8 *sqrt* und *ln* von negativen Zahlen sind nicht definiert.

Lösungen:

1	7.5	2	16
3	500.0 *oder* .5$_{10}$3	4	500.0 *oder* .5$_{10}$3
5	0.1	6	- 1.0
7	*nicht definiert*	8	*nicht definiert*
9	1.0	10	- 0.2
11	0.02	12	1
13	- 1	14	0.785381 . . . ($\pi/4$)

Aufgabe 2.1.2.

a, b und c seien **real**-Variable mit den Werten $a = 1.0$, $b = 2.0$ und $c = 3.0$. i und j seien **integer**-Variable mit den Werten $i = 2$ und $j = 5$. Ferner ist das **integer array** ar[1 : 2] mit den Werten $ar[1]=1$ und $ar[2]=2$ gegeben.