Manfred Nagl

Einführung in die Programmiersprache Ada

Programmiersprachen

Einführung in ALGOL 68,
von H. Feldmann

Einführung in die Programmiersprache Pascal,
von K.-H. Becker und G. Lamprecht

Technisch-naturwissenschaftlicher Pascal-Trainer,
von H. Kohler

Einführung in die Programmiersprache Ada
von Manfred Nagl

Einführung in die Programmiersprache FORTRAN 77,
von G. Lamprecht

FORTRAN-Trainer,
von H. Kohler

Einführung in die Programmiersprache SIMULA,
von G. Lamprecht

Einführung in die Programmiersprache BASIC,
von W.-D. Schwill und R. Weibezahn

Einführung in die Programmiersprache COBOL,
von W.-M. Kähler

PEARL, Process and Experiment Automation Realtime Language,
von W. Werum und H. Windauer

Einführung in das Datenanalysesystem SPSS und SPSS^x,
von W.-M. Kähler

SAS für Anfänger,
von W.-M. Kähler und W. Schulte

SPSS^x für Anfänger,
von W.-M. Kähler

Vieweg

Manfred Nagl

Einführung in die Programmiersprache Ada

2., neubearbeitete und erweiterte Auflage

Springer Fachmedien Wiesbaden GmbH

1. Auflage 1982
Nachdruck 1983
2., neubearbeitete und erweiterte Auflage 1988

Ursprünglich erschienen bei Friedr. Vieweg & Sohn Verlagsgesellschaft mbH,
Braunschweig 1988

ISBN 978-3-528-13347-4 ISBN 978-3-663-14162-4 (eBook)
DOI 10.1007/978-3-663-14162-4

Vorwort zur 1. Auflage

Dieses Buch ist nicht für jemanden gedacht, der einen Ada-Übersetzer schreiben will. Es steht hier eher die ***methodische Verwendung*** der Sprache im Vordergrund als die Abgrenzung erlaubter von nichterlaubten (syntaktischen) Konstruktionen. Trotzdem soll beim Nachschlagen wegen einer Unklarheit bezüglich Ada hier eine präzise Antwort auffindbar sein. Dieses Buch ist auch ***nicht*** als ***Einführung*** in das systematische Programmieren ***für Anfänger*** gedacht. Statt dessen wendet es sich an Personen, die bereits eine gewisse Erfahrung im Programmieren mit mindestens einer höheren Programmiersprache haben. Dies kann auch FORTRAN oder COBOL sein. Für diesen Personenkreis sollte die nachfolgende Ausarbeitung sowohl als Textbuch für Programmierkurse (auch wenn diese mangels eines verfügbaren Übersetzers vielerorts noch als Trockenkurs abgehalten werden müssen) als auch zum Selbststudium geeignet sein. Die Kenntnis einer neueren höheren Programmiersprache, wie etwa PASCAL, ist zwar nicht Voraussetzung für das Lesen, doch erleichtert sie den Einstieg.

Ada entstand aufgrund einer ***Initiative*** des ***Verteidigungsministeriums*** der USA, das seine Ausgaben für Software in den Griff zu bekommen versuchte (1975 etwa 3,5 Milliarden $). Ein großer Teil dieser Ausgaben ging in den Bereich der sogenannten eingebetteten Systeme (z.B. Bordcomputer eines Flugzeugs), der sich durch Effizienzvorgaben (Speicherplatz, Laufzeit), Hardwarenähe (z.B. durch Einsatz nichtstandardmäßiger Ein-/Ausgabe) und Notwendigkeit von Konzepten für Realzeitanforderungen/Nebenläufigkeit auszeichnet. Insbesondere durch Unterstützung der Zuverlässigkeit, Portabilität und Wartungsfreundlichkeit von Software sollte eine ***Reduktion*** der ***Kosten*** erzielt werden. Das Ergebnis der Initiative ist eine universelle höhere Programmiersprache, die sich für einen breiten Einsatz eignet, nicht nur auf dem Gebiet der eingebetteten Systeme.

Ada bietet für Kontroll- und Datenstrukturen ***vielfältige Konstrukte*** an. Ada eignet sich wegen des Paketkonzepts, der Hilfsmittel zur getrennten Übersetzung und der generischen Programmeinheiten insbesondere auch für das Programmieren "großer" Probleme, die von einer Programmierermannschaft und nicht von einem einzelnen Programmierer bearbeitet werden. Hier ist die Beachtung der Erkenntnisse des Software-Engineering, insbesondere bezüglich der Programmiermethodik, unerläßlich. Wir haben deshalb in Kapitel 1 die wichtigsten Grundbegriffe aus diesem Bereich zusammengetragen. Aber auch während der restlichen Ausarbeitung wurde stets versucht, die methodische Sicht bei der Erörterung der Ada-Sprachelemente zu betonen.

Ada ist kein Forschungsergebnis, sondern das Resultat eines ***Entwicklungsprozesses***, der ***öffentlich*** und unter großer Anteilnahme der Öffentlichkeit (aus Industrie, Verwaltung und Universitäten) ablief. Insoweit faßt Ada eher bisher Bekanntes zusammen und repräsentiert den Stand der Technik auf dem Programmiersprachensektor, als neue, bahnbrechende Wege zu gehen. Aus dieser Öffentlichkeit ist bereits jetzt, vor dem Großeinsatz dieser Sprache, ein Erfolg der Ada-Unternehmung erkenntlich: Mit der Diskussion über Ada wurde auch stets über Konzepte in Programmiersprachen diskutiert und damit zur Verbreitung und Erweiterung von Kenntnis über Programmiersprachen beigetragen.

Bücher über Programmierung fallen hauptsächlich in zwei Kategorien. Die eine Kategorie enthält Einführungen in die Programmierung, heute meist mit dem Zusatz methodisch oder systematisch. Um dieser Zielsetzung gerecht zu werden, sollten hierbei alle Gesichtspunkte des Programmierens angesprochen werden, insbesondere die Verifikation, die Effizienzanalyse und die Programmiermethodik. Eine solche Darstellung ist jedoch einerseits weitgehend programmiersprachenunabhängig und andererseits nur für "kleine" Probleme machbar. Man sollte sich dabei einer einfachen, aber sauberen Programmiersprache bedienen, wie etwa PASCAL oder ELAN. Ein Buch über Ada muß schon wegen des Umfangs der Sprache, die ja auch weniger für die Ausbildung als für den Einsatz in großem Maßstab erfunden wurde, in die zweite Kategorie von Büchern fallen, nämlich in die der ***Einführungen*** in eine bestimmte ***Programmiersprache***. Hier darf dann auch einige Kenntnis über die Programmierung vorausgesetzt werden. Wir haben bei der Erläuterung von Ada versucht, die sinnvolle und methodische Verwendung dieser Sprachkonstrukte in den Vordergrund zu rücken und nicht nur die Details zu sehen, wie das eine oder andere hinzuschreiben ist.

Jede Programmiersprache, insbesondere aber eine so umfangreiche wie Ada, ist stark ***rekursiv***. Man weiß nicht, wo mit der Erläuterung begonnen werden soll, da die einzelnen Konstrukte direkt oder indirekt gegenseitig voneinander abhängen. Dadurch wird in vielen Fällen Wiederholung erzwungen. Wir haben versucht, aus dieser Not eine Tugend zu machen. So wurden die komplizierten Sprachelemente im allgemeinen stets zuerst in einfacher Form erläutert, die allgemeine Verwendung und das dahinterstehende Konzept wurde später nachgetragen. Eine Einführung in eine Programmiersprache sollte sich in ihrem Aufbau nicht am Sprachreport orientieren. Bei diesem ist es eine Zielsetzung, diese obige Rekursivität sichtbar werden zu lassen, während es hier aus didaktischen Gründen Zielsetzung ist, diese zunächst zu verbergen.

Ein einführendes Buch über eine Programmiersprache ist kein geeigneter Platz, sich kritisch mit einer Programmiersprache auseinanderzusetzen. Wir haben deshalb an einigen Stellen bewußt auf Randbemerkungen verzichtet. Die Akzeptanz einer Programmiersprache hängt nicht von ihren "technischen" Eigenschaften ab, sondern auch von dem Vorhandensein ***verständlicher*** Einführungen und Nachschlagewerke. Wir hoffen, zu diesem Gesichtspunkt der Pragmatik von Ada einen kleinen Beitrag geleistet zu haben.

Nun eine ***Übersicht***: Kapitel 1 beschäftigt sich nach einer kurzen Darstellung der Entwicklungsgeschichte von Ada mit einigen Grundbegriffen des Software-Engineering.

In Kapitel 2 wird die Notation der Syntax erläutert, und die Grundsymbole der Sprache werden aufgeführt.

Kapitel 3 ist den Elementen des Programmierens im Kleinen gewidmet und dabei hauptsächlich den Strukturen zur Ablaufkontrolle. Dazu zählen insbesondere die schon klassischen Kontrollstrukturen if-, case-, while- und for-Anweisung, die Funktionen und Prozeduren und die Sprünge. Nicht klassisch ist hier lediglich die Ausnahmebehandlung. Datenstrukturen werden hier nur andeutungsweise angesprochen, um kleine Programme oder Programmstücke formulieren zu können. Die Textein-/ausgabe wird jedoch bereits hier abgehandelt.

Kapitel 4 erläutert die Datenstrukturierung, die in neuen Programmiersprachen umfangreicher ist als die Ablaufstrukturierung. Hierzu zählen die Aufzählungstypen, die Felder und Verbunde, die in Ada in speziellen Formen als Felder mit unspezifizierten Grenzen und Verbunden mit Diskriminanten auftreten können. Erst dann folgt das Typkonzept mit der Erklärung der Begriffe Typ, Untertyp, abgeleiteter Typ. Den numerischen Typen, nämlich den ganzzahligen und den reellen Datentypen, ist hier in Ada mehr Aufmerksamkeit geschenkt worden als in anderen Programmiersprachen, insoweit als sich der Programmierer über die Definition neuer numerischer Typen unabhängig von den vordefinierten numerischen Typen machen kann. Ein ausführlicher Abschnitt über Zeiger und Haldenobjekte und ihre Verwendung in der Listenverarbeitung schließt das Kapitel ab.

Während Kapitel 3 und 4 eher die konventionellen Sprachelemente enthalten, die hauptsächlich dem Programmieren im Kleinen, d.h. der Implementierung einzelner Moduln dienen, bietet das Kapitel 5, teilweise auch 6, Elemente für das Programmieren im Großen, die während des Entwurfs bzw. während der Wartung großer Software-Systeme benötigt werden. Hierzu zählen hauptsächlich das Paketkonzept, das Konzept der Generizität, der privaten Typen und die Hilfsmittel zur getrennten Übersetzung, nämlich Bibliothekseinheiten und Untereinheiten. Da sich hinter letzterem mehr verbirgt als nur Hilfen für die textuelle Aufteilung des Programms, haben wir in einem sehr ausführlichen Abschnitt versucht, den Bezug zu Modulkonzepten herauszuarbeiten.

Kapitel 6 dient der Erklärung der nebenläufigen Programmierung. Nach Einführung der hierfür vorgesehenen Programmeinheit (der Task), der automatischen Aktivierung und der normalen Beendigung folgt das Rendezvous-Konzept als Konstrukt zur Synchronisation und zum gegenseitigen Ausschluß. Der nächste Abschnitt widmet sich der nichtdeterministischen Auswahl zwischen verschiedenen Interaktionswünschen auf der Seite der akzeptierenden Task. Verzögerung, Unterbrechung, Ausnahmebehandlung, normale und anomale Taskbeendigung schließen sich an sowie Ada-Spezielles wie Tasktypen und Entryfamilien. Ein größeres Beispiel schließt das Kapitel ab.

Das letzte Kapitel erläutert die Beziehungen zur Umwelt. Hierzu zählt die Ein-/Ausgabe mit ihrer Unterscheidung zwischen internen und externen Dateien, die neben Ein-/Ausgaberoutinen auch die Dateiverwaltung mit enthält. Schließlich gibt es noch vielfältige Möglichkeiten der Angabe von Darstellungen auf der Basismaschine.

Jedes Kapitel endet mit Übungsaufgaben, die der Leser zur Vertiefung seiner Ada-Kenntnisse benutzen kann. Ein umfangreiches Literaturverzeichnis soll Hinweise zum weiteren Studium geben. Die vordefinierten Pragmas und die vordefinierten Ausnahmen sind in Form zweier Anhänge am Ende zusammengestellt. Die vordefinierten

Attribute werden in den entsprechenden Buchabschnitten erläutert. Dem leichten Nachschlagen schließlich dient ein Anhang, der die gesamte Grammatik enthält, sowie ein ausführliches Register.

Ich möchte dieses Vorwort mit einer ***Danksagung*** an alle diejenigen abschließen, die zu der Gestalt dieses Buches in seiner jetzigen Form durch Kritik, Anregungen und Hilfe beigetragen haben. Hier sind Herr Dr. H. Hummel und Dr. Plödereder, München, sowie Herr Kollege J. Perl (jetzt Mainz) zu nennen. Zu besonderem Dank wegen sehr intensiver Mitarbeit bin ich jedoch Herrn Kollegen J. Ebert und den Herren Dr. G. Engels und Dr. W. Schäfer (alle früher in Osnabrück), sowie Herrn Dr. R. Gall, Erlangen, verpflichtet. Schließlich gilt mein Dank auch Frau K. Guss für die große Geduld und Sorgfalt beim Schreiben dieses Manuskripts sowie dem Vieweg-Verlag für das Anfertigen der Zeichnungen. Für Fehler und Mängel ist der Autor allein verantwortlich. Es kann auch kein Compiler dafür zur Rechenschaft gezogen werden, die (hoffentlich nicht zu zahlreichen) Syntaxfehler nicht erkannt zu haben.

Osnabrück, im Juni 1982 Manfred Nagl

Vorwort zur 2. Auflage

Seit dem Erscheinen der ersten Auflage dieses Buches hat sich ***einiges geändert***: Ada ist seit 1983 standardisiert worden, Ada-Veranstaltungen finden heute nicht mehr nur als Trockenkurse statt, weil in den letzten Jahren effizientere und preiswertere Ada-Compiler verfügbar wurden, in der Industrie gibt es mehr und mehr Software-Projekte, die in Ada implementiert werden, und die Initiativen und Aktivitäten um Ada herum, die man in den USA als 'Ada culture' bezeichnet, haben sich verstärkt und ausgeweitet (Validierungsprozedur, Ada-Software-Entwicklungsumgebung, STARS-Projekt usw.). Ferner, was nicht gering geschätzt werden darf, hat ein Arbeitskreis eine deutsche Ada-Terminologie erarbeitet, die in dieser 2. Auflage übernommen wurde.

Ada ist für die Erstellung großer Softwaresysteme gedacht, wo die Erkenntnisse, Konzepte und Methoden der Softwaretechnik eingesetzt werden sollten. Ada bietet hierfür vielerlei Konstrukte an. Dies betrifft insbesondere die Konstrukte für das Programmieren im Großen, die insbesondere die Wartung vereinfachen, sich aber ebenfalls auf die lebenszyklusbegleitenden Bereiche Qualitätssicherung, Projektmanagement und Dokumentation günstig auswirken. Man kann Ada deshalb ohne Übertreibung als ***die Softwaretechnik-Programmiersprache*** bezeichnen!

Ein Buch über Ada sollte diesen Zusammenhang deutlich werden lassen. Nach Kenntnis des Autors war die 1. Auflage dieses Buches das erste Buch, das diesen Zusammenhang herausgearbeitet hat. Diese Verflechtung mit der Softwaretechnik wurde in der 2. Auflage noch einmal beträchtlich verstärkt (insbesondere durch Kap. 1 und Kap. 5). Erfahrungen des Autors mit Ada-Vorlesungen und Industrie-Seminaren bestätigen, daß bei diesem didaktischen Ansatz der gedankliche Hintergrund der Ada-Konzepte vermittelt werden kann, d.h. sich die Erörterung nicht nur auf die Aufzählung der Sprachkonstrukte beschränkt.

Ada gehört zur Familie der imperativen und prozeduralen Programmiersprachen und liegt damit auf der Linie der ***"klassischen" Programmiersprachen*** (FORTRAN - Algol 60 - PL/I - Algol 68 - Pascal). In Ada sind Ideen einer Reihe anderer neuerer Programmiersprachen eingeflossen (Modula, Alphard, CLU, LIS, Mesa etc.). Diese Sprachlinie zeichnet sich durch das Bestreben nach Sicherheit (durch viele Compilezeit-Prüfungen) und Effizienz (insbesondere für den erzeugten Code) aus. Diese Ausrichtung ist ganz im Sinne der oben angedeuteten Verflechtung mit der Softwaretechnik.

Trotz dieser eher konservativen Ausrichtung von Ada hat der ***Sprachvorschlag*** nicht nur Zustimmung hervorgerufen (vgl. Literatur in Abschnitt 3). Dabei richtet sich die ***Kritik*** weniger gegen den Ansatz der Sprache an sich, sondern eher gegen einzelne Sprachkonstrukte, insbesondere jedoch gegen den Gesamtumfang der Sprache. Auf diese Kritik wird in der folgenden Darstellung nicht im einzelnen eingegangen. Obwohl sich der Autor in etlichen Punkten dieser Kritik anschließen kann, so teilt er doch die Überzeugung, daß Ada für den professionellen Bereich das beste ist, was an Programmiersprachen z.Zt. zur Verfügung steht und für den nächsten überschaubaren Zeitraum zur Verfügung stehen wird.

Neben dieser eher konservativen Ausrichtung von Ada haben in den letzten Jahren einige Programmiersprachen in der wissenschaftlichen Diskussion Bedeutung erlangt oder eine Renaissance erfahren, die völlig ***anderen Softwareerstellungs-Paradigmen*** folgen, die wir in diesem Buch nicht erläutern. Es sind dies, um nur die wichtigsten aufzuzählen: Programmieren ist Logik (logische Programmiersprachen), Programmieren ist Erstellen loser Einheiten mit Vererbung und Botschaftsaustausch (objektorientierte Sprachen), Programmieren ist Zusammenbau von Funktionen (funktionale Sprachen), Programmieren ist Spezifizieren (Spezifikationssprachen, Transformationssprachen), Programmieren ist Zusammenstellen von Fakten und Regeln (KI-Sprachen) usw. Diese Paradigmen können eines Tages für die tägliche praktische Arbeit des Software-Ingenieurs große Bedeutung erlangen. Dies ist jedoch eher für einen längerfristigen Zeitpunkt der Fall.

Eine verständliche ***Ada-Einführung*** zu schreiben, ist ***nicht einfach.*** Dies hat zum einen mit der Verflechtung mit der Softwaretechnik zu tun, da nicht bei jedem Leser Basiswissen über diesen Bereich vorausgesetzt werden kann. Zum anderen redet man bei der Erläuterung von Ada, wegen des zusammenfassenden Charakters dieser Sprache, implizit auch immer über Konzepte von (klassischen) Programmiersprachen. Ein Blick in das Inhaltsverzeichnis so manchen Buches über Programmiersprachenkonzepte (vgl. Literaturabschnitt 4) bestätigt dies. Dort findet man nämlich oft die Kapiteleinteilung des Ada-Sprachreports wieder. Letztlich erfordert das bereits im Vorwort der 1. Auflage diskutierte Rekursivitätsproblem von Programmiersprachen und der Umfang von Ada eine sorgfältige didaktische Ausarbeitung. Jede Ada-Einführung, die alle diese Probleme verschleiert und den Eindruck erweckt, als sei Ada eine Sprache für Anfänger oder im Handumdrehen zu vermitteln, ist die Zeit nicht wert, mit der man sich mit ihr beschäftigt.

Die Verflechtung mit der Softwaretechnik bzw. die Diskussion über Programmiersprachenkonzepte ***bereichert*** an sich die ***Denkwelt*** jedes Lesers und hat damit Auswirkungen auf seine tägliche Arbeit auch dann, wenn er in der nächsten Zeit keine Chance hat, Ada für ein konkretes Projekt einzusetzen. Dies gilt übrigens auch für die oben angedeuteten anderen Softwareerstellungs-Paradigmen. Eine Beschäftigung mit diesen Paradigmen bringt nicht nur Bereicherung, da sie in andere Denkwelten einführt, sondern sie läßt auch die Konzepte von Ada in einem deutlicheren Licht erscheinen.

Wenn für Ada als Programmiersprache ***Teilsprachenbildung*** verboten ist, d.h. jeder Compiler Ada voll "verstehen" können muß, so gilt dies ***nicht*** notwendigerweise ***für die Lehre.*** So ist der Pascal-Teil von Ada allein durch Kap. 3 und 4 beschrieben, wobei noch einige Abschnitte wegfallen (etwa 3.9, 4.7 und 4.8) und einige sich vereinfachen. In diesem Sinne ist Ada auch für die Anfängerausbildung einsetzbar. Ist man nur an sequentieller Programmierung interessiert (z.B. übliche mathematisch-technische Anwendungen, betriebswirtschaftliche Anwendungen, Systemprogrammierung ohne Nebenläufigkeit), so kommt noch Kap. 5 hinzu. Für numerische Anwendungen ist darüber hinaus ein detailliertes Studium der Abschnitte 4.7 und 4.8 nötig. Nur wenn man sich mit nebenläufiger und hardwarenaher Programmierung gleichzeitig beschäftigt, ist ein Studium der gesamten Sprache (bis auf die numerischen Datentypen) nötig.

Ich möchte dieses Vorwort mit einem ***Dank*** an alle diejenigen abschließen, die zu der 2. Auflage des Buches beigetragen haben. Hier ist zunächst Herr C. Lewerentz zu nennen, der das Textsystem geschrieben hat, mit dem dieses Manuskript neu erstellt wurde, und der bei Auftreten von Schwierigkeiten seines Pilotsystems nie die Geduld mit den Bedienern und seinem Produkt verlor. Insbesondere möchte ich mich bei Frau G. Holmann, Osnabrück, bedanken, die trotz des Umfangs des Manuskripts, der Schwierigkeiten mit dem Textsystem und der mehrmaligen Verbesserungen stets mit Engagement, Sorgfalt und unglaublicher Geduld dieses Manuskript nicht nur geschrieben, sondern auch in eine druckreife Form gebracht hat. Schließlich bin ich Herrn Dr. Jackel, Koblenz, und Herrn Heimann, insbesondere aber den Herren A. Schürr und B. Westfechtel für das kritische Durcharbeiten des Manuskripts und die vielen sich daraus ergebenden Verbesserungsvorschläge dankbar.

Aachen, im September 1987 Manfred Nagl

INHALT